Applied Mathematical Modelling 40 (2016) 7796–7815

Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

Two-stage assembly scheduling problem for minimizing total tardiness with setup times

Ali Allahverdi[a,*], Harun Aydilek[b], Asiye Aydilek[c]

[a] Department of Industrial and Management Systems Engineering, Kuwait University, P.O. Box 5969, Safat, Kuwait
[b] Department of Mathematics and Natural Sciences, Gulf University for Science and Technology, P.O. Box 7207, Hawally 32093, Kuwait
[c] Department of Economics and Finance, Gulf University for Science and Technology, P.O. Box 7207, Hawally 32093, Kuwait

ARTICLE INFO

Article history:
Received 8 January 2015
Revised 22 February 2016
Accepted 23 March 2016
Available online 9 April 2016

Keywords:
Algorithm
Assembly flowshop
Dominance relation
Scheduling
Total tardiness

ABSTRACT

The two-stage assembly flowshop scheduling problem has received much attention from researchers because it is applicable to many real-life environments. The objective of minimizing total tardiness is important because the fulfillment of due dates for customers must be considered when making scheduling decisions. However, the setup times were assumed to be zero in previous studies, which may not be realistic or appropriate for some scheduling environments because treating the setup times separately from the processing times increases the level of machine utilization and reduces total tardiness. In this study, we investigate the two-stage assembly flowshop scheduling problem with separate setup times to minimize the total tardiness. We propose two new algorithms and adapt four existing algorithms, which are different versions of simulated annealing, genetic, and insertion algorithms. Moreover, we formulate the problem mathematically, where we develop a dominance relation and we utilize the dominance relation in our proposed algorithms. Extensive computational experiments indicated that one of the proposed algorithms performed much better than the others on average, i.e., the error using the best algorithm was 54% to 98% less than that with the other algorithms. Computational experiments were also conducted with zero setup times in order to compare the performance of our proposed algorithms with that of the best known previously reported algorithm in the zero setup time case. When the setup times were zero, the best proposed algorithm reduced the error of the best previously reported algorithm by 48% when both algorithms were run for the same computational time. Therefore, our best proposed algorithm can be used for both zero and non-zero setup times. Hence, for the first time, the present study considers the problem with separate setup times as well as proposing much better algorithms for the zero setup time case.

1. Introduction

A set of n jobs needs to be processed on a two-stage assembly flowshop, where m machines exist at the first stage and a single assembly machine at the second stage. Each job has $m+1$ operations, where m operations need to be processed in parallel by the m machines at stage one, whereas the last operation needs to be processed by the assembly machine at

* Corresponding author. Tel.: +96524987874.
E-mail addresses: ali.allahverdi@ku.edu.kw (A. Allahverdi), aydilek.h@gust.edu.kw (H. Aydilek), aydilek.a@gust.edu.kw (A. Aydilek).

http://dx.doi.org/10.1016/j.apm.2016.03.037
S0307-904X(16)30185-8/

stage two. The last operation starts after the completion of all m operations at the first stage. This problem is known as the two-stage assembly scheduling problem.

The two-stage assembly scheduling problem has applications in many industries, and thus it has received much attention from many researchers. For example, Potts et al. [25] described an application to personal computer manufacturing, while Lee et al. [19] described an application to a fire engine assembly plant. Allahverdi and Al-Anzi [9] presented another application to the area of queries scheduling on distributed database systems. We can consider the production of an item with different components by a manufacturing firm as an assembly flowshop scheduling problem. Each component is produced by a different machine at the firm and all of the components are assembled by the assembly machine at the final stage. Therefore, the problem of two-stage assembly flowshop scheduling is realistic, practical, and it can be used in many real-life applications.

Lee et al. [19] and Potts et al. [25] addressed the two stage assembly flowshop scheduling problem by minimizing the makespan, where they proved that the problem is NP-hard in the strong sense provided that there are two or more machines at the first stage. Lee et al. [19] presented some polynomially solvable cases as well as a branch-and-bound algorithm. They also proposed a few heuristics and analyzed their error bounds. Potts et al. [25] showed that permutation schedules are dominant and presented a heuristic. Hariri and Potts [15] developed a lower bound and provided several dominance relations. In addition, Hariri and Potts [15] developed some dominance relations and lower bounds, which they incorporated in a branch-and-bound algorithm. Haouari and Daouas [16] also constructed a branch-and-bound algorithm, whereas Sun et al. [28] investigated different heuristics. However, all of these previous studies ignored the setup times. Allahverdi and Al-Anzi [8] studied the problem where setup times were treated as separate from processing times, and they proposed an efficient algorithm as well as providing two heuristics. All of these studies addressed the problem based on the makespan performance measure.

The problem with the objective of minimizing the total or mean completion time has also been investigated in previous studies. Thus, Tozkapan et al. [31] constructed a dominance relation and a lower bound, where they utilized the dominance relation and the bound in a branch-and-bound algorithm, and proposed two heuristics. Subsequently, Al-Anzi and Allahverdi [4] presented two algorithms and showed that one of the algorithms can yield the optimal solution under certain conditions. They also presented several heuristics for the problem. However, both Tozkapan et al. [31] and Al-Anzi and Allahverdi [4] ignored the setup times. Allahverdi and Al-Anzi [6] considered the problem by treating the setup times as separate from the processing times. Allahverdi and Al-Anzi [6] also proposed a few heuristics and evaluated the performance of the heuristics using randomly generated data.

Allahverdi and Al-Anzi [7] provided several heuristics for the problem with the objective of minimizing a convex combination of the makespan and mean completion time. Torabzadeh and Zandieh [30] also studied the same problem and showed that their heuristic outperforms those of Allahverdi and Al-Anzi [7]. Furthermore, Shokrollahpour et al. [27] addressed the problem and indicated that their proposed algorithm performs better than the best heuristic described by Allahverdi and Al-Anzi [7]. Recently, Tian et al. [29] proposed a particle swarm optimization (PSO) algorithm for the problem, but the setup times were treated as separate from the processing times. Tian et al. [29] showed that their PSO performs better than several other heuristics, including tabu search.

Allahverdi and Al-Anzi [9] studied the two stage assembly flowshop scheduling problem with the objective of minimizing the maximum lateness. They presented a dominance relation and proposed a few heuristics. Al-Anzi and Allahverdi [3] investigated the problem with separate setup times. In addition, Al-Anzi and Allahverdi [2] explored the problem based on the weighted sum of the makespan and maximum lateness, and they proposed some heuristics.

Maboudian and Shafaei [21] addressed the two stage assembly flowshop scheduling problem by treating the setup times as separate and sequence-dependent. They presented a nonlinear programming formulation to minimize the makespan and maximum tardiness simultaneously, where Lingo software was employed to evaluate the performance of their mathematical model. They were able to solve the problem when the number of jobs, n, did not exceed nine. In addition, Mozdgir et al. [24] considered the two-stage assembly flowshop scheduling problem with a number of non-identical machines at the second stage, by using the setup times to minimize a weighted sum of the makespan and total completion time. They described a mixed integer programming model and proposed a hybrid variable neighborhood search algorithm, where they indicated that the model could solve problems with up to eight jobs. Moreover, Dalfard et al. [13] studied the two-stage assembly flowshop scheduling problem with sequence-dependent setup times to minimize the sum of the total weighted squared tardiness, makespan, total weighted squared earliness, and number of tardy jobs. They presented an integer programming model and a genetic algorithm, where they clearly demonstrated that for large values of n ($n > 10$), it was not possible to use their model to obtain the optimal solution. According to the mathematical models developed by Maboudian and Shafaei [21], Mozdgir et al. [24], and Dalfard et al. [13], it is clear that finding an optimal solution is possible only for a small number of jobs for the two-stage assembly flowshop problem. Hence, these mathematical models are not useful in practice because real-life problems include a large number of jobs.

Many studies have aimed to minimize job completion time-related performance measures such as the makespan and total completion time in the two-stage assembly flowshop scheduling problem. However, few have considered due date-related performance measures such as minimizing the total tardiness for the two-stage assembly flowshop scheduling problem. Minimizing the total tardiness is significant in environments where there is a penalty for completing a job beyond its due date. The cost of scheduling increases as the difference between a job's completion time and its due date increases. The cost may also include loss of goodwill, damage to reputation, and a penalty. Moreover, studies of scheduling have shown

that meeting customer due dates is a key concern for many manufacturing systems because the cost of tardiness might be substantial in some manufacturing systems. Moreover, due date-related performance measures (such as total tardiness) are utilized to evaluate the performance of managers in manufacturing or production environments, such as semiconductor and automobile manufacturing industries according to Seo et al. [26].

No previous studies have aimed to minimize the total tardiness with separate setup times, although the assumption of zero setup times is not realistic in many practical environments. Allahverdi [5] reported that the setup process is not a value-added factor, and thus it needs to be considered explicitly when scheduling decisions are made to improve resource utilization, meet deadlines, increase productivity, and eliminate waste in a firm.

To the best of our knowledge, Allahverdi and Aydilek [10] is the only previous study to consider the problem with respect to minimizing the total tardiness for the two-stage assembly flowshop problem, although they assumed that the setup times are zero. In some applications, this assumption is valid, but it is not valid in many applications, e.g., plastic manufacturing [11,22]. In this study, we consider the two-stage assembly flowshop scheduling problem with non-zero setup times with the objective of minimizing the total tardiness. We propose different algorithms and compare the performance of these algorithms with each other. Moreover, an optimal solution is obtained using a mathematical formulation based on the model described by Dalfard et al. [13] in order to compare the performance of our best algorithm with the optimal solution for a small value of n.

The formulation of this problem is presented in the next section and a dominance relation is provided in the following section. Different algorithms are described in Section 4 and their evaluations are presented in Section 5. Finally, we give our conclusions in Section 6.

2. Problem formulation

n jobs are available for processing and each job comprises a set of $m+1$ operations. The first m operations are processed in parallel by m machines at stage one. The last operation is processed at stage two by the assembly machine. The setup times for each machine at stage one and for the assembly machine are treated as separate from the processing times. The following assumptions are made.

- Setup times are sequence independent.
- All jobs are available at time zero.
- Pre-emption of jobs is not allowed.
- Transportation times are zero.
- Machines are available continuously.
- Inserted idle times are not allowed.
- There is no limit on the buffer size between the two stages.

The variables used to formulate the problem are listed as follows.
Let

$p_{i,k}$: processing time of job i on machine k (at stage one), $i=1,\ldots,n$, $k=1,\ldots,m$,
$s_{i,k}$: setup time of job i on machine k (at stage one), $i=1,\ldots,n$, $k=1,\ldots,m$,
$p_{[i,k]}$: processing time of the job in position i on machine k (at stage one),
$s_{[i,k]}$: setup time of the job in position i on machine k (at stage one),
p_i: processing time of job i on assembly machine (at stage two),
s_i: setup time of job i on assembly machine (at stage two),
$p_{[i]}$: processing time of the job in position i on assembly machine (at stage two),
$s_{[i]}$: setup time of the job in position i on assembly machine (at stage two),
d_i: due date of job i,
$d_{[i]}$: due date of the job in position i of a given sequence,
$C_{[i]}$: completion time of the job in position i of a given sequence,
$T_{[i]}$: tardiness of the job in position i of a given sequence,
TT: total tardiness.

The operation time of job i, $p_{i,}$, at the second stage may start only after all of the operations, $p_{i,k}$ ($k=1,\ldots,m$), at the first stage are complete. Moreover, job i is complete once all of its operations at the first stage and its operation at the second stage are complete.

It should be noted that the setup times and processing times for the machines at the first stage can be combined because there will be no idle times on the machines at the first stage. If $sp_{i,k}$ denotes the combined setup and processing times of job i on machine k (at stage one), then $sp_{i,k}=s_{i,k}+p_{i,k}$. We consider the setup times as separate on the first stage machines and we present them explicitly to compare the performance of the proposed heuristics with a benchmark case of zero setup times from a previous study. However, the setup times generally cannot be combined with the processing times on the assembly machine at the second stage. That is because combining the setup times with the processing times at the second stage may lead to an increase in the completion time of a job, which may increase the tardiness of the job. Hence, a smaller

total tardiness can only be achieved if the setup times are considered as separate from the processing times at the second stage. In particular, better utilization of the assembly machine can only be achieved when the setup times are considered as separate from the processing times.

As reported by Allahverdi and Al-Anzi [6], the completion time of the job in position j of a given sequence, π, can be computed as:

$$C_{[j]}(\pi) = \sum_{i=1}^{j} (s_{[i]} + p_{[i]}) + \Delta_j , \tag{1}$$

where

$$\Delta_j = \max\{0, \sigma_1, \sigma_2, \ldots, \sigma_j\} \tag{2}$$

and where

$$\sigma_j = \max_{k=1,\ldots,m} \left\{ \sum_{i=1}^{j} (s_{[i,k]} + p_{[i,k]}) \right\} - \sum_{i=1}^{j-1} (s_{[i]} + p_{[i]}) - s_{[j]}. \tag{3}$$

Hence, the tardiness of the job in position j can be written as follows.

$$T_{[j]}(\pi) = \max \left\{ \sum_{i=1}^{j} (s_{[i]} + p_{[i]}) + \Delta_j - d_{[j]}, 0 \right\} \tag{4}$$

Thus, the total tardiness of the sequence π is

$$TT(\pi) = \sum_{i=1}^{n} T_{[i]}(\pi) \tag{5}$$

3. Dominance relation

In this section, we develop a dominance relation for minimizing the total tardiness. The dominance relation is utilized in the algorithms presented in the next section.

Let π_1 and π_2 be two sequences such that $\pi_1 = (\ldots, i, j, \ldots)$, where job i is in an arbitrary position τ and job j is in position $\tau + 1$. The sequence π_2 is exactly the same as π_1, except that job j is in position τ and job i is in position $\tau + 1$, i.e., the jobs in positions τ and $\tau + 1$ are interchanged so that $\pi_2 = (\ldots, j, i, \ldots)$. The following lemmas are used in the proof of Theorem 1.

Lemma 1. *$T_{[r]}(\pi_2) = T_{[r]}(\pi_1)$ for $r = 1, \ldots, \tau - 1$.*

Proof. The two sequences, π_1 and π_2, have the same jobs in all of the positions of $1,\ldots,\tau - 1$.

Lemma 2. *$T_{[r]}(\pi_2) \le T_{[r]}(\pi_1)$ for $r = \tau + 2,\ldots,n$ if $\max\{\sigma_\tau(\pi_2), \sigma_{\tau+1}(\pi_2)\} \le \max\{\sigma_\tau(\pi_1), \sigma_{\tau+1}(\pi_1)\}$.*

Proof. The two sequences, π_1 and π_2, have the same job in all positions, except for positions τ and $\tau + 1$. Therefore, from Eq. (3), it follows that $\sigma_r(\pi_2) = \sigma_r(\pi_1)$ for $r = 1,\ldots,\tau-1,\tau+2,\ldots,n$.

Using Eqs. (2)–(4), $T_{[r]}(\pi_2) \le T_{[r]}(\pi_1)$, if $\max\{\sigma_\tau(\pi_2), \sigma_{\tau+1}(\pi_2)\} \le \max\{\sigma_\tau(\pi_1), \sigma_{\tau+1}(\pi_1)\}$ ■

Theorem 1. *Consider a two-stage assembly flowshop scheduling problem and assume that two adjacent jobs i and j satisfy the following conditions.*

(i) $s_{j,k} + p_{j,k} \le s_{i,k} + p_{i,k} \le p_j + s_i \; \forall \, k$
(ii) $s_j + p_j + d_i \le s_i + p_i + d_j$
(iii) $s_i \le s_j$
(iv) $d_j \le d_i$

Then, $TT(\pi_2) \le TT(\pi_1)$, which implies that job j should precede job i in order to minimize the total tardiness.

Proof of Theorem 1. Consider the two sequences π_1 and π_2, as described earlier. From Eq. (3),

$$\sigma_\tau(\pi_1) = \max_{k=1,\ldots,m} \left\{ \sum_{r=1}^{\tau-1} (s_{[r,k]} + p_{[r,k]}) + s_{i,k} + p_{i,k} \right\} - \sum_{r=1}^{\tau-1} (s_{[r]} + p_{[r]}) - s_i \tag{6}$$

$$\sigma_\tau(\pi_2) = \max_{k=1,\ldots,m}\left\{\sum_{r=1}^{\tau-1}(s_{[r,k]}+p_{[r,k]}) + s_{j,k} + p_{j,k}\right\} - \sum_{r=1}^{\tau-1}(s_{[r]}+p_{[r]}) - s_j \tag{7}$$

$$\sigma_{\tau+1}(\pi_1) = \max_{k=1,\ldots,m}\left\{\sum_{r=1}^{\tau-1}(s_{[r,k]}+p_{[r,k]}) + s_{i,k} + p_{i,k} + s_{j,k} + p_{j,k}\right\} - \sum_{r=1}^{\tau-1}(s_{[r]}+p_{[r]}) - s_i - p_i - s_j \tag{8}$$

$$\sigma_{\tau+1}(\pi_2) = \max_{k=1,\ldots,m}\left\{\sum_{r=1}^{\tau-1}(s_{[r,k]}+p_{[r,k]}) + s_{j,k} + p_{j,k} + s_{i,k} + p_{i,k}\right\} - \sum_{r=1}^{\tau-1}(s_{[r]}+p_{[r]}) - s_j - p_j - s_i. \tag{9}$$

Condition (i) in the theorem states that

$$s_{j,k} + p_{j,k} \le s_{i,k} + p_{i,k},$$

which implies that

$$\sum_{r=1}^{\tau-1}(s_{[r,k]}+p_{[r,k]}) + s_{j,k} + p_{j,k} \le \sum_{r=1}^{\tau-1}(s_{[r,k]}+p_{[r,k]}) + s_{i,k} + p_{i,k}\,\forall\, k,$$

which also implies that

$$\max_{k=1,\ldots,m}\left\{\sum_{r=1}^{\tau-1}(s_{[r,k]}+p_{[r,k]}) + s_{j,k} + p_{j,k}\right\} \le \max_{k=1,\ldots,m}\left\{\sum_{r=1}^{\tau-1}(s_{[r,k]}+p_{[r,k]}) + s_{i,k} + p_{i,k}\right\}$$

By condition (iii) in the theorem, the last inequality implies that

$$\max_{k=1,\ldots,m}\left\{\sum_{r=1}^{\tau-1}(s_{[r,k]}+p_{[r,k]}) + s_{j,k} + p_{j,k}\right\} - s_j \le \max_{k=1,\ldots,m}\left\{\sum_{r=1}^{\tau-1}(s_{[r,k]}+p_{[r,k]}) + s_{i,k} + p_{i,k}\right\} - s_i. \tag{10}$$

Moreover, condition (i) in the theorem also states that

$$s_{i,k} + p_{i,k} - p_j - s_i \le 0 \forall k,$$

which implies that

$$\sum_{r=1}^{\tau-1}(s_{[r,k]}+p_{[r,k]}) + s_{j,k} + p_{j,k} + s_{i,k} + p_{i,k} - p_j - s_i - s_j \le \sum_{r=1}^{\tau-1}(s_{[r,k]}+p_{[r,k]}) + s_{j,k} + p_{j,k} - s_j\,\forall\, k,$$

which also implies that

$$\max_{k=1,\ldots,m}\left\{\sum_{r=1}^{\tau-1}(s_{[r,k]}+p_{[r,k]}) + s_{j,k} + p_{j,k} + s_{i,k} + p_{i,k}\right\} - s_j - p_j - s_i \le$$
$$\max_{k=1,\ldots,m}\left\{\sum_{r=1}^{\tau-1}(s_{[r,k]}+p_{[r,k]}) + s_{j,k} + p_{j,k}\right\} - s_j. \tag{11}$$

Inequalities (10) and (11) imply that

$$\max\left[\begin{array}{l}\max\limits_{k=1,\ldots,m}\left\{\sum\limits_{r=1}^{\tau-1}(s_{[r,k]}+p_{[r,k]}) + s_{j,k} + p_{j,k}\right\} - s_j, \\ \max\limits_{k=1,\ldots,m}\left\{\sum\limits_{r=1}^{\tau-1}(s_{[r,k]}+p_{[r,k]}) + s_{j,k} + p_{j,k} + s_{i,k} + p_{i,k}\right\} - s_j - p_j - s_i\end{array}\right] \le$$
$$\max\left[\begin{array}{l}\max\limits_{k=1,\ldots,m}\left\{\sum\limits_{r=1}^{\tau-1}(s_{[r,k]}+p_{[r,k]}) + s_{i,k} + p_{i,k}\right\} - s_i, \\ \max\limits_{k=1,\ldots,m}\left\{\sum\limits_{r=1}^{\tau-1}(s_{[r,k]}+p_{[r,k]}) + s_{i,k} + p_{i,k} + s_{j,k} + p_{j,k}\right\} - s_i - p_i - s_j\end{array}\right],$$

which is also equal to the following.

$$\max\left[\begin{array}{l}\max\limits_{k=1,\ldots,m}\left\{\sum\limits_{r=1}^{\tau-1}(s_{[r,k]}+p_{[r,k]})+s_{j,k}+p_{j,k}\right\}-\sum\limits_{r=1}^{\tau-1}(s_{[r]}+p_{[r]})-s_j,\\ \max\limits_{k=1,\ldots,m}\left\{\sum\limits_{r=1}^{\tau-1}(s_{[r,k]}+p_{[r,k]})+s_{j,k}+p_{j,k}+s_{i,k}+p_{i,k}\right\}-\sum\limits_{r=1}^{\tau-1}(s_{[r]}+p_{[r]})-s_j-p_j-s_i\end{array}\right]\leq$$
$$\max\left[\begin{array}{l}\max\limits_{k=1,\ldots,m}\left\{\sum\limits_{r=1}^{\tau-1}(s_{[r,k]}+p_{[r,k]})+s_{i,k}+p_{i,k}\right\}-\sum\limits_{r=1}^{\tau-1}(s_{[r]}+p_{[r]})-s_i,\\ \max\limits_{k=1,\ldots,m}\left\{\sum\limits_{r=1}^{\tau-1}(s_{[r,k]}+p_{[r,k]})+s_{i,k}+p_{i,k}+s_{j,k}+p_{j,k}\right\}-\sum\limits_{r=1}^{\tau-1}(s_{[r]}+p_{[r]})-s_i-p_i-s_j\end{array}\right] \tag{12}$$

Hence, from Eqs. (6)–(9) and inequality (12), it follows that

$$\max\{s_t(p_2), s_{t+1}(p_2)\}\leq\max\{s_t(p_1), s_{t+1}(p_1)\} \tag{13}$$

By the definition, we have the following.

$$T_{[\tau]}(\pi_1)=\max\left\{\sum_{r=1}^{\tau-1}(s_{[r]}+p_{[r]})+s_i+p_i+\max\{\Delta_{\tau-1}(\pi_1),\sigma_\tau(\pi_1)\}-d_i,0\right\} \tag{14}$$

$$T_{[\tau]}(\pi_2)=\max\left\{\sum_{r=1}^{\tau-1}(s_{[r]}+p_{[r]})+s_j+p_j+\max\{\Delta_{\tau-1}(\pi_2),\sigma_\tau(\pi_2)\}-d_j,0\right\} \tag{15}$$

$$T_{[\tau+1]}(\pi_1)=\max\left\{\sum_{r=1}^{\tau-1}(s_{[r]}+p_{[r]})+s_i+p_i+s_j+p_j+\max\{\Delta_{\tau-1}(\pi_1),\sigma_\tau(\pi_1),\sigma_{\tau+1}(\pi_1)\}-d_j,0\right\} \tag{16}$$

$$T_{[\tau+1]}(\pi_2)=\max\left\{\sum_{r=1}^{\tau-1}(s_{[r]}+p_{[r]})+s_j+p_j+s_i+p_i+\max\{\Delta_{\tau-1}(\pi_2),\sigma_\tau(\pi_2),\sigma_{\tau+1}(\pi_2)\}-d_i,0\right\} \tag{17}$$

From inequality (13) and Eqs. (16) and (17), and condition (iv) in the theorem, $d_j\leq d_i$,

$$T_{[\tau+1]}(\pi_2)\leq T_{[\tau+1]}(\pi_1). \tag{18}$$

From Eqs. (2) and (3) and inequality (10), it follows that

$$\max\{\Delta_{\tau-1}(\pi_2),\sigma_\tau(\pi_2)\}\leq\max\{\Delta_{\tau-1}(\pi_1),\sigma_\tau(\pi_1)\}. \tag{19}$$

From Eqs. (14) and (15), inequality (19), and condition (ii) in the theorem, it also follows that

$$T_{[\tau]}(\pi_2)\leq T_{[\tau]}(\pi_1) \tag{20}$$

$$T_{[\tau]}(\pi_2)+T_{[\tau+1]}(\pi_2)\leq T_{[\tau]}(\pi_1)+T_{[\tau+1]}(\pi_1). \tag{21}$$

Then, from Eq. (13), inequality (21), and Lemmas 1 and 2, it follows that

$$TT(\pi_2)\leq TT(\pi_1)$$

□

Corollary 1. Consider a two-stage assembly flowshop scheduling problem where the setup times are zero. If two adjacent jobs i and j satisfy the following conditions

(i) $p_{j,k} \le p_{i,k} \le p_j \; \forall \, k$
(ii) $p_j + d_i \le p_i + d_j$
(iii) $d_j \le d_i$,

then $TT(\pi_2) \le TT(\pi_1)$ which implies that job j should precede job i to minimize the total tardiness.

Proof of Corollary 1. The proof follows from Theorem 1 where the setup times are replaced with zero.

4. Algorithms

There are no previous reports of the two-stage assembly flowshop scheduling problem with setup times where the objective is minimizing the total tardiness. However, for the problem without setup times, Allahverdi and Aydilek [10] proposed a simulated annealing algorithm (SA) and a SA combined with an insertion algorithm called PSA. They showed that their algorithms performed considerably better than several other algorithms. Therefore, we adapt these algorithms to our problem with setup times, which we denote as AA-SA and AA-PSA. We use the algorithms AA-SA and AA-PSA as benchmarks to compare with the algorithms proposed in Section 4.4. Furthermore, in the computational experiments, we consider the case with zero setup times as well as using AA-SA and AA-PSA reduced to SA and PSA, respectively. The computational experiments conducted with zero setup times indicated that the algorithms presented in Section 4 perform better than the best previously reported algorithms. Therefore, in the present study, we consider the setup times as separate, but we also propose new and much better performing algorithms for the zero setup time case.

No algorithm has been proposed previously for the two-stage assembly flowshop problem with separate setup times, but two algorithms were proposed recently for related problems, i.e., for the special case of our problem. Therefore, in addition to AA-SA and AA-PSA, we consider two further algorithms to compare with our algorithms. We summarize the AA-SA and AA-PSA algorithms in Section 4.1, and the other two algorithms in Sections 4.2 and 4.3. Finally, we present the proposed algorithms in Section 4.4.

4.1. AA-SA and AA-PSA algorithms

Allahverdi and Aydilek [10] proposed an SA algorithm for the two-stage assembly flowshop scheduling problem with zero setup times to minimize the total tardiness. Their SA is a standard SA except that it finds an initial solution according to on one of their lemmas. They showed that their SA performs better than several other algorithms, including a genetic algorithm and cloud theory-based SA. We adapt their SA algorithm to our problem with separate setup times and we denote it as AA-SA. As noted earlier, AA-SA reduces to SA when the setup times are set at zero.

Allahverdi and Aydilek [10] also proposed an insertion algorithm, which they combined with their SA algorithm and it is called PSA. They showed that their PSA algorithm improves the performance of their SA by more than 60%. We adapt their combined algorithm PSA to our problem with separate and non-zero setup times, and we denote it as AA-PSA. Similarly, the algorithm AA-PSA reduces to PSA for the case when the setup times are set at zero.

4.2. Algorithm LZ-HGA

The two-stage assembly flowshop problem reduces to the regular two-machine flowshop scheduling problem when there is only one machine at stage one. Li and Zhang [20] proposed a hybrid genetic algorithm for the m-machine flowshop scheduling problem with separate setup times to minimize the total tardiness. We adapt their algorithm to our problem for three reasons. First, their two-machine problem is a special case of our problem. Second, they also treat setup times as separate. Third, they showed that their algorithm performs well at minimizing the total tardiness, which is also the objective of the present study. Their algorithm differs from the standard genetic algorithm mainly in terms of the crossover operator, selection strategy, and local search method. Li and Zhang [20] constructed probability-based multi-point crossover to inherit good genes and eliminate bad genes in the offspring with high probabilities. They used the n-tournament method as the selection strategy to decrease the computational time. Moreover, they developed three versions of a local search method in their genetic algorithm to improve their results. The details of their algorithm are not given here for the sake of brevity. The number of generations in the algorithm is set so the computational times of the algorithms are the same. We denote their adapted algorithm as LZ-HGA.

4.3. Algorithm MZY-SA

The two-stage flowshop problem is a special case of the two-stage assembly flowshop problem, i.e., when $m = 1$ at stage one. Mousavi et al. [23] proposed a SA algorithm for a hybrid flowshop scheduling problem with separate setup times to minimize the total tardiness. We adapt their algorithm to our problem for three reasons. They also treat setup times separately and consider the weighted sum of the total tardiness and makespan. Their problem reduces to the single performance

measure of total tardiness when the weight of the makespan is set at zero. Moreover, a flowshop is a special case of a hybrid flowshop where there is only one machine at each stage.

The algorithm proposed by Mousavi et al. [23] is a standard SA algorithm but with three differences. The first difference is the normalization of the objective function. The second difference is the neighborhood search mechanism employed for updating the current solution to obtain better results. The third difference is the incorporation of a local search. The details of the algorithm proposed by Mousavi et al. [23] are not given for the sake of brevity. We set the number of repetitions in the algorithm so the computational times of the algorithms are equal. We adapt their algorithm to our problem and denote it by MZY-SA.

4.4. Proposed algorithms N-SA and N-PSA

We propose two algorithms. The first algorithm is a SA algorithm that incorporates the following features:

(i) a problem-specific initial sequence,
(ii) Theorem 1 developed in Section 3, and
(iii) a combination of two operators for sequence updating.

We denote the first algorithm as N-SA. The second algorithm is an insertion algorithm that uses the output solution of N-SA and Theorem 1. We denote the second algorithm as N-PSA. We describe N-SA algorithm first before explaining N-PSA.

We need an initial sequence to initiate the SA algorithm. In general, the choice of the initial sequence affects the algorithm's performance. To construct an initial sequence, we first find an aggregate processing time for job i as $AP_0(i) = \max\{\max_{j=1,\ldots,m}(p_{i,j}+s_{i,j}), (s_i+p_i)\}$, and we then apply the shortest processing time rule to AP_0. Hence, we obtain the initial sequence s_0. We also considered other aggregate processing times to obtain the initial sequences, e.g., $AP_1(i) = \{\max_{j=1,\ldots,m}(p_{i,j}+s_{i,j})+(s_i+p_i)\}$, $AP_2(i) = \max_{j=1,\ldots,m}(p_{i,j}+s_{i,j})$ and $AP_3(i) = \max\{(\max_{j=1,\ldots,m}(p_{i,j}+s_{i,j})+\min_{j=1,\ldots,m}(p_{i,j}+s_{i,j}))/2, (s_i+p_i)\}$, as well as using the earliest due date sequence. We evaluated the performance of AP_r sequences based on randomly generated sample problems (given in Section 5) for different combinations of n, m, k, T, and R, as explained in the next section. Based on the computational experiments, we observed that the sequence obtained from AP_0 generally performed better than the others at a significance level of 0.05. Hence, we use s_0 as the initial sequence in Step 2 of the N-SA algorithm. Moreover, SA has certain input parameters. The parameters of the SA algorithm, i.e., the initial temperature T_i, final temperature T_f, cooling factor cf, and the number of repetitions N_r, were taken from Allahverdi and Aydilek [10] because they considered the same problem, but without setup times. We consider the setup times separately, so we fine-tuned the parameters around the initial parameters by considering three values: a smaller value, the same value, and a larger value. We randomly generated the data, as explained in Section 5. We considered the values as: 0.05, 0.10, and 0.15 for T_i; 0.00,005, 0.0001, and 0.0002 for T_f; 0.975, 0.980, and 0.985 for cf; and 45, 50, and 55 values for N_r. The computational experiments demonstrated that the N-SA algorithm performed better on average with values of: $T_i = 0.15$, $T_f = 0.0002$, $cf = 0.975$, and $N_r = 50$. The steps in the N-SA algorithm are described as follows.

N-SA algorithm

1. Set the initial parameters T_i, T_f, cf, and N_r.
2. Set the initial sequence s_0, as explained above.
3. Apply Theorem 1 to all of the consecutive pairs of s_0 and obtain s_a
4. Set the temperature $temp = T_i$ and sequence $s = s_a$.
5. Set $j = 1$
6. Generate two random integers, k and l, between 1 and n. Swap the jobs in positions k and l of sequence s to obtain sequence s_s.
7. Insert the job in position k of sequence s to position l and designate this new sequence as s_i.
8. Evaluate $F = TT(s)$, $F_s = TT(s_s)$ and $F_i = TT(s_i)$, where $TT(\pi)$ is the total tardiness of a sequence π.
9. Let $F_m = \min\{F_s, F_i\}$ and s_m is the corresponding sequence.
10. If $F_m < F$, then update s with s_m; otherwise, update sequence s with s_m and probability $e^{-\frac{dif}{temp}}$ w, where $dif = (F_m - F)/F$.
11. Set $j = j+1$. If $j = N_r + 1$, go to Step 12; otherwise, go to Step 6.
12. Update the temperature, $temp = temp * cf$.
13. If $temp < T_f$, go to Step 14; otherwise, go to Step 5.
14. s is the output sequence.

The N-PSA algorithm has two inputs: an initial sequence and the number of iterations. We set the initial sequence as the output sequence of N-SA. However, any other sequence could be used. The number of iterations was set at 12 to ensure a fair comparison between N-PSA and AA-PSA. In general, as the number of iterations increases, the algorithm performs better but at the expense of a greater computation time. We observed that setting a number greater than 12 did not improve the results significantly. The steps of the N-PSA algorithm are described as follows.

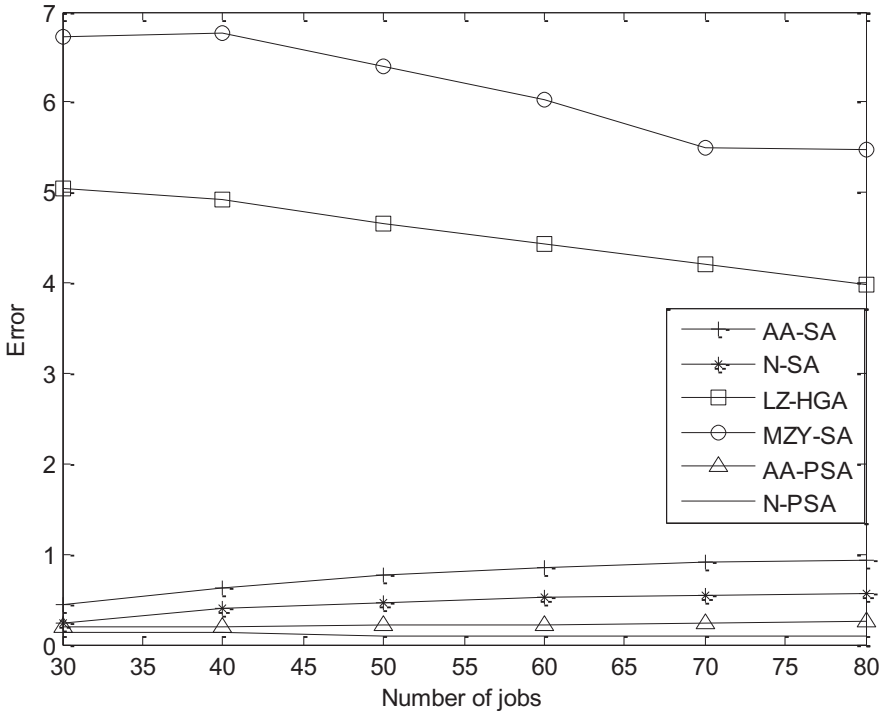

Fig. 1. Error versus number of jobs for all six algorithms.

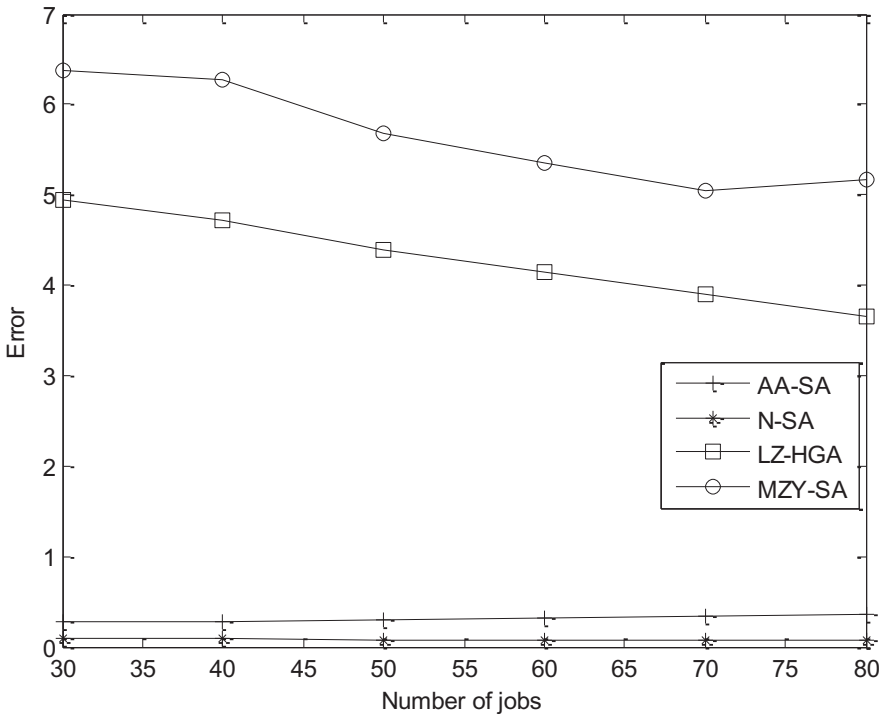

Fig. 2. Error versus number of jobs.

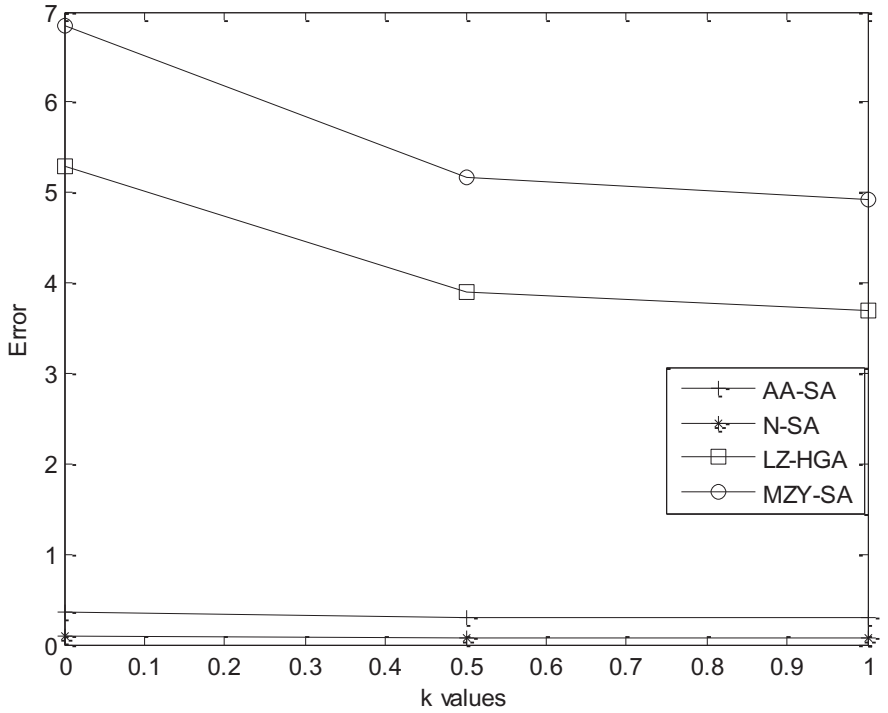

Fig. 3. Error versus k values.

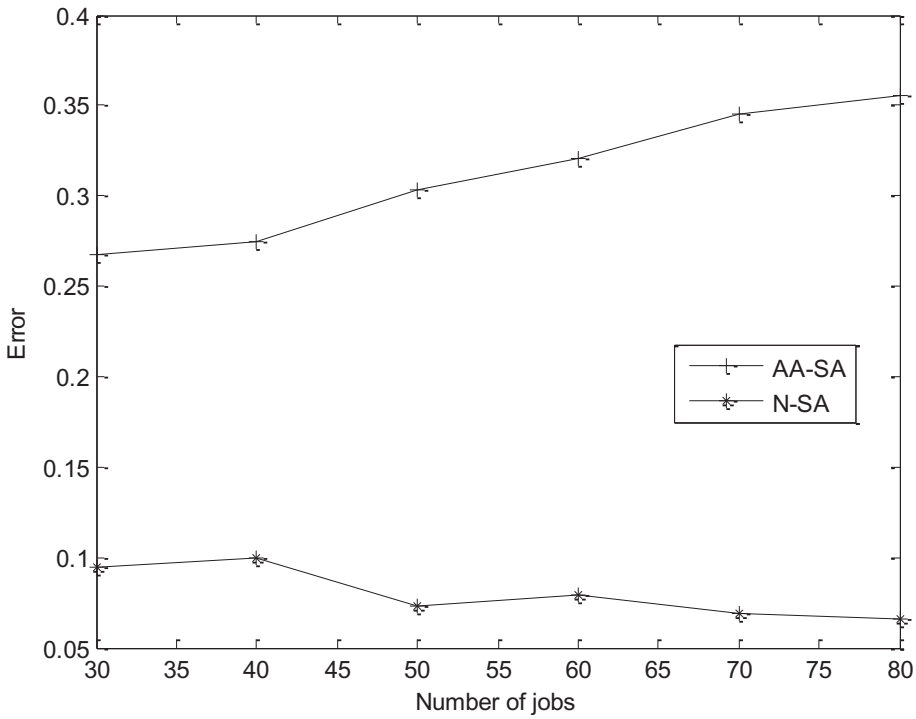

Fig. 4. Error versus number of jobs, but only for AA-SA and N-SA.

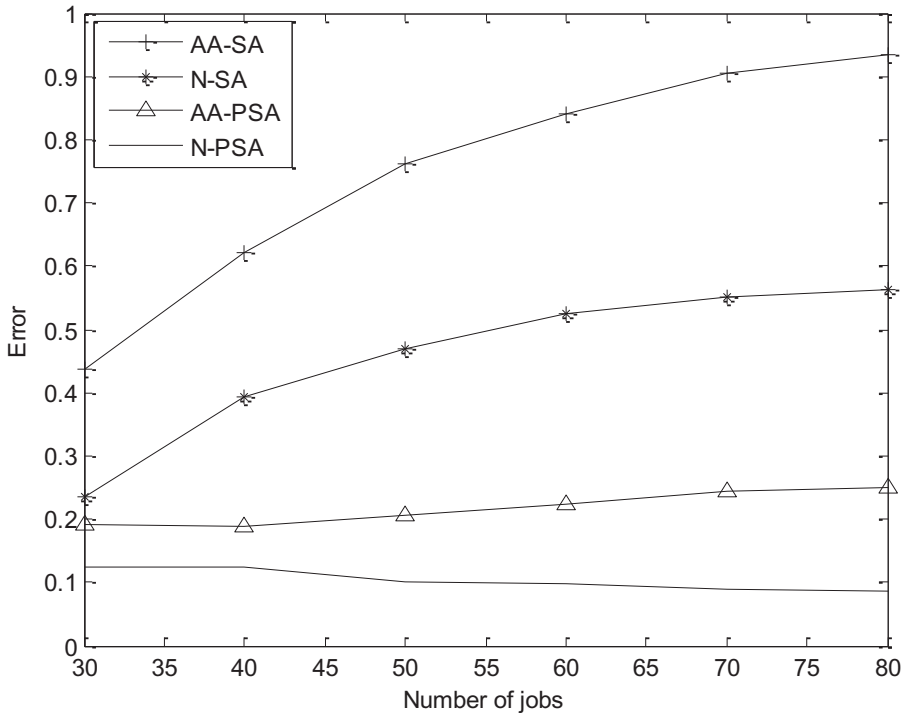

Fig. 5. Error versus number of jobs.

N-PSA algorithm

1. Take the output of N-SA as the initial sequence s_{in} (it should be noted that any other sequence could generally be used).
2. Apply Theorem 1 to all of the consecutive pairs of s_{in} and obtain sa_{in}
3. Set a value for the number of iterations L; let $s_b = sa_{in}$, and let $T_{temp} = TT(s_b)$.
4. Set $r = 1$ and $s_r = s_b$.
5. Set $j = 1$.
6. Set $p = 1$.
7. Take the job in position j of sequence s_r and insert it into position p, $p \neq j$, and designated it as s_{temp}. If $p = j$, then go to Step 10.
8. Compute $TT(s_{temp})$.
9. If $TT(s_{temp}) <= T_{temp}$,
then set $s_b = s_{temp}$, and $T_{temp} = TT(s_{temp})$.
10. Set $p = p+1$, if $p = n+1$, go to Step 11; otherwise, go to Step 7.
11. Set $j = j+1$, if $j = n+1$, go to Step 12; otherwise, go to Step 6.
12. Set $r = r+1$, if $r = L+1$, go to Step 13; else if $TT(s_r) > TT(s_b)$, set $s_r = s_b$ and go to Step 5; else go to Step 13.
13. Set $\theta = s_b$.
14. Set $i = 1$.
15. Apply a pairwise swap by swapping the two jobs in positions i and $i+1$ of sequence θ, and designated the resulting sequence after the exchange as ϕ.
If $TT(\phi) < TT(\theta)$, then let $\theta = \phi$.
16. Set $i = i+1$, if $i = n$ and go to Step 17; otherwise, go to Step 15
17. The solution is sequence θ.

It should be noted that the complexity of the N-PSA algorithm is $O(L^* n^2)$.

5. Evaluation of the algorithms

We present the evaluations of the existing and proposed algorithms in this section. The computations were implemented on a PC with an Intel Core i7-3520 M CPU processor at 2.9 GHz with 8 GB RAM.

We randomly generated job processing times on all of the machines at the first and second stages from a uniform distribution of U(1100). Hall and Posner [14] recommended this distribution because it has a large variance and many previous studies have used this distribution in their experiments. Similarly, we generated setup times from a uniform distribution of U(0, k^*100), where the parameter k denotes the expected ratio of the setup times relative to the processing times. For example, when $k = 1$, the setup and processing times are expected to be equal, whereas for $k = 0.50$, the setup time is expected to be 50% of the processing time. Clearly, when $k = 0$, the problem reduces to the zero setup time problem. We considered k with values of 0, 0.5, or 1.0. We considered a value of 0 to compare the performance of the proposed algorithms with those of the existing algorithms with zero setup times because there have been no previous reports of the two-stage

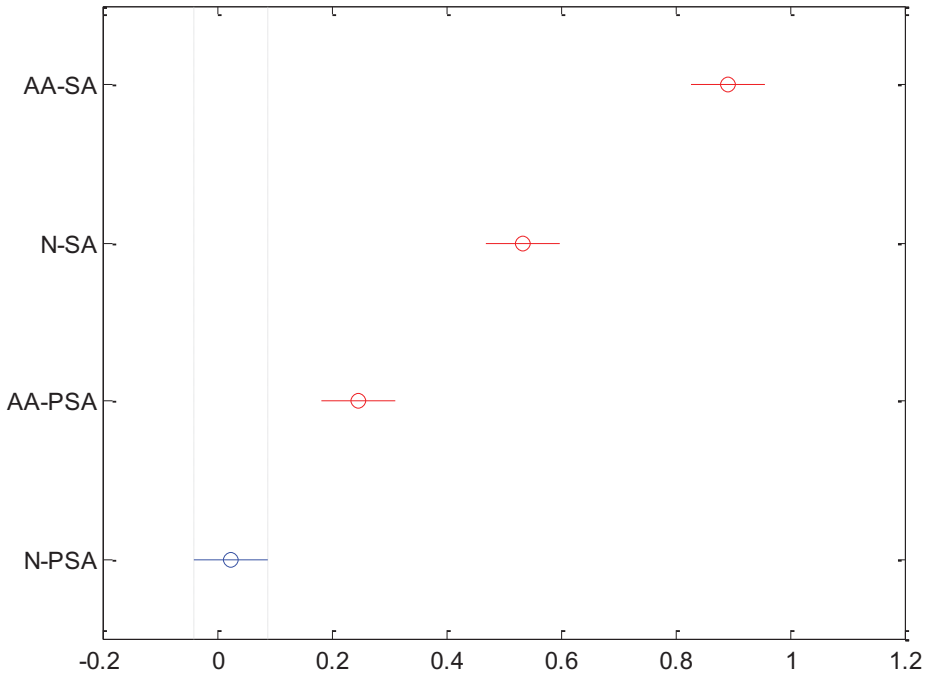

Fig. 6. Confidence intervals for $n = 70$, $m = 5$, $R = 1.0$, $T = 0.4$, $k = 0.5$.

assembly flowshop scheduling problem with setup times and the objective of minimizing the total tardiness. However, some algorithms exist for the problem with zero setup times.

We considered a different number of jobs (n), i.e., $n = 30$, 40, 50, 60, 70, and 80. We also considered four different values for the number of machines at the first stage, i.e., $m = 2$, 5, 10, and 12.

We generated the due dates for jobs from a uniform distribution over the interval of $[LC_{max}(1-T-R/2),\ LC_{max}(1-T+R/2)]$, where LC_{max} denotes a lower bound for the makespan. The parameters T and R denote the tardiness factor and relative range for the due dates, respectively. The generation of due dates using this method is common in previous scheduling studies for use in computational experiments (e.g., [18]). We used the following LC_{max} value.

$$LC_{\max} = \max\left(\max_{k=1,\ldots,m}\left\{\sum_{r=1}^{n}(p_{[r,k]} + s_{[r,k]})\right\} + \min_{i}\{p_i + s_i\},\ \sum_{r=1}^{n}(p_{[r]} + s_{[r]})\right)$$

In previous scheduling studies, e.g., Vallada and Ruiz [32], a value between 0 and 1 was commonly assumed for the tardiness parameter (T) and range parameter (R). Hence, we considered values of 0.2, 0.4, and 0.6 for T, and values of 0.2, 0.6, and 1 for R.

In total, we analyzed 648 combinations of n, m, k, R, and T. For each selected combination, 40 replicates were generated.

The performance of the algorithms was assessed using the percentage relative error (Error) as a performance measure. Error is defined as 100*(TT of the algorithm – TT of the best algorithm)/TT of the best algorithm, where TT denotes the total tardiness.

We compared the six algorithms, i.e., AA-SA, N-SA, LZ-HGA, MZY-SA, AA-PSA, and N-PSA, in two different groups: group one comprised AA-SA, N-SA, LZ-HGA, and MZY-SA, and group two comprised AA-PSA and N-PSA. We then compared the two best performing algorithms in the first group with those in the second group. Hence, we conducted a complete comparison of the six algorithms in both groups. We analyzed the algorithms in two groups for two reasons. The first and most important reason is that there was a huge difference in the performance of the algorithms in the second group and some of the algorithms in the first group. Thus, as shown in Fig. 1, it would have been difficult to distinguish the performance of the best algorithms. It should be noted that the y-axis in Fig. 1, as well as the other figures, represents the error as a percentage. The second reason is that the CPU times of the algorithms in the first group were equal and those of the second group were equal, but the CPU times of the algorithms in the second group were naturally larger than those of the algorithms in the first group (at least AA-SA, N-SA) because an insertion algorithm is combined with SA for the algorithms in the second group.

The four algorithms, i.e., AA-SA, N-SA, LZ-HGA, and MZY-SA, were compared so their CPU times were equal. The results of the computational experiments are presented in Tables 1–3, where the results for $n = 30$ and 40, and $m = 2$ are omitted due to space limitations. However, all of the results are summarized in the figures. Fig. 2 shows the error versus the number of jobs, which clearly demonstrates that both AA-SA and N-SA performed much better than LZ-HGA and MZY-SA. The same

Table 1
Percentage errors for the algorithms in group one for $k = 0$.

				m = 5			10			12	
				T			T			T	
n	R	Algorithm	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6
50	0.2	*AA-SA*	0.44	0.20	0.32	0.32	0.37	0.29	0.30	0.51	0.34
		N-SA	0.05	0.13	0.12	0.11	0.12	0.15	0.15	0.06	0.03
		LZ-HGA	5.91	5.85	4.96	5.12	4.74	5.43	4.74	4.41	4.29
		MZY-SA	6.22	11.31	9.17	4.48	5.48	7.69	3.64	7.71	7.61
50	0.6	*AA-SA*	0.25	0.42	0.38	0.47	0.41	0.18	0.46	0.27	0.40
		N-SA	0.07	0.05	0.12	0.04	0.10	0.18	0.02	0.10	0.07
		LZ-HGA	5.67	5.72	5.79	5.52	4.64	4.12	4.52	4.86	4.23
		MZY-SA	8.87	6.21	9.34	4.65	7.16	4.47	3.28	4.86	7.76
50	1.0	*AA-SA*	0.33	0.27	0.19	0.76	0.27	0.26	0.36	0.26	0.49
		N-SA	0.21	0.11	0.05	0.08	0.06	0.05	0.03	0.06	0.08
		LZ-HGA	5.09	5.97	5.56	4.58	4.92	4.97	4.46	3.93	4.15
		MZY-SA	8.16	8.19	9.56	4.23	5.43	6.51	4.83	3.09	7.24
60	0.2	*AA-SA*	0.61	0.14	0.29	0.15	0.55	0.40	0.47	0.57	0.61
		N-SA	0.01	0.20	0.09	0.28	0.19	0.10	0.09	0.01	0.02
		LZ-HGA	5.21	5.67	4.80	3.89	4.57	4.20	4.90	4.14	4.00
		MZY-SA	3.99	4.72	4.76	2.72	6.07	7.45	7.05	4.28	5.76
60	0.6	*AA-SA*	0.55	0.56	0.45	0.28	0.40	0.38	0.50	0.22	0.46
		N-SA	0.02	0.04	0.08	0.18	0.15	0.09	0.03	0.05	0.05
		LZ-HGA	4.46	5.84	5.32	4.57	5.02	4.62	3.91	4.28	4.22
		MZY-SA	3.68	11.14	6.31	6.53	6.74	5.25	5.87	2.88	7.39
60	1.0	*AA-SA*	0.15	0.19	0.36	0.36	0.39	0.37	0.34	0.69	0.27
		N-SA	0.27	0.16	0.08	0.19	0.18	0.18	0.04	0.09	0.03
		LZ-HGA	5.57	5.16	5.22	4.92	4.53	4.00	4.17	4.08	3.90
		MZY-SA	3.77	5.29	6.84	3.61	5.42	4.47	5.79	8.01	5.37
70	0.2	*AA-SA*	0.23	0.50	0.41	0.39	0.49	0.12	0.35	0.50	0.26
		N-SA	0.07	0.07	0.01	0.06	0.00	0.13	0.07	0.00	0.14
		LZ-HGA	4.57	4.57	4.49	4.05	4.63	3.72	4.16	3.92	3.71
		MZY-SA	5.39	5.42	3.61	6.97	3.46	2.93	8.18	2.78	2.58
70	0.6	*AA-SA*	0.25	0.44	0.29	0.38	0.46	0.19	0.47	0.36	0.28
		N-SA	0.09	0.08	0.26	0.04	0.05	0.14	0.06	0.09	0.13
		LZ-HGA	5.47	5.54	4.40	4.60	4.22	4.42	3.82	4.70	4.15
		MZY-SA	7.45	6.26	8.66	3.36	2.78	8.38	6.14	4.60	6.06
70	1.0	*AA-SA*	0.37	0.25	0.33	0.42	0.34	0.52	0.26	0.40	0.52
		N-SA	0.06	0.12	0.09	0.08	0.12	0.09	0.09	0.06	0.08
		LZ-HGA	5.91	5.28	5.45	4.37	3.83	4.20	3.84	4.32	3.86
		MZY-SA	4.43	8.58	7.34	4.50	4.61	3.51	5.57	5.21	3.52
80	0.2	*AA-SA*	0.56	0.47	0.36	0.50	0.41	0.53	0.26	0.32	0.59
		N-SA	0.05	0.08	0.15	0.08	0.05	0.03	0.20	0.14	0.02
		LZ-HGA	4.76	5.36	4.59	3.52	3.93	3.90	4.27	4.11	3.40
		MZY-SA	7.44	5.92	6.88	4.53	6.11	7.37	7.69	7.07	3.35
80	0.6	*AA-SA*	0.37	0.33	0.38	0.38	0.42	0.65	0.41	0.36	0.30
		N-SA	0.05	0.14	0.02	0.00	0.07	0.02	0.01	0.09	0.03
		LZ-HGA	4.57	4.25	5.37	4.04	3.89	3.75	3.83	4.02	3.65
		MZY-SA	3.15	8.02	8.39	7.66	6.92	4.15	5.86	7.59	4.81
80	1.0	*AA-SA*	0.41	0.37	0.41	0.35	0.36	0.50	0.49	0.64	0.51
		N-SA	0.11	0.08	0.05	0.04	0.18	0.06	0.02	0.05	0.00
		LZ-HGA	5.51	4.49	5.09	3.72	4.00	4.07	3.72	3.18	3.59
		MZY-SA	7.92	7.49	6.87	3.20	4.37	3.04	4.83	3.73	5.45

conclusion can be obtained by analyzing the error with respect to the k values in Fig. 3. Figs. 2 and 3 show that the LZ-HGA and MZY-SA algorithms performed considerably worse than the others with smaller n and k values. The same conclusion can be made with respect to the m values. As shown in Figs. 2 and 3, AA-SA and N-SA appeared to obtain similar performance, but the newly proposed N-SA algorithm actually performed significantly better than AA-SA according to Fig. 4. Fig. 4 was obtained by excluding the LZ-HGA and MZY-SA algorithms from Fig. 2 to facilitate a clearer comparison. Fig. 4 clearly indicates that the error with N-SA was much smaller than that with AA-SA in terms of n. Similar comparisons were obtained with respect to m, k, R, and T; however, the corresponding figures are not provided due to space limitations. The overall errors using the AA-SA, N-SA, LZ-HGA, and MZY-SA algorithms were 0.31, 0.08, 4.29, and 5.65, respectively. The overall error with N-SA was about 74% smaller than that with AA-SA, thereby demonstrating the higher performance of the

Table 2
Percentage errors for the algorithms in group one for $k = 0.5$.

				m = 5			10			12	
				T			T			T	
n	R	Algorithm	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6
50	0.2	*AA-SA*	0.27	0.31	0.43	0.30	0.29	0.32	0.21	0.26	0.14
		N-SA	0.12	0.01	0.04	0.16	0.00	0.14	0.07	0.20	0.08
		LZ-HGA	3.96	4.00	4.01	3.67	3.35	3.32	3.78	3.41	3.56
		MZY-SA	6.40	2.84	2.63	4.39	5.92	6.24	7.38	6.07	4.87
50	0.6	*AA-SA*	0.38	0.30	0.21	0.30	0.21	0.19	0.35	0.32	0.41
		N-SA	0.07	0.00	0.06	0.18	0.06	0.14	0.03	0.05	0.02
		LZ-HGA	4.08	3.94	4.28	3.44	3.40	3.48	3.49	3.58	3.70
		MZY-SA	7.95	6.50	4.00	5.95	2.46	6.30	4.10	3.67	6.01
50	1.0	*AA-SA*	0.41	0.16	0.46	0.29	0.24	0.32	0.28	0.21	0.24
		N-SA	0.01	0.09	0.01	0.02	0.05	0.10	0.03	0.05	0.07
		LZ-HGA	4.29	4.45	3.92	3.68	3.32	3.28	3.24	3.19	3.15
		MZY-SA	6.93	6.25	5.26	5.45	4.90	4.76	3.55	5.24	2.58
60	0.2	*AA-SA*	0.35	0.47	0.29	0.23	0.42	0.44	0.22	0.38	0.54
		N-SA	0.04	0.12	0.11	0.07	0.05	0.05	0.10	0.09	0.06
		LZ-HGA	3.58	3.89	4.20	3.80	3.23	3.30	3.15	2.88	3.35
		MZY-SA	3.08	4.18	4.32	4.88	5.32	4.33	4.10	3.47	6.19
60	0.6	*AA-SA*	0.34	0.27	0.23	0.26	0.46	0.17	0.19	0.32	0.22
		N-SA	0.03	0.12	0.05	0.12	0.00	0.08	0.22	0.10	0.07
		LZ-HGA	3.97	4.11	3.41	3.52	2.84	3.37	2.77	3.32	3.20
		MZY-SA	6.18	7.77	3.90	6.74	3.19	4.49	3.66	4.85	2.99
60	1.0	*AA-SA*	0.34	0.25	0.42	0.28	0.39	0.38	0.30	0.18	0.21
		N-SA	0.08	0.04	0.05	0.06	0.07	0.06	0.07	0.12	0.10
		LZ-HGA	3.62	3.53	3.69	3.52	3.52	3.28	3.26	3.24	3.40
		MZY-SA	6.15	6.81	5.18	3.20	5.73	2.46	5.21	4.83	2.83
70	0.2	*AA-SA*	0.55	0.31	0.44	0.55	0.51	0.35	0.47	0.43	0.46
		N-SA	0.04	0.06	0.12	0.12	0.01	0.13	0.00	0.08	0.04
		LZ-HGA	3.74	3.70	3.28	3.08	3.42	3.16	2.53	2.80	3.23
		MZY-SA	4.16	3.61	2.17	4.00	5.56	5.33	3.01	3.08	4.86
70	0.6	*AA-SA*	0.19	0.32	0.32	0.29	0.23	0.14	0.12	0.23	0.38
		N-SA	0.03	0.01	0.03	0.08	0.15	0.10	0.09	0.15	0.05
		LZ-HGA	3.56	3.27	3.58	3.04	3.33	2.71	2.62	3.00	2.72
		MZY-SA	3.98	3.81	6.68	3.25	6.49	4.07	4.61	4.64	1.84
70	1.0	*AA-SA*	0.46	0.28	0.21	0.33	0.41	0.37	0.40	0.23	0.35
		N-SA	0.04	0.02	0.06	0.11	0.02	0.09	0.03	0.02	0.07
		LZ-HGA	3.47	4.01	3.96	3.03	3.20	3.42	3.42	3.03	3.07
		MZY-SA	3.81	3.42	7.18	3.19	5.33	3.23	5.11	4.38	3.66
80	0.2	*AA-SA*	0.18	0.33	0.26	0.45	0.41	0.29	0.21	0.40	0.21
		N-SA	0.13	0.03	0.07	0.04	0.05	0.03	0.14	0.01	0.07
		LZ-HGA	3.50	3.71	3.03	2.51	2.66	2.58	2.71	2.84	2.96
		MZY-SA	6.45	6.15	3.08	3.63	3.63	1.95	4.31	4.47	5.65
80	0.6	*AA-SA*	0.22	0.26	0.39	0.26	0.27	0.18	0.21	0.55	0.45
		N-SA	0.04	0.17	0.02	0.10	0.08	0.06	0.11	0.05	0.02
		LZ-HGA	3.56	3.55	3.43	2.90	2.59	2.75	2.44	2.54	2.57
		MZY-SA	4.68	3.83	2.84	4.13	5.05	3.37	1.99	3.93	4.87
80	1.0	*AA-SA*	0.46	0.41	0.40	0.23	0.42	0.35	0.28	0.22	0.23
		N-SA	0.00	0.00	0.05	0.05	0.07	0.02	0.14	0.11	0.04
		LZ-HGA	3.48	3.47	3.18	2.89	3.05	2.61	3.00	2.80	2.76
		MZY-SA	6.18	4.33	6.13	3.22	2.83	2.48	3.66	2.43	2.51

proposed N-SA algorithm. It can be seen that the N-SA algorithm performed substantially better than AA-SA even when $k = 0$, which is the case with zero setup times.

The best performing algorithms in the first group were AA-SA and N-SA. Hence, they were compared with the AA-PSA and N-PSA algorithms from the second group. The computational experiments are summarized in Fig. 5 with respect to n, which shows clearly that the AA-PSA and N-PSA algorithms performed much better than AA-SA and N-SA. Hence, the latter two algorithms were excluded from further analyses to better distinguish the differences between the two algorithms in the second group.

The AA-PSA and N-PSA algorithms were compared under the same CPU, and the results are provided in Tables 4–6, where the results obtained for $n = 30$ and 40, and $m = 2$ are excluded due to space limitations. The overall error with N-PSA was about 52% smaller than that using AA-PSA.

Table 3
Percentage errors for the algorithms in group one for $k = 1.0$.

				m = 5			10			12	
				T			T			T	
n	R	Algorithm	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6
50	0.2	*AA-SA*	0.30	0.29	0.37	0.27	0.27	0.41	0.43	0.31	0.24
		N-SA	0.07	0.03	0.03	0.02	0.06	0.01	0.04	0.06	0.07
		LZ-HGA	3.54	3.64	4.41	3.16	3.75	3.34	3.24	2.99	3.40
		MZY-SA	3.88	3.01	3.90	6.09	5.76	3.61	4.27	4.25	4.81
50	0.6	*AA-SA*	0.26	0.26	0.39	0.17	0.18	0.19	0.31	0.27	0.17
		N-SA	0.12	0.10	0.04	0.03	0.01	0.04	0.05	0.09	0.08
		LZ-HGA	3.74	3.60	3.33	3.44	3.22	3.38	3.12	3.23	3.44
		MZY-SA	7.12	4.90	5.34	4.05	5.18	5.05	5.61	2.65	5.78
50	1.0	*AA-SA*	0.40	0.44	0.43	0.40	0.13	0.32	0.38	0.45	0.48
		N-SA	0.00	0.00	0.04	0.04	0.08	0.05	0.03	0.08	0.08
		LZ-HGA	3.95	3.49	3.88	3.36	3.53	3.36	3.33	3.18	3.19
		MZY-SA	7.09	5.72	7.42	3.87	3.19	2.76	6.33	2.96	3.17
60	0.2	*AA-SA*	0.28	0.07	0.19	0.31	0.23	0.23	0.25	0.38	0.21
		N-SA	0.02	0.08	0.10	0.08	0.05	0.08	0.04	0.01	0.04
		LZ-HGA	3.92	3.97	3.69	3.12	3.06	3.23	2.66	3.02	2.99
		MZY-SA	4.09	3.48	4.25	3.27	3.25	3.21	2.72	3.02	3.04
60	0.6	*AA-SA*	0.41	0.33	0.46	0.23	0.37	0.38	0.27	0.22	0.19
		N-SA	0.10	0.08	0.00	0.13	0.01	0.03	0.08	0.22	0.07
		LZ-HGA	4.04	3.81	4.03	3.56	2.99	2.81	2.79	2.86	2.84
		MZY-SA	4.46	4.24	5.66	5.28	2.08	5.01	3.20	2.35	3.90
60	1.0	*AA-SA*	0.37	0.30	0.30	0.19	0.25	0.32	0.32	0.25	0.29
		N-SA	0.01	0.04	0.06	0.17	0.12	0.08	0.03	0.02	0.02
		LZ-HGA	3.93	3.50	4.11	3.95	3.21	3.04	2.62	2.86	2.87
		MZY-SA	4.01	6.16	5.35	2.65	5.74	5.64	3.21	5.35	3.26
70	0.2	*AA-SA*	0.43	0.16	0.26	0.27	0.46	0.41	0.33	0.35	0.37
		N-SA	0.04	0.03	0.08	0.08	0.02	0.02	0.07	0.07	0.04
		LZ-HGA	3.61	3.47	3.52	3.06	2.93	3.06	2.70	2.88	2.67
		MZY-SA	5.08	3.47	3.35	5.02	3.76	4.16	4.58	3.24	4.64
70	0.6	*AA-SA*	0.30	0.52	0.47	0.32	0.42	0.28	0.36	0.57	0.35
		N-SA	0.14	0.03	0.01	0.02	0.05	0.00	0.06	0.00	0.13
		LZ-HGA	3.66	3.41	3.50	2.91	2.89	2.85	2.35	2.58	2.82
		MZY-SA	5.69	4.36	5.47	5.19	2.16	3.09	1.92	3.42	3.69
70	1.0	*AA-SA*	0.26	0.26	0.37	0.37	0.37	0.25	0.26	0.27	0.43
		N-SA	0.06	0.09	0.12	0.02	0.06	0.06	0.08	0.12	0.00
		LZ-HGA	3.01	3.24	3.31	2.84	2.80	2.78	2.98	3.20	3.02
		MZY-SA	5.23	5.21	4.89	5.28	2.76	2.11	4.43	6.00	3.97
80	0.2	*AA-SA*	0.24	0.40	0.13	0.40	0.36	0.60	0.37	0.43	0.24
		N-SA	0.03	0.01	0.15	0.04	0.08	0.02	0.05	0.07	0.03
		LZ-HGA	3.18	3.09	2.78	2.85	2.83	2.77	2.86	2.68	2.47
		MZY-SA	6.07	2.12	4.17	3.92	4.65	3.96	2.26	4.79	3.60
80	0.6	*AA-SA*	0.46	0.39	0.14	0.36	0.50	0.44	0.43	0.31	0.31
		N-SA	0.02	0.03	0.11	0.01	0.06	0.03	0.05	0.06	0.08
		LZ-HGA	3.34	3.02	3.19	2.62	2.77	2.61	2.45	2.54	2.68
		MZY-SA	3.82	3.79	6.24	1.99	2.08	5.08	2.03	2.42	5.08
80	1.0	*AA-SA*	0.25	0.20	0.23	0.58	0.38	0.32	0.24	0.33	0.30
		N-SA	0.12	0.12	0.08	0.02	0.11	0.02	0.10	0.03	0.07
		LZ-HGA	3.15	3.17	3.21	2.92	2.98	2.51	2.79	2.76	2.46
		MZY-SA	4.67	5.97	2.61	3.60	2.79	4.35	3.54	4.49	4.35

In order to confirm the results of our analysis using a statistical test, the algorithms were compared with Tukey's honestly significant difference (HSD) test at the 2.5% level of significance. Fig. 6 shows the results for a sample case of HSD, which is representative of the vast majority of cases. In general, the results confirmed our conclusions.

The AA-SA and N-SA algorithms had the same CPU time, but the overall error with N-SA was about 74% smaller than that using AA-SA, thereby demonstrating the high performance of the proposed N-SA algorithm. Under the same CPU time, the performance of the proposed N-PSA algorithm was 52% better than that of AA-PSA. The CPU time required by the algorithms in the first group was less than 40 s for the largest problem size ($n = 80$ and $m = 12$). The CPU time required by the algorithms in the second group was less than 8 s even for the largest size problem. Assuming that the computational time is not an issue, the best algorithm is N-PSA because the overall error with N-PSA was about 77% smaller than that

Table 4
Percentage errors for the best four algorithms with $k = 0$.

				m = 5			10			12	
				T			T			T	
n	R	Algorithm	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6
50	0.2	*AA-SA*	0.88	0.69	0.78	0.82	0.88	0.75	0.75	1.00	0.78
		N-SA	0.43	0.54	0.52	0.54	0.55	0.53	0.53	0.48	0.41
		AA-PSA	0.29	0.13	0.23	0.33	0.30	0.19	0.23	0.48	0.18
		N-PSA	0.06	0.14	0.20	0.16	0.14	0.18	0.14	0.04	0.05
50	0.6	*AA-SA*	0.66	0.85	0.85	0.92	0.91	0.68	0.95	0.79	0.79
		N-SA	0.42	0.42	0.52	0.42	0.52	0.59	0.45	0.54	0.40
		AA-PSA	0.14	0.28	0.37	0.36	0.32	0.10	0.40	0.22	0.30
		N-PSA	0.13	0.02	0.15	0.07	0.15	0.22	0.01	0.07	0.04
50	1.0	*AA-SA*	0.82	0.72	0.70	1.27	0.82	0.80	0.79	0.76	0.84
		N-SA	0.62	0.49	0.49	0.52	0.53	0.51	0.41	0.49	0.38
		AA-PSA	0.21	0.18	0.07	0.52	0.22	0.18	0.24	0.15	0.26
		N-PSA	0.24	0.13	0.18	0.09	0.07	0.13	0.07	0.09	0.02
60	0.2	*AA-SA*	1.08	0.72	0.78	0.75	1.09	0.91	1.05	1.13	1.09
		N-SA	0.42	0.68	0.51	0.78	0.65	0.53	0.58	0.50	0.44
		AA-PSA	0.38	0.15	0.19	0.11	0.41	0.29	0.42	0.37	0.45
		N-PSA	0.03	0.25	0.14	0.29	0.24	0.06	0.06	0.05	0.02
60	0.6	*AA-SA*	1.10	1.06	1.06	0.78	0.92	0.89	1.00	0.81	1.02
		N-SA	0.50	0.47	0.61	0.60	0.59	0.52	0.47	0.56	0.53
		AA-PSA	0.46	0.35	0.38	0.14	0.26	0.26	0.37	0.18	0.28
		N-PSA	0.03	0.06	0.15	0.22	0.09	0.09	0.02	0.13	0.12
60	1.0	*AA-SA*	0.72	0.72	1.00	0.89	0.96	0.94	1.00	1.19	0.87
		N-SA	0.74	0.61	0.63	0.64	0.66	0.67	0.61	0.52	0.54
		AA-PSA	0.10	0.11	0.38	0.29	0.32	0.32	0.26	0.46	0.17
		N-PSA	0.27	0.16	0.08	0.17	0.19	0.23	0.04	0.12	0.06
70	0.2	*AA-SA*	0.86	1.10	1.03	0.97	1.02	0.85	0.88	1.08	0.96
		N-SA	0.61	0.59	0.55	0.56	0.47	0.74	0.52	0.51	0.74
		AA-PSA	0.25	0.37	0.30	0.35	0.30	0.12	0.23	0.33	0.22
		N-PSA	0.11	0.08	0.01	0.04	0.03	0.20	0.06	0.00	0.17
70	0.6	*AA-SA*	0.98	1.09	0.84	0.89	1.07	0.80	1.04	0.94	0.87
		N-SA	0.72	0.64	0.72	0.48	0.57	0.66	0.55	0.58	0.63
		AA-PSA	0.24	0.32	0.18	0.23	0.27	0.11	0.41	0.26	0.24
		N-PSA	0.20	0.19	0.26	0.03	0.13	0.17	0.09	0.07	0.07
70	1.0	*AA-SA*	0.99	0.87	0.90	1.12	0.86	1.11	0.92	1.08	1.12
		N-SA	0.60	0.65	0.57	0.68	0.56	0.60	0.66	0.65	0.60
		AA-PSA	0.21	0.22	0.20	0.22	0.20	0.38	0.20	0.30	0.38
		N-PSA	0.05	0.15	0.05	0.15	0.14	0.12	0.08	0.10	0.10
80	0.2	*AA-SA*	1.13	1.02	1.06	1.10	1.02	1.16	0.88	0.96	1.16
		N-SA	0.55	0.55	0.74	0.60	0.58	0.57	0.72	0.68	0.52
		AA-PSA	0.42	0.31	0.27	0.39	0.28	0.39	0.23	0.26	0.33
		N-PSA	0.03	0.11	0.20	0.10	0.11	0.05	0.10	0.14	0.06
80	0.6	*AA-SA*	0.99	1.03	1.04	0.99	1.01	1.25	1.02	0.97	0.91
		N-SA	0.59	0.74	0.59	0.53	0.58	0.54	0.54	0.61	0.56
		AA-PSA	0.22	0.30	0.22	0.22	0.29	0.43	0.22	0.27	0.25
		N-PSA	0.08	0.25	0.09	0.01	0.08	0.06	0.03	0.09	0.07
80	1.0	*AA-SA*	1.06	0.97	1.00	0.99	1.03	1.14	1.13	1.22	1.14
		N-SA	0.66	0.60	0.56	0.59	0.75	0.61	0.58	0.55	0.56
		AA-PSA	0.29	0.24	0.24	0.21	0.30	0.34	0.39	0.43	0.44
		N-PSA	0.15	0.16	0.13	0.07	0.16	0.10	0.01	0.10	0.00

using N-SA. Hence, the proposed N-PSA algorithm, which has a complexity of $O(L^*n^2)$, is recommended for separate setup times, as well as for the case with zero setup times.

Given that N-PSA was the best algorithm, we also investigated its performance with respect to the optimal solution for small size problems. An optimal solution was obtained by using a mathematical formulation based on the model described by Dalfard et al. [13], where the parameters β, γ, δ were set at zero, w_j was set at one, and T_j^2 was replaced with T_j in the objective function. Moreover, the transportation times were replaced with zero. Thus, we compared the performance of the proposed algorithm N-PSA with the optimal solution and the results are summarized in Table 7 as percentages of the absolute error. The absolute error with the proposed algorithm was around 0.05%, which is extremely small, as shown in the table.

Table 5
Percentage errors for the best four algorithms with $k = 0.5$.

				m = 5			m = 10			m = 12	
				T			T			T	
n	R	Algorithm	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6
50	0.2	*AA-SA*	0.71	0.70	0.87	0.80	0.80	0.71	0.65	0.74	0.62
		N-SA	0.49	0.36	0.42	0.58	0.45	0.46	0.45	0.60	0.49
		AA-PSA	0.16	0.20	0.30	0.25	0.20	0.24	0.15	0.15	0.06
		N-PSA	0.13	0.05	0.09	0.18	0.04	0.13	0.03	0.15	0.07
50	0.6	*AA-SA*	0.87	0.81	0.71	0.73	0.69	0.72	0.82	0.74	0.89
		N-SA	0.50	0.45	0.49	0.54	0.47	0.59	0.44	0.41	0.43
		AA-PSA	0.20	0.22	0.11	0.23	0.12	0.17	0.26	0.16	0.30
		N-PSA	0.12	0.02	0.10	0.21	0.10	0.20	0.03	0.06	0.04
50	1.0	*AA-SA*	0.90	0.63	0.90	0.71	0.70	0.79	0.70	0.70	0.71
		N-SA	0.44	0.50	0.40	0.39	0.45	0.51	0.39	0.47	0.47
		AA-PSA	0.30	0.13	0.34	0.18	0.17	0.30	0.16	0.13	0.13
		N-PSA	0.04	0.13	0.04	0.08	0.10	0.11	0.08	0.04	0.08
60	0.2	*AA-SA*	0.91	0.92	0.82	0.67	0.93	0.89	0.79	0.88	1.01
		N-SA	0.52	0.51	0.56	0.46	0.49	0.44	0.58	0.52	0.47
		AA-PSA	0.23	0.28	0.20	0.06	0.28	0.27	0.21	0.24	0.39
		N-PSA	0.04	0.09	0.12	0.08	0.10	0.06	0.06	0.09	0.03
60	0.6	*AA-SA*	0.84	0.80	0.70	0.79	0.97	0.67	0.74	0.82	0.75
		N-SA	0.46	0.57	0.45	0.57	0.45	0.51	0.67	0.53	0.53
		AA-PSA	0.28	0.19	0.13	0.17	0.34	0.14	0.10	0.26	0.18
		N-PSA	0.05	0.17	0.04	0.17	0.01	0.10	0.18	0.09	0.07
60	1.0	*AA-SA*	0.81	0.81	0.93	0.77	0.88	0.86	0.84	0.70	0.76
		N-SA	0.49	0.53	0.50	0.48	0.49	0.48	0.52	0.56	0.58
		AA-PSA	0.14	0.17	0.36	0.15	0.28	0.24	0.22	0.13	0.19
		N-PSA	0.09	0.07	0.05	0.06	0.11	0.05	0.05	0.08	0.08
70	0.2	*AA-SA*	0.99	0.89	1.03	1.04	1.09	0.90	0.95	1.02	0.97
		N-SA	0.41	0.56	0.62	0.53	0.51	0.60	0.43	0.58	0.48
		AA-PSA	0.34	0.26	0.32	0.36	0.43	0.23	0.28	0.41	0.32
		N-PSA	0.04	0.09	0.17	0.12	0.02	0.16	0.00	0.06	0.03
70	0.6	*AA-SA*	0.71	0.94	0.88	0.83	0.78	0.67	0.73	0.76	0.96
		N-SA	0.48	0.55	0.52	0.55	0.62	0.54	0.61	0.60	0.54
		AA-PSA	0.13	0.29	0.23	0.16	0.13	0.08	0.14	0.11	0.32
		N-PSA	0.04	0.00	0.05	0.12	0.19	0.11	0.08	0.15	0.10
70	1.0	*AA-SA*	1.02	0.83	0.72	0.93	1.00	0.94	0.98	0.81	0.87
		N-SA	0.52	0.50	0.50	0.63	0.53	0.58	0.53	0.52	0.52
		AA-PSA	0.35	0.15	0.14	0.23	0.36	0.21	0.36	0.13	0.21
		N-PSA	0.09	0.03	0.07	0.14	0.08	0.09	0.02	0.09	0.06
80	0.2	*AA-SA*	0.81	0.87	0.85	0.97	0.93	0.85	0.78	0.90	0.76
		N-SA	0.66	0.50	0.57	0.49	0.50	0.52	0.63	0.45	0.54
		AA-PSA	0.09	0.24	0.16	0.27	0.30	0.15	0.14	0.33	0.17
		N-PSA	0.22	0.02	0.11	0.04	0.01	0.05	0.10	0.00	0.09
80	0.6	*AA-SA*	0.85	0.77	0.96	0.89	0.82	0.82	0.84	1.08	1.02
		N-SA	0.59	0.60	0.52	0.64	0.55	0.61	0.64	0.51	0.52
		AA-PSA	0.10	0.12	0.29	0.23	0.23	0.19	0.16	0.47	0.41
		N-PSA	0.09	0.21	0.01	0.15	0.11	0.09	0.11	0.04	0.03
80	1.0	*AA-SA*	0.95	1.05	0.96	0.84	0.97	0.91	0.81	0.80	0.83
		N-SA	0.43	0.56	0.54	0.58	0.54	0.51	0.60	0.61	0.56
		AA-PSA	0.25	0.28	0.32	0.22	0.31	0.23	0.21	0.15	0.17
		N-PSA	0.00	0.04	0.05	0.09	0.05	0.05	0.10	0.11	0.03

6. Concluding remarks

In this study, we considered the two-stage assembly flowshop scheduling problem using the total tardiness as a performance measure. We treated the setup times separately from the processing times, thereby helping to increase the machine utilization and decreasing the total tardiness. The AA-SA and AA-PSA algorithms are adapted versions of SA and PSA, which are the best known algorithms according to previous studies for the problem with zero setup times. The LZ-HGA and MZY-SA algorithms are adapted versions of two recently described algorithms, which were developed for similar problems, where the setup times are considered explicitly. In addition, N-SA and N-PSA are two newly algorithms that we proposed for solving the problem. Furthermore, we developed a dominance relation, which we incorporated in the proposed algorithms. We conducted extensive computational experiments to evaluate the algorithms. In general, we found that the initial sequences,

Table 6
Percentage errors for the best four algorithms with $k = 1.0$.

				m = 5			10			12	
				T			T			T	
n	R	Algorithm	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6
50	0.2	*AA-SA*	0.79	0.74	0.77	0.72	0.79	0.90	0.96	0.79	0.78
		N-SA	0.49	0.41	0.37	0.41	0.51	0.43	0.50	0.48	0.53
		AA-PSA	0.21	0.17	0.25	0.17	0.27	0.31	0.36	0.33	0.22
		N-PSA	0.05	0.07	0.05	0.03	0.09	0.01	0.04	0.07	0.08
50	0.6	*AA-SA*	0.67	0.69	0.80	0.67	0.59	0.67	0.81	0.69	0.57
		N-SA	0.46	0.46	0.40	0.46	0.36	0.46	0.49	0.45	0.43
		AA-PSA	0.09	0.16	0.22	0.14	0.10	0.16	0.25	0.16	0.15
		N-PSA	0.10	0.04	0.03	0.05	0.03	0.12	0.08	0.07	0.06
50	1.0	*AA-SA*	0.80	0.87	0.89	0.86	0.61	0.79	0.79	0.96	0.87
		N-SA	0.35	0.38	0.45	0.44	0.49	0.45	0.39	0.52	0.42
		AA-PSA	0.26	0.17	0.29	0.22	0.02	0.21	0.21	0.36	0.28
		N-PSA	0.04	0.04	0.04	0.09	0.14	0.12	0.03	0.09	0.08
60	0.2	*AA-SA*	0.79	0.62	0.73	0.80	0.75	0.79	0.77	0.82	0.72
		N-SA	0.46	0.55	0.56	0.50	0.49	0.56	0.49	0.40	0.49
		AA-PSA	0.24	0.07	0.20	0.16	0.19	0.18	0.16	0.24	0.14
		N-PSA	0.02	0.07	0.06	0.08	0.05	0.17	0.04	0.01	0.06
60	0.6	*AA-SA*	0.91	0.85	0.97	0.77	0.86	0.86	0.81	0.72	0.68
		N-SA	0.53	0.52	0.45	0.59	0.44	0.45	0.54	0.63	0.49
		AA-PSA	0.28	0.21	0.35	0.20	0.28	0.30	0.22	0.17	0.15
		N-PSA	0.12	0.08	0.02	0.14	0.02	0.04	0.09	0.13	0.07
60	1.0	*AA-SA*	0.92	0.77	0.81	0.73	0.70	0.78	0.80	0.76	0.85
		N-SA	0.48	0.45	0.50	0.62	0.50	0.47	0.44	0.46	0.51
		AA-PSA	0.25	0.21	0.20	0.13	0.14	0.22	0.22	0.20	0.16
		N-PSA	0.05	0.00	0.11	0.15	0.11	0.07	0.02	0.05	0.04
70	0.2	*AA-SA*	1.00	0.68	0.85	0.79	1.00	1.00	0.89	0.86	0.87
		N-SA	0.53	0.48	0.59	0.53	0.49	0.53	0.56	0.51	0.48
		AA-PSA	0.30	0.08	0.18	0.16	0.36	0.28	0.32	0.24	0.25
		N-PSA	0.06	0.07	0.14	0.10	0.03	0.04	0.09	0.05	0.01
70	0.6	*AA-SA*	0.88	1.04	0.94	0.80	0.92	0.86	0.94	1.11	0.88
		N-SA	0.63	0.49	0.42	0.44	0.48	0.50	0.55	0.47	0.58
		AA-PSA	0.20	0.38	0.30	0.17	0.26	0.19	0.28	0.45	0.24
		N-PSA	0.15	0.03	0.03	0.03	0.06	0.00	0.08	0.01	0.09
70	1.0	*AA-SA*	0.83	0.80	0.91	0.89	0.91	0.89	0.81	0.75	0.93
		N-SA	0.56	0.55	0.58	0.46	0.52	0.61	0.55	0.53	0.44
		AA-PSA	0.21	0.21	0.29	0.26	0.34	0.22	0.19	0.17	0.35
		N-PSA	0.07	0.11	0.11	0.01	0.04	0.16	0.10	0.05	0.00
80	0.2	*AA-SA*	0.80	1.00	0.75	0.96	0.88	1.21	0.90	0.93	0.81
		N-SA	0.51	0.53	0.68	0.53	0.52	0.55	0.51	0.50	0.52
		AA-PSA	0.20	0.37	0.06	0.23	0.27	0.44	0.20	0.23	0.16
		N-PSA	0.01	0.00	0.19	0.09	0.10	0.01	0.03	0.05	0.04
80	0.6	*AA-SA*	0.99	0.90	0.77	0.91	1.05	1.08	0.93	0.93	0.86
		N-SA	0.48	0.48	0.65	0.49	0.53	0.58	0.49	0.59	0.55
		AA-PSA	0.32	0.27	0.16	0.27	0.38	0.33	0.29	0.23	0.21
		N-PSA	0.03	0.04	0.14	0.03	0.03	0.12	0.05	0.07	0.06
80	1.0	*AA-SA*	0.80	0.74	0.83	1.10	0.95	0.95	0.83	0.93	0.83
		N-SA	0.58	0.58	0.60	0.47	0.60	0.56	0.61	0.55	0.54
		AA-PSA	0.19	0.14	0.16	0.47	0.26	0.26	0.21	0.24	0.21
		N-PSA	0.15	0.12	0.14	0.04	0.15	0.09	0.11	0.03	0.06

which were improved by the dominance relation in the N-SA and N-PSA algorithms, affected the final solutions obtained by the two algorithms, although the effect was small in some cases.

The experiments indicated that the overall errors the AA-SA, N-SA, LZ-HGA, MZY-SA, AA-PSA, and N-PSA algorithms were 0.75, 0.46, 4.54, 6.15, 0.22, and 0.10, respectively. Therefore, the AA-SA, N-SA, AA-PSA, and N-PSA algorithms clearly performed better than the LZ-HGA and MZY-SA algorithms. Moreover, the N-SA algorithm performed better than AA-SA and the N-PSA algorithm performed better than AA-PSA. In particular, the N-SA algorithm decreased the error AA-SA by about 74% while the N-PSA algorithm reduced the error AA-PSA by about 52%.

When the setup times are zero, the AA-SA and AA-PSA algorithms reduce to the SA and PSA algorithms, respectively, which are known to be the best algorithms for this problem according to previous studies. We found that the proposed N-SA and N-PSA algorithms reduced the error compared with the best algorithms, i.e., SA and PSA, by about 71% and 48%,

Table 7
Percentage errors with N-PSA for small size problems.

		k						
		0.5			1			
		m			m			
	n	5	10	12	5	10	12	Average
$T = 0.4$, $R = 0.6$	6	0.15	0.00	0.00	0.06	0.00	0.03	0.04
	7	0.00	0.05	0.07	0.04	0.09	0.08	0.06
	8	0.01	0.07	0.09	0.07	0.09	0.03	0.06
	9	0.06	0.10	0.02	0.00	0.05	0.00	0.04
	10	0.03	0.01	0.02	0.17	0.02	0.07	0.05
$T = 0.4$, $R = 1.0$	6	0.01	0.00	0.01	0.03	0.04	0.00	0.02
	7	0.00	0.00	0.07	0.08	0.00	0.00	0.03
	8	0.04	0.05	0.06	0.02	0.07	0.10	0.06
	9	0.08	0.02	0.00	0.04	0.02	0.07	0.04
	10	0.04	0.03	0.25	0.00	0.07	0.10	0.08
$T = 0.6$, $R = 0.6$	6	0.00	0.00	0.05	0.01	0.00	0.01	0.01
	7	0.04	0.03	0.00	0.08	0.00	0.01	0.03
	8	0.03	0.00	0.04	0.08	0.17	0.02	0.06
	9	0.02	0.07	0.11	0.00	0.02	0.04	0.05
	10	0.25	0.04	0.02	0.18	0.09	0.11	0.12
$T = 0.6$, $R = 1.0$	6	0.00	0.00	0.00	0.00	0.02	0.00	0.00
	7	0.02	0.00	0.00	0.02	0.16	0.11	0.05
	8	0.06	0.01	0.01	0.06	0.03	0.08	0.04
	9	0.00	0.00	0.10	0.02	0.05	0.07	0.04
	10	0.01	0.20	0.10	0.00	0.00	0.04	0.06
Average		0.04	0.03	0.05	0.05	0.05	0.05	0.05

respectively. Therefore, the N-SA and N-PSA algorithms outperformed the previous best algorithms for the problem with zero setup times.

In summary, we recommend using N-PSA because this algorithm reduced the error obtained with N-SA by about 77% and that with AA-PSA by about 52%.

In this study, we considered sequence-independent setup times, but the setup times are sequence-dependent for some manufacturing systems (e.g., see Ahmadizar and Shahmaleki [1] and Keshavarz et al. [17]). Therefore, a possible extension could include investigating the problem with respect to sequence-dependent setup times. Moreover, we proposed a deterministic model in the current study but some environments may require a model that involves uncertainty (e.g., Aydilek and Allahverdi [12]). Therefore, another possible extension would be to address the problem with uncertain processing or setup times. A further possible extension might be to utilize our dominance relation in implicit enumeration techniques, such as a branch-and-bound algorithm.

Acknowledgments

This research was supported by Kuwait University Research Administration project number EI01/13. We would like to thank the anonymous referees and the associate editor for their thorough reviews and comments on four revisions, which improved the quality and presentation of the paper.

References

[1] F. Ahmadizar, P. Shahmaleki, Group-shop scheduling with sequence-dependent set-up and transportation times, Appl. Math. Model. 38 (2014) 5080–5091.
[2] F.S. Al-Anzi, A. Allahverdi, Heuristics for a two-stage assembly flowshop with bicriteria of maximum lateness and makespan, Comput. Oper. Res. 36 (2009) 2682–2689.
[3] F.S. Al-Anzi, A. Allahverdi, A self-adaptive differential evolution heuristic for two-stage assembly scheduling problem to minimize maximum lateness with setup times, Eur. J. Oper. Res. 182 (2007) 80–94.
[4] F.S. Al-Anzi, A. Allahverdi, A hybrid tabu search heuristic for the two-stage assembly scheduling problem, Int. J. Oper. Res. 3 (2006) 109–119.
[5] A. Allahverdi, The third comprehensive survey on scheduling problems with setup times/costs, Eur. J. Oper. Res. 246 (2015) 345–378.
[6] A. Allahverdi, F.S. Al-Anzi, The two-stage assembly scheduling problem to minimize total completion time with setup times, Comput. Oper. Res. 36 (2009) 2740–2747.
[7] A. Allahverdi, F.S. Al-Anzi, The two-stage assembly flowshop scheduling problem with bicriteria of makespan and mean completion time, Int. J. Adv. Manuf. Technol. 37 (2008) 166–177.
[8] A. Allahverdi, F.S. Al-Anzi, Evolutionary heuristics and an algorithm for the two-stage assembly scheduling problem to minimize makespan with setup times, Int. J. Prod. Res. 44 (2006) 4713–4735.
[9] A. Allahverdi, F.S. Al-Anzi, A PSO and a tabu search heuristics for assembly scheduling problem of the two-stage distributed database application, Comput. Oper. Res. 33 (2006) 1056–1080.
[10] A. Allahverdi, H. Aydilek, The two stage assembly flowshop scheduling problem to minimize total tardiness, J. Intell. Manuf. 26 (2015) 225–237.
[11] A. Allahverdi, C.T. Ng, T.C.E. Cheng, M.Y. Kovalyov, A survey of scheduling problems with setup times or costs, Eur. J. Oper. Res. 187 (2008) 985–1032.
[12] H. Aydilek, A. Allahverdi, A polynomial time heuristic for the two-machine flowshop scheduling problem with setup times and random processing times, Appl. Math. Modell. 37 (2013) 7164–7173.

[13] V.M. Dalfard, A. Ardakani, T.N. Banihashemi, Hybrid genetic algorithm for assembly flow-shop scheduling problem with sequence-dependent setup and transportation times, Tehnickivjesnik 18 (2012) 497–504.
[14] N.G. Hall, M.E. Posner, Generating experimental data for computational testing with machine scheduling applications, Oper. Res. 49 (2001) 854–865.
[15] A.M.A Hariri, C.N. Potts, A branch and bound algorithm for the two-stage assembly scheduling problem, Eur. J. Oper. Res. 103 (1997) 547–556.
[16] M. Haouari, T. Daouas, Optimal scheduling of the 3-machine assembly-type flow shop, RAIRO Rech. Oper. 33 (1999) 439–445.
[17] T. Keshavarz, M. Savelsbergh, N. Salmasi, A branch-and-bound algorithm for the single machine sequence-dependent group scheduling problem with earliness and tardiness penalties, Appl. Math. Model. 39 (2015) 6410–6424.
[18] Y.D. Kim, A new branch and bound algorithm for minimizing mean tardiness in two-machine flowshops, Comput. Oper. Res. 20 (1993) 391–401.
[19] C.Y. Lee, T.C.E. Cheng, B.M.T. Lin, Minimizing the makespan in the 3-machine assembly-type flowshop scheduling problem, Manag. Sci. 39 (1993) 616–625.
[20] X. Li, Y. Zhang, Adaptive hybrid algorithms for the sequence-dependent setup time permutation flow shop scheduling problem, IEEE Trans. Autom. Sci. Eng., 9 (2012) 578–595.
[21] Y. Maboudian, R. Shafaei, Modeling a bi-criteria two stage assembly flow shop scheduling problem with sequence dependent setup times, in: Proceedings of IEEE International Conference on Industrial Engineering and Engineering Management, 2009, pp. 1748–1752.
[22] E. Mehdizadeh, R. Tavakkoli-Moghaddam, M. Yazdani, A vibration damping optimization algorithm for a parallel machines scheduling problem with sequence-independent family setup times, Appl. Math. Model. 39 (2015) 6845–6859.
[23] S.M. Mousavi, M. Zandieh, M. Yazdani, A simulated annealing/local search to minimize the makespan and total tardiness on a hybrid flowshop, Int. J. Adv. Manuf. Technol. 64 (2013) 369–388.
[24] A. Mozdgir, S.M.T. FatemiGhomi, F. Jolai, J. Navaei, Two-stage assembly flow-shop scheduling problem with non-identical assembly machines considering setup times, Int. J. Prod. Res. 51 (2013) 3625–3642.
[25] C.N. Potts, S.V. Sevast'janov, V.A. Strusevich, L.N. Van Wassenhove, C.M. Zwaneveld, The two-stage assembly scheduling problem: complexity and approximation, Oper. Res. 43 (1995) 346–355.
[26] D.K. Seo, C.M. Klein, W. Jang, Single machine stochastic scheduling to minimize the expected number of tardy jobs using mathematical programming models, Comput. Ind. Eng. 48 (2005) 153–161.
[27] E. Shokrollahpour, M. Zandieh, B. Dorri, A novel imperialist competitive algorithm for bi-criteria scheduling of the assembly flowshop problem, Int. J. Prod. Res. 49 (2011) 3087–3103.
[28] X. Sun, K. Morizawa, H. Nagasawa, Powerful heuristics to minimize makespan in fixed, 3-machine, assembly-type flowshop scheduling, Eur. J. Oper. Res. 146 (2003) 498–516.
[29] Y. Tian, D. Liu, D. Yuan, K. Wang, A discrete PSO for two-stage assembly scheduling problem, Int. J. Adv. Manuf. Technol. 66 (2013) 481–499.
[30] E. Torabzadeh, M. Zandieh, Cloud theory-based simulated annealing approach for scheduling in the two-stage assembly flowshop, Adv. Eng. Softw. 41 (2010) 1238–1243.
[31] A. Tozkapan, O. Kirca, C.S. Chung, A branch and bound algorithm to minimize the total weighted flowtime for the two-stage assembly scheduling problem, Comput. Oper. Res. 30 (2003) 309–320.
[32] E. Vallada, R. Ruiz, Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem, OMEGA The Int. J. Manag. Sci. 38 (2010) 57–67.