Research Article

A Two-Stage Assembly-Type Flowshop Scheduling Problem for Minimizing Total Tardiness

Ju-Yong Lee¹ and June-Young Bang²

¹System Engineering Team, Device Solution Business, Samsung Electronics Co., Ltd., Yongin-si, Gyeonggi-do 17113, Republic of Korea

²Department of Industrial Management Engineering, Sungkyul University, Anyang-si, Gyeonggi-do 14097, Republic of Korea

Correspondence should be addressed to June-Young Bang; jybang@sungkyul.ac.kr

Received 11 March 2016; Accepted 20 June 2016

Academic Editor: Fons J. Verbeek

Copyright © 2016 J.-Y. Lee and J.-Y. Bang. 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.

This research considers a two-stage assembly-type flowshop scheduling problem with the objective of minimizing the total tardiness. The first stage consists of two independent machines, and the second stage consists of a single machine. Two types of components are fabricated in the first stage, and then they are assembled in the second stage. Dominance properties and lower bounds are developed, and a branch and bound algorithm is presented that uses these properties and lower bounds as well as an upper bound obtained from a heuristic algorithm. The algorithm performance is evaluated using a series of computational experiments on randomly generated instances and the results are reported.

1. Introduction

This paper focuses on the problem of scheduling a given set of jobs with different due dates in a two-stage assembly-type flowshop in which the first stage consists of two independent machines and the second stage consists of a single machine. In the first (or subassembly) stage, two types of components are fabricated separately and independently on the two different machines; then, they are assembled into the final product in the second (or assembly) stage. In the system, the two components can be produced on the two machines in the first stage at the same time, while they are fed into the second stage only after both components are completed. In this paper, a branch and bound algorithm is presented for a two-stage assembly-type flowshop scheduling problem with the objective of minimizing the total tardiness.

This study considers a two-stage assembly-type flowshop scheduling problem to minimize total tardiness. This scheduling problem is denoted by $AF2 \parallel \sum T_i$ in the three-field notation of Graham et al. [1], where $AF2$ refers to a two-stage assembly-type flowshop with two machines in the first stage and T_i is the tardiness of job i defined as $T_i = \max\{C_i - d_i, 0\}$, where C_i and d_i are the completion time and due date of job i , respectively. Note that a typical two-machine flowshop

is a special case of a two-stage assembly-type flowshop if the first stage has one machine instead of two independent machines. Thus, the problem considered in this study can be easily proven to be NP-hard in the strong sense because an ordinary two-machine flowshop scheduling problem with the objective of minimizing the total tardiness is NP-hard in the strong sense [2].

Two-stage assembly-type flowshops are found in various manufacturing systems. For example, Lee et al. [3] described a fire engine assembly system in which the body shell and chassis are produced independently on two machines and then brought into a final assembly stage. Furthermore, Sung and Kim [4] considered a table assembly system where the board and legs are made separately in the first stage and then they are assembled together in the second stage; and they also considered a cross-docking distribution problem where two products are gathered and then repacked. In addition, many practical problems can be modeled as two-stage assembly-type flowshops.

There have been many studies on two-stage assembly-type flowshop scheduling problems. For problems with the objective of minimizing makespan, Lee et al. [3] proved that the problem is strongly NP-complete, although there are only two machines in the first stage; they also discussed some

polynomially solvable cases of the problem and proposed a branch and bound (B&B) algorithm. Furthermore, Potts et al. [5] showed that the search for an optimal solution may be restricted to permutation schedules. In addition, Hariri and Potts [6] and Haouari and Daouas [7] proposed B&B algorithms, while Sun et al. [8] develop heuristic algorithms. On the other hand, Koulamas and Kyparisis [9] generalized the problem into a three-stage assembly-type flowshop and proposed heuristic algorithms. Furthermore, Sung and Juhn [10] proposed a B&B algorithm for the problem in which one type of components is outsourced subject to the job-dependent lead-time in the first stage.

In addition, for problems with the objective of minimizing the total (weighted) completion time, Tozkan et al. [11] developed two heuristic algorithms and a B&B algorithm, and Al-Anzi and Allahverdi [12] proposed several heuristic algorithms, while Allahverdi and Al-Anzi [13] proposed three heuristic algorithms for the problem considering separate setup times. On the other hand, Sung and Kim [4] developed a B&B algorithm and heuristic algorithm for the problem in which the second stage consists of two identical parallel machines.

Although there have been many studies on problems with the objective of minimizing the makespan or total flow time, there has not been significant progress in problems with respect to the performance measures related to due dates. Allahverdi and Al-Anzi [14] proposed a mathematical formulation and heuristic algorithms for a problem with the objective of minimizing the maximum tardiness, and Al-Anzi and Allahverdi [15] addressed the same problem while considering separate setup times. In addition, Allahverdi and Aydilek [16] proposed heuristic algorithms in order to minimize the total tardiness.

In this study, a branch and bound (B&B) algorithm is presented for the two-stage assembly-type flowshop scheduling problem with the objective of minimizing the total tardiness. Dominance properties and lower bounds are developed, and these can be used in the B&B algorithm. Also, a heuristic algorithm is proposed that can be used to find an (initial) upper bound in the B&B algorithm as well as to find a good solution quickly. Note that this study is the first attempt to develop an optimal solution algorithm for the problem considered in this paper, that is, $AF2 \parallel \sum T_i$.

The remainder of this paper is organized as follows. In the next section, we describe the problem considered in this study in more detail and give a mixed integer programming formulation for the problem. Then, the dominance properties associated with the problem are given in Section 3, and the lower bounds and a B&B algorithm are given in Section 4. For the evaluation of the performance of the B&B algorithm, a series of computational experiments are performed and the results are reported in Section 5. Finally, Section 6 concludes the paper with a short summary.

2. Problem Description

In this paper, a two-stage assembly-type flowshop scheduling problem is considered. There are two machines in the first

stage, while there is a single machine in the second stage. There are n jobs ($J_1, J_2, \dots, J_n$) to be processed in the flowshop and each job (say job i) consists of three operations (O_{ia} , O_{ib} , and O_{il}). The first two operations (O_{ia} and O_{ib}) are performed separately and independently on different machines (M_a and M_b , resp.) in the first stage and the last operation (O_{il}) is performed on a single machine (M_l) in the second stage. Here, it is assumed that O_{il} can be started only after both O_{ia} and O_{ib} are completed in the first stage. Other assumptions include the following:

- (1) All jobs are available at time zero.
- (2) Preemption of the jobs is not allowed.
- (3) There is no precedence relationship between the jobs.
- (4) There is no setup time required before a job is processed.
- (5) Machine idle time in the first stage is not allowed.

In this problem, there are $(n!)^3$ alternatives for ordering jobs on the machines. However, there is an optimal solution among $n!$ permutation schedules in which the operating sequences of the jobs are the same on the machines. This is because Tozkan et al. [11] proved that permutation schedules are dominant for the two-stage assembly scheduling problem with the objective of minimizing the total completion time and their proof is sufficient to show that permutation schedules are dominant for any regular performance measure, including total tardiness. Therefore, only permutation schedules are considered in order to find an optimal solution. That is, the sequence of jobs on the three machines (two in the first stage and one in the second stage) is the same.

A mathematical formulation of the problem considered in this study is provided below. The notation that is used in the formulation and throughout this paper is provided in Notation.

Now, a mixed integer program is given:

$$[P] \text{ Minimize } \sum_{r=1}^n T_{[r]} \quad (1)$$

$$\text{subject to } T_{[r]} \geq C_{[r]} - \sum_{i=1}^n d_i x_{ir} \quad r = 1, \dots, n \quad (2)$$

$$C_{[r]} = s_{[r]l} + \sum_{i=1}^n p_{il} x_{ir} \quad \forall r \quad (3)$$

$$s_{[r]l} \geq s_{[r]a} + \sum_{i=1}^n p_{ia} x_{ir} \quad \forall r \quad (4)$$

$$s_{[r]l} \geq s_{[r]b} + \sum_{i=1}^n p_{ib} x_{ir} \quad \forall r \quad (5)$$

$$s_{[r+1]l} \geq C_{[r]} \quad r = 1, \dots, n-1 \quad (6)$$

$$s_{[r+1]a} \geq s_{[r]a} + \sum_{i=1}^n p_{ia} x_{ir} \quad (7)$$

$$r = 1, \dots, n-1$$

$$s_{[r+1]b} \geq s_{[r]b} + \sum_{i=1}^n p_{ib} x_{ir} \quad (8)$$

$$r = 1, \dots, n-1$$

$$s_{[1]a} = 0 \quad (9)$$

$$s_{[1]b} = 0 \quad (10)$$

$$\sum_{r=1}^n x_{ir} = 1 \quad i = 1, \dots, n \quad (11)$$

$$\sum_{i=1}^n x_{ir} = 1 \quad \forall r \quad (12)$$

$$T_{[r]}, C_{[r]}, s_{[r]a}, s_{[r]b}, s_{[r]l} \geq 0 \quad \forall r \quad (13)$$

$$x_{ir} \in \{0, 1\} \quad \forall i, r. \quad (14)$$

The objective function, which is to be minimized, represents the total tardiness. Constraints (2) and (3) are used to compute the tardiness and completion time of each job. Constraint sets (4)–(6) specify the starting time of each job on the machine in the second stage. Also, constraint sets (7)–(10) specify the starting times of each job on the machines in the first stage. Constraints (11) and (12) ensure that each job is placed at only one position in the sequence and only one job can be placed at each position, respectively. Constraints (13) and (14) define the domain of the decision variables.

3. Dominance Properties

In this section, the dominance properties of the problem considered in this study are developed, which are used in the branch and bound (B&B) algorithm proposed in this paper. These properties are used to delete nodes (in the B&B tree) associated with partial schedules that are dominated by others in the B&B algorithm. A partial schedule (σ) is said to be *dominated* by another partial schedule (σ') when it results in a complete schedule better than the best possible complete schedule resulting from σ . Thus, when a node (corresponding to a partial schedule) is generated in the B&B algorithm, it can be deleted if its corresponding partial schedule is dominated by another partial schedule. The notation that is used in the statements and proofs of the properties and for the description of the algorithms presented in this paper is provided in Notation.

The following propositions provide the dominance properties that are used to identify the partial schedules that are dominated by others. First, Propositions 1 and 2 are obtained through modifying the dominance conditions given in Schaller [17].

Proposition 1. *For any partial schedule (σ) in which job i is scheduled last and job j immediately precedes job i and another partial schedule (σ') which is identical to σ except that jobs i and j are interchanged, if (a) $C_j(\sigma') \leq d_j$ and (b1) $C_j(\sigma') \leq C_i(\sigma)$ or (b2) $C_j(\sigma') \leq \Delta(\sigma')$, σ is dominated by*

σ' , where $\Delta(\sigma') = \max\{c_{j1}(\sigma') + \min_{q \notin S(\sigma')} \{p_{q1}\}, c_{j2}(\sigma') + \min_{q \notin S(\sigma')} \{p_{q2}\}\}$.

Proof. Because there is no machine idle time in the first stage, the completion times of the last jobs on the machines are the same in both partial schedules σ and σ' . If (a) is satisfied, job j is not tardy in σ and σ' ; that is, $T_j(\sigma) = T_j(\sigma') = 0$. Also, because job i is completed earlier in σ' than in σ , the tardiness of job i may be decreased or unchanged; that is, $T_i(\sigma) \geq T_i(\sigma')$. On the other hand, if (b1) is satisfied, the interchange does not delay the starting time of the next job to be scheduled after σ' . If (b2) is satisfied, the starting time of the next job to be scheduled after σ' does not depend on the completion time of job j in σ' because $\Delta(\sigma') = \max\{c_{j1}(\sigma') + \min_{q \notin S(\sigma')} \{p_{q1}\}, c_{j2}(\sigma') + \min_{q \notin S(\sigma')} \{p_{q2}\}\}$ is the lower bound on the completion time of the next job in the first stage after σ' . Consequently, the interchange can decrease the tardiness of jobs i and j , and it does not delay the starting time of the next job to be scheduled after σ' . Therefore, σ is dominated by σ' . $\square$

Proposition 2. *For any partial schedule (σ) in which job i is scheduled last and job j is completed earlier than job i and another partial schedule (σ') which is identical to σ except that jobs i and j are interchanged, if (a) $\sum_{q \in S_{ji}(\sigma)} T_q(\sigma) \geq \sum_{q \in S_{ji}(\sigma')} T_q(\sigma')$ and (b) $\sum_{q \in S_{ji}(\sigma)} T_q(\sigma) - \sum_{q \in S_{ji}(\sigma')} T_q(\sigma') \geq n^u(\sigma) \cdot [C_j(\sigma') - \max\{C_i(\sigma), \Delta(\sigma')\}]$, σ is dominated by σ' , where $\Delta(\sigma') = \max\{c_{j1}(\sigma') + \min_{q \notin S(\sigma')} \{p_{q1}\}, c_{j2}(\sigma') + \min_{q \notin S(\sigma')} \{p_{q2}\}\}$.*

Proof. If (a) is satisfied, because σ' is obtained by interchanging job i and job j only, the tardiness of σ' is no greater than the tardiness of σ . However, the starting time of each job to be scheduled after job j in a complete schedule resulting from σ' can be increased by at most $C_j(\sigma') - \max\{C_i(\sigma), \Delta(\sigma')\}$. Then, the increase of the total tardiness caused by this interchange is less than or equal to $n^u(\sigma) \cdot [C_j(\sigma') - \max\{C_i(\sigma), \Delta(\sigma')\}] - \sum_{q \in S_{ji}(\sigma)} T_q(\sigma) + \sum_{q \in S_{ji}(\sigma')} T_q(\sigma')$, which is less than or equal to 0. Therefore, σ is dominated by σ' . $\square$

The following two propositions show the dominance properties related to two adjacent jobs scheduled in a partial schedule.

Proposition 3. *For any partial schedule (σ) in which job i is scheduled last and job j immediately precedes job i and another partial schedule (σ') which is identical to σ except that jobs i and j are interchanged, if (a) $C_j(\sigma') \leq C_i(\sigma)$ and $C_i(\sigma') \leq C_j(\sigma)$ and (b) $d_i \leq d_j$, σ is dominated by σ' .*

Proof. Because $C_j(\sigma') \leq C_i(\sigma)$, the interchange does not delay the starting time of the next job to be scheduled after σ' . Thus, in order to prove the proposition, it must be proved that the tardiness may be decreased or unchanged by interchanging job i and job j in the partial schedule σ . Let ΔT_i

and ΔT_j be the changes in the tardiness of job i and job j due to the interchange, respectively. Then,

$$\begin{aligned}\Delta T_i &= \max \{C_i(\sigma') - d_i, 0\} - \max \{C_i(\sigma) - d_i, 0\} \\ &= \max \{C_i(\sigma'), d_i\} - \max \{C_i(\sigma), d_i\}, \\ \Delta T_j &= \max \{C_j(\sigma') - d_j, 0\} - \max \{C_j(\sigma) - d_j, 0\} \\ &= \max \{C_j(\sigma'), d_j\} - \max \{C_j(\sigma), d_j\},\end{aligned}\quad (15)$$

because $\max\{x_1 - x_2, 0\} - \max\{x_3 - x_2, 0\} = \max\{x_1, x_2\} - \max\{x_3, x_2\}$ for any three real numbers x_1, x_2 , and x_3 . The following cases are considered in order to determine the relationship between $\Delta T_i + \Delta T_j$ and 0.

Case 1. If $d_j > C_j(\sigma')$, then $d_j > C_j(\sigma') > C_j(\sigma)$; hence, $\Delta T_j = 0$. Then, $\Delta T_i + \Delta T_j \leq 0$.

Case 2. If $C_j(\sigma') > d_j > C_j(\sigma)$, then $C_i(\sigma) \geq C_j(\sigma') > d_j > d_i$ and $d_j > C_j(\sigma) \geq C_i(\sigma')$. Then,

$$\begin{aligned}\Delta T_i + \Delta T_j &= (\max \{C_i(\sigma'), d_i\} - C_i(\sigma)) \\ &\quad + (C_j(\sigma') - d_j) \\ &= (\max \{C_i(\sigma'), d_i\} - d_j) \\ &\quad + (C_j(\sigma') - C_i(\sigma)) \leq 0.\end{aligned}\quad (16)$$

Case 3. If $C_j(\sigma) > d_j$, then $C_i(\sigma) \geq C_j(\sigma') > C_j(\sigma) > d_j > d_i$ and $C_j(\sigma) \geq \max\{C_i(\sigma'), d_i\}$. Then,

$$\begin{aligned}\Delta T_i + \Delta T_j &= (\max \{C_i(\sigma'), d_i\} - C_i(\sigma)) \\ &\quad + (C_j(\sigma') - C_j(\sigma)) \\ &= (\max \{C_i(\sigma'), d_i\} - C_j(\sigma)) \\ &\quad + (C_j(\sigma') - C_i(\sigma)) \leq 0.\end{aligned}\quad (17)$$

Therefore, the sum of tardiness is decreased or unchanged through the interchange of the two jobs. This completes the proof. $\square$

Proposition 4. For any partial schedule (σ) in which job i is scheduled last and job j immediately precedes job i and another partial schedule (σ') which is identical to σ except that jobs i and j are interchanged, if $d_i \leq d_j$, $p_{il} - d_i \leq p_{jl} - d_j$, $\max\{p_{ia}, p_{ib}\} \leq p_{il}$, and $s_{il}(\sigma') \leq s_{jl}(\sigma)$, σ is dominated by σ' .

Proof. First, it is shown that $C_j(\sigma') \leq C_i(\sigma)$. Let s^a and s^b denote the start time of job j on machine a and machine b , respectively, in schedule σ . Then,

$$\begin{aligned}C_i(\sigma) &= \max \{s^a + p_{ja} + p_{ia}, s^b + p_{jb} + p_{ib}, s_{jl}(\sigma) + p_{jl}\} \\ &\quad + p_{il} \\ C_j(\sigma') &= \max \{s^a + p_{ia} + p_{ja}, s^b + p_{ib} + p_{jb}, s_{il}(\sigma') + p_{il}\} \\ &\quad + p_{jl}.\end{aligned}\quad (18)$$

Because $\max\{s^a + p_{ja}, s^b + p_{jb}\} \leq s_{jl}(\sigma)$ and $\max\{p_{ia}, p_{ib}\} \leq p_{il} \leq p_{jl}$, $\max\{s^a + p_{ja} + p_{ia}, s^b + p_{jb} + p_{ib}, s_{jl}(\sigma) + p_{jl}\} = s_{jl}(\sigma) + p_{jl}$, and, hence, $C_i(\sigma) = s_{jl}(\sigma) + p_{jl} + p_{il}$. The following are considered in order to determine the relationship between $C_j(\sigma') - C_i(\sigma)$ and 0.

Case 1. If $\max\{s^a + p_{ia} + p_{ja}, s^b + p_{ib} + p_{jb}, s_{il}(\sigma') + p_{il}\} = s^a + p_{ia} + p_{ja}$, then $C_j(\sigma') = s^a + p_{ia} + p_{ja} + p_{jl}$. Then, because $s^a + p_{ja} \leq s_{jl}(\sigma)$ and $p_{ia} \leq p_{il}$, $C_j(\sigma') - C_i(\sigma) = \{(s^a + p_{ja}) - s_{jl}(\sigma)\} + (p_{ia} - p_{il}) \leq 0$.

Case 2. If $\max\{s^a + p_{ia} + p_{ja}, s^b + p_{ib} + p_{jb}, s_{il}(\sigma') + p_{il}\} = s^b + p_{ib} + p_{jb}$, then $C_j(\sigma') = s^b + p_{ib} + p_{jb} + p_{jl}$. Then, because $s^b + p_{jb} \leq s_{jl}(\sigma)$ and $p_{ib} \leq p_{il}$, $C_i(\sigma) - C_j(\sigma') = \{(s^b + p_{jb}) - s_{jl}(\sigma)\} + (p_{ib} - p_{il}) \leq 0$.

Case 3. If $\max\{s^a + p_{ia} + p_{ja}, s^b + p_{ib} + p_{jb}, s_{il}(\sigma') + p_{il}\} = s_{il}(\sigma') + p_{il}$, then $C_j(\sigma') = s_{il}(\sigma') + p_{il} + p_{jl}$. Then, because $s_{il}(\sigma') \leq s_{jl}(\sigma)$, $C_j(\sigma') - C_i(\sigma) = s_{il}(\sigma') - s_{jl}(\sigma) \leq 0$.

Because $C_j(\sigma') \leq C_i(\sigma)$, the interchange does not delay the starting time of the next job to be scheduled after σ' . Thus, in order to prove the proposition, it is only needed to be shown that $T_i(\sigma') + T_j(\sigma') \leq T_j(\sigma) + T_i(\sigma)$. Then,

$$\begin{aligned}T_j(\sigma) &= \max \{0, s_{jl}(\sigma) + p_{jl} - d_j\} \\ T_i(\sigma) &= \max \{0, s_{il}(\sigma) + p_{il} - d_i\} \\ T_i(\sigma') &= \max \{0, s_{il}(\sigma') + p_{il} - d_i\} \\ T_j(\sigma') &= \max \{0, \\ &\quad \max \{s^a + p_{ia} + p_{ja}, s^b + p_{ib} + p_{jb}, s_{il}(\sigma') + p_{il}\} \\ &\quad + p_{jl} - d_j\}.\end{aligned}\quad (19)$$

Because $p_{il} - d_i \leq p_{jl} - d_j$ and $s_{il}(\sigma') \leq s_{jl}(\sigma)$, $T_i(\sigma') \leq T_j(\sigma)$. Also, because $C_j(\sigma') \leq C_i(\sigma)$ and $d_i \leq d_j$, $T_j(\sigma') \leq T_i(\sigma)$. Therefore, the sum of tardiness is decreased or unchanged by the interchange of the two jobs. This completes the proof. $\square$

The following proposition is extended from a dominance rule that is used for a single-machine weighted tardiness problem, suggested by Potts and Van-Wassenhove [18].

Proposition 5. Consider two partial schedules, that is, a partial schedule (σ) and another partial schedule (σ') , which consist of the same set of jobs included in σ ; that is, $S(\sigma) = S(\sigma')$ and the sequences are different. If (a) $\Gamma_l(\sigma') \leq \Gamma_l(\sigma)$ and (b) $\sum_{i \in S(\sigma')} T_i(\sigma') \leq \sum_{i \in S(\sigma)} T_i(\sigma)$, σ is dominated by σ' .

Proof. Because the two partial schedules (σ) and (σ') consist of the same set of jobs and only the sequences are different, $\Gamma_a(\sigma) = \Gamma_a(\sigma')$ and $\Gamma_b(\sigma) = \Gamma_b(\sigma')$. If (a) is satisfied, the starting time of the next job to be scheduled after σ' is no greater than the starting time of the next job to be scheduled after σ . Furthermore, the tardiness of the jobs under σ' is no greater than that under σ . Therefore, if $\Gamma_l(\sigma') \leq \Gamma_l(\sigma)$ and $\sum_{i \in S(\sigma')} T_i(\sigma') \leq \sum_{i \in S(\sigma)} T_i(\sigma)$, σ is dominated by σ' . $\square$

4. Branch and Bound Algorithm

This section presents the proposed branch and bound (B&B) algorithm for the problem considered in this study. In the following, the methods to construct a B&B tree in the algorithm, to obtain an initial feasible solution (or upper bound), and to compute lower bounds associated with nodes generated in the B&B tree are described.

4.1. Branching. In this algorithm, the B&B tree is constructed as follows. Each node in the B&B tree corresponds to a partial schedule, and a node at the j th level in the tree corresponds to a partial schedule in which the first j jobs are scheduled in front of the complete schedules that result from the partial schedule. Thus, when a node branches, or when child nodes are generated from a parent node or when a job is newly included in the partial schedule associated with the node, the new job to be included is appended at the end of the partial schedule. Because the sequences of n jobs must be considered in this scheduling problem, a complete schedule is at the n th level in the B&B tree.

In the B&B algorithm, an initial upper bound, or initial feasible solution, is obtained at the root node of the B&B tree using a heuristic algorithm to be presented in the next section. For branching, that is, for selecting a node to generate branches from, the depth-first rule is employed in the algorithm. That is, a node with the most jobs included in the partial schedule that it is associated with is selected for branching. In case of ties, a node with the minimum lower bound is selected. When child nodes are generated from a selected parent node, it is verified whether the schedules associated with the child nodes are dominated by others using the dominance properties given in the previous section. For each child node that is not dominated, a lower bound is computed. Nodes with lower bounds greater than or equal to the current upper bound are deleted from further consideration. The upper bound is updated whenever a complete schedule better than the current upper bound is found. The algorithm is terminated if there are no more

dangling nodes in the B&B tree, which can be selected for branching.

4.2. Upper Bound. In order to obtain an initial upper bound at the root node of the B&B tree, a simple heuristic algorithm that consists of two phases (one for obtaining an initial solution and one for improving it) is used. The initial solution is obtained using the heuristic algorithm that is modified from a constructive algorithm developed by Nawaz et al. [19], called NEH, and, then, the solution is improved using pairwise interchanges of jobs. The NEH is known to work well in flowshop scheduling problems and it has been modified by many other researchers for various scheduling problems. The following summarizes the procedure of the two-phase heuristic, called mNEH-PI.

Procedure 1 (mNEH-PI).

Step 0. Obtain a seed sequence by sorting jobs in nondecreasing order of due dates. Let σ be this sequence and set $\sigma^* = \emptyset$, where σ^* is the (partial) schedule constructed so far in the heuristic. Set $r = 1$. Go to Step 1.

Step 1. Select the r th job in σ . Go to Step 2.

Step 2. For $i = 1$ to $|\sigma^*| + 1$, do the following: insert the selected job into the i th position of σ^* and then compute the total tardiness of the schedule that is composed of the schedule of these $|\sigma^*| + 1$ jobs followed by the sequence of jobs that are placed at the $(r + 1)$ th position through the n th position in σ . Go to Step 3.

Step 3. Select a partial schedule that results in the minimum total tardiness in Step 2. Let σ^* be the selected (partial) schedule. Go to Step 4.

Step 4. Let $r \leftarrow r + 1$. If $r \leq n$, go to Step 1; otherwise, go to Step 5. (σ^* is the initial solution of the first phase.)

Step 5. Set $i = 1$ and $j = 2$. Go to Step 6.

Step 6. Generate a new sequence by interchanging two jobs in the i th and j th positions in σ^* . If the total tardiness of the new sequence is less than that of σ^* , then replace σ^* with the new sequence and go to Step 5; otherwise, go to Step 7.

Step 7. Let $j \leftarrow j + 1$. If $j \leq n$, go to Step 6; otherwise, let $i \leftarrow i + 1$ and $j \leftarrow i + 1$. If $i < n$, go to Step 6; otherwise, terminate. (σ^* is the final solution of this heuristic.)

4.3. Lower Bound. In the B&B algorithm, a lower bound on the total tardiness (of the complete schedules resulting from the partial schedule associated with a node) is computed for each node that is not eliminated by the dominance rules given in the previous section. In the algorithm, two lower bounds are used and computed for jobs that are not included in the partial schedule associated with the current node. These bounds are based on Theorem 6, which was proposed by Chu

[20] and independently by Kondakci et al. [21] and Kim [22]. Here, $d_{[j]}$ denotes the due date of a job with the j th earliest due date among jobs in a partial schedule.

Theorem 6. For nonnegative real numbers c_j and d_j corresponding to job j , if $C_1 \leq C_2 \leq \dots \leq C_n$, then the following inequality holds:

$$\sum_{j=1}^n \max\{C_j - d_{[j]}, 0\} \leq \sum_{j=1}^n \max\{C_j - d_j, 0\}. \quad (20)$$

Based on Theorem 6, if lower bounds on the completion times of the jobs can be computed, a lower bound on the total tardiness of the jobs can be calculated.

The first lower bound (LB_1) is developed based on the following proposition, which gives a lower bound on the completion time of the job that is completed j th in a complete schedule. The proposition is obtained by modifying the result of Kim [22], which is used in two-machine flowshop total tardiness problem.

Proposition 7. In a two-stage flowshop, the completion time of the j th job in an arbitrary schedule is no less than $\max\{\max\{P_{ja}, P_{jb}\} + P_{[1]l}, \max\{p_{[1]a}, p_{[1]b}\} + P_{jl}\}$, where $p_{[1]k} = \min_i\{p_{ik}\}$ for $k = a, b, l$ and P_{jk} is the sum of j shortest

processing times on machine k among all jobs; that is, $p_{[1]k} \leq p_{[2]k} \leq \dots \leq p_{[n]k}$ and $P_{jk} = \sum_{i=1}^j p_{[i]k}$.

Proof. P_{ja} and P_{jb} are lower bounds on the time needed to process j jobs on machines a and b , respectively. Thus, $\max\{P_{ja}, P_{jb}\}$ is a lower bound on the time required to process j jobs in the first stage. Thus, the completion time of the j th job is no less than the sum of $\max\{P_{ja}, P_{jb}\}$ and the minimum processing time among all jobs in the second stage, $p_{[1]l}$. Similarly, it is no less than $\max\{p_{[1]a}, p_{[1]b}\} + P_{jl}$. $\square$

From Proposition 7, a lower bound on the total tardiness of jobs that are not included in a partial schedule, σ (or to be scheduled after σ), can be computed. Then, the lower bound is computed as $\sum_{j=1}^{n^u(\sigma)} \max\{0, \max\{\max\{P_{ja}, P_{jb}\} + p_{[1]l}, \max\{p_{[1]a}, p_{[1]b}\} + P_{jl}\} - d_{[j]}\}$.

The lower bound can be improved by considering the completion times of the last jobs on the machines in σ , that is, $\Gamma_a(\sigma)$, $\Gamma_b(\sigma)$, and $\Gamma_l(\sigma)$. Meanwhile, if machine l in the second stage is working when the processing in the first stage of the next job after σ is completed, the last operation of the job can be started only after machine l becomes available. Then, $\Gamma_l(\sigma) + P_{jl}$ can also be used to compute a lower bound. Thus, the improved lower bound is used in the B&B algorithm and it is computed as follows:

$$LB_1 = \sum_{j=1}^{n^u(\sigma)} \max\{0, \max\{\max\{\Gamma_a(\sigma) + P_{ja}, \Gamma_b(\sigma) + P_{jb}\} + p_{[1]l}, \max\{\max\{\Gamma_a(\sigma) + p_{[1]a}, \Gamma_b(\sigma) + p_{[1]b}\}, \Gamma_l(\sigma) + P_{jl}\} - d_{[j]}\}\}. \quad (21)$$

The second bound (LB_2) is obtained by relaxing the problem considered in this study into a two-machine flowshop tardiness problem in which the two machines in the first stage are considered as a single machine. Here, let G_1 be the minimum value of the completion times of the two machines in the first stage and G_2 the completion time of the machine in the second stage. The following procedure is used to obtain LB_2 .

Procedure 2 (computation of LB_2).

Step 0. Set $j = 1$, $G_1 = \min\{\Gamma_a(\sigma), \Gamma_b(\sigma)\}$, $G_2 = \Gamma_l(\sigma)$, and $LB_2 = 0$.

Step 1. Let $LB_2 \leftarrow LB_2 + \max\{0, \max\{G_1 + \max\{p_{[j]a}, p_{[j]b}\}, G_2\} + p_{[j]l} - d_{[j]}\}$, $G_2 \leftarrow \max\{G_1 + \max\{p_{[j]a}, p_{[j]b}\}, G_2\} + p_{[j]l}$, and $G_1 \leftarrow G_1 + \min\{p_{[j]a}, p_{[j]b}\}$.

Step 2. Let $j \leftarrow j + 1$. If $j > n^u(\sigma)$, terminate; otherwise, go to Step 1.

The maximum value between the two lower bounds, $\max\{LB_1, LB_2\}$, can be used as the lower bound for jobs not in the partial schedule in the proposed B&B algorithm. Then,

a lower bound for the partial schedule corresponding to each node in the B&B tree can be obtained by adding this lower bound to the total tardiness of the jobs included in the partial schedule.

5. Computational Experiments

In order to evaluate the performance of the branch and bound (B&B) algorithm proposed in this paper, a series of computational tests were performed on randomly generated problem instances. In the instances, five levels (8, 12, 16, 20, and 24) for the number of jobs were considered. Processing times and due dates were generated according to the method used in Kim [23]. The processing time of an operation was generated from $DU(1, 100)$, where $DU(a, b)$ denotes the discrete uniform distribution with a range $[a, b]$. Then, the due dates of the jobs were generated from $DU(P^*(1 - T - R/2), P^*(1 - T + R/2))$, where P^* is the time at which all jobs are completed, and T and R are the tardiness factor and relative range of due dates parameters, respectively. Here, P^* is estimated using the lower bound on the makespan; that is, $P^* = \max\{\max\{\sum_{i=1}^n p_{ia}, \sum_{i=1}^n p_{ib}\} + \min_{i \in J}\{p_{il}\}, \sum_{i=1}^n p_{il}\}$, and three levels for each of T and R were used for generation;

that is, $T = (0.1, 0.3, \text{ and } 0.5)$ and $R = (0.8, 1.3, \text{ and } 1.8)$. If the resulting due dates were less than 0, they were reset to 0. For the experiments, 450 instances were generated, 10 instances for each combination of the number of jobs, T , and R . The algorithm was coded in Java, and the experiments were performed on a personal computer with a quad-core processor operating at 3.2 GHz and 4 GB RAM. In order to avoid excessive computation times for the experiments, a limit of 3600 seconds was set for the CPU time for each instance.

First, the performance of the B&B algorithm proposed in this study was tested through comparisons with that of the mixed integer program (MIP) given in Section 2 in terms of computation times and the numbers of instances that were solved within the time limit. Note that a commercial integer programming solver (CPLEX 12.5) was used to solve the MIP. The results of the test are given in Table 1. As seen in the table, the CPU times required to solve instances with up to 12 jobs did not differ significantly. However, for instances with 16 to 24 jobs, the B&B algorithm required a significantly shorter time than CPLEX. In addition, the B&B algorithm found optimal solutions for instances with up to 20 jobs, whereas CPLEX did not find optimal solutions for 10 instances with 20 jobs. For instances with 24 jobs, the B&B algorithm found more optimal solutions than CPLEX. Generally, the outperformance of the B&B algorithm over CPLEX became more significant as the problem size increased. Note that the test results of the proposed B&B algorithm are used in the next tests.

As seen from Table 1, the performance of the B&B algorithm was affected by the tightness of due dates or the tardiness factor (T) and relative range of due dates (R). In order to see the effect, an analysis of variance (ANOVA) was performed for the instances with up to 20 jobs because several instances with 24 jobs were not solved within the time limit. From the results given in Table 2, the performance of the B&B algorithm was significantly affected by the values of T and R , but not by their interaction ($T \times R$). As seen in Table 1, the instances with a small T were solved within shorter computation times. Note that when T is smaller, the due dates are looser in general and, in this case, the total tardiness of jobs in an optimal solution was small (or 0 in several instances). In cases with looser due dates, the heuristic algorithm used in the root node of the B&B tree may find a tighter initial upper bound; hence, many partial schedules could be fathomed in the earlier stages of the B&B algorithm. Furthermore, the algorithm worked better when R was large, possibly because the due dates of jobs were widely dispersed, which resulted in the dominance properties used in the B&B algorithm being applied more often. In contrast, when T was large and R was small, the performance of the algorithm was not good.

In order to verify the effectiveness of the lower bounds used in the B&B algorithm, two B&B algorithms were compared; the BB_{LB+} with the lower bounds and the BB_{LB-} without the lower bounds. All dominance properties given in Section 3 were implemented in both algorithms. In order to avoid excessive computation times, instances with up to 20 jobs were used. The results are summarized in Table 3. As seen from the table, instances with up to 12 jobs were solved within 1 second on average, even though the lower bounds

TABLE 1: Results of the test on the B&B algorithm and CPLEX.

n	(T, R)	B&B			CPLEX		
		ACPUT [†]	MCPUT [‡]	NI [#]	ACPUT	MCPUT	NI
8	(0.1, 0.8)	<0.01	<0.01	10	0.04	0.03	10
	(0.1, 1.3)	<0.01	<0.01	10	0.05	0.05	10
	(0.1, 1.8)	<0.01	<0.01	10	0.06	0.06	10
	(0.3, 0.8)	<0.01	<0.01	10	0.05	0.03	10
	(0.3, 1.3)	<0.01	<0.01	10	0.04	0.04	10
	(0.3, 1.8)	<0.01	<0.01	10	0.07	0.06	10
	(0.5, 0.8)	<0.01	<0.01	10	0.05	0.05	10
	(0.5, 1.3)	<0.01	<0.01	10	0.05	0.05	10
12	(0.5, 1.8)	<0.01	<0.01	10	0.07	0.06	10
	(0.1, 0.8)	0.04	0.00	10	0.13	0.06	10
	(0.1, 1.3)	0.00	0.00	10	0.11	0.05	10
	(0.1, 1.8)	0.02	0.00	10	0.55	0.42	10
	(0.3, 0.8)	0.03	0.02	10	0.24	0.09	10
	(0.3, 1.3)	0.03	0.02	10	0.71	0.40	10
	(0.3, 1.8)	0.02	0.02	10	1.14	0.72	10
	(0.5, 0.8)	0.05	0.04	10	0.26	0.17	10
16	(0.5, 1.3)	0.05	0.02	10	0.97	0.35	10
	(0.5, 1.8)	0.06	0.02	10	0.66	0.65	10
	(0.1, 0.8)	0.00	0.00	10	0.07	0.06	10
	(0.1, 1.3)	0.06	0.00	10	0.25	0.08	10
	(0.1, 1.8)	0.12	0.09	10	4.57	1.15	10
	(0.3, 0.8)	1.21	0.91	10	0.62	0.65	10
	(0.3, 1.3)	0.10	0.06	10	4.05	1.53	10
	(0.3, 1.8)	0.85	0.17	10	77.79	10.82	10
20	(0.5, 0.8)	2.10	1.20	10	4.10	3.74	10
	(0.5, 1.3)	1.26	1.07	10	8.80	4.79	10
	(0.5, 1.8)	1.09	0.87	10	208.34	80.99	10
	(0.1, 0.8)	0.00	0.00	10	0.12	0.11	10
	(0.1, 1.3)	0.44	0.00	10	8.82	0.11	10
	(0.1, 1.8)	0.85	0.13	10	32.97	2.66	10
	(0.3, 0.8)	34.42	24.09	10	18.43	1.88	10
	(0.3, 1.3)	7.68	0.66	10	758.16	10.90	8
24	(0.3, 1.8)	14.57	6.41	10	1804.62	1817.27	5
	(0.5, 0.8)	151.51	71.92	10	239.56	40.82	10
	(0.5, 1.3)	61.48	11.08	10	386.61	180.46	10
	(0.5, 1.8)	32.53	7.96	10	1230.12	275.37	7
	(0.1, 0.8)	0.00	0.00	10	0.20	0.19	10
	(0.1, 1.3)	7.15	0.00	10	18.10	0.40	10
	(0.1, 1.8)	5.71	1.02	10	514.38	96.66	9
	(0.3, 0.8)	292.95	81.57	10	3.34	2.23	10
	(0.3, 1.3)	692.23	9.70	9	1248.93	255.20	7
	(0.3, 1.8)	304.40	34.13	10	2905.16	3600.00	2
	(0.5, 0.8)	1210.80	431.44	8	1209.62	135.64	7
	(0.5, 1.3)	946.30	166.13	8	2129.49	3055.50	5
	(0.5, 1.8)	669.19	161.96	9	2634.28	3600.00	3

[†] Average CPU time (3600 seconds was used as the CPU time for the instances that were not solved to optimality in 3600 seconds).

[‡] Median CPU time.

[#] Number of instances (out of 10 instances) that were solved within 3600 seconds.

were not used. However, as the number of jobs increased, BB_{LB+} outperformed BB_{LB-} , and the outperformance became clearer as the number of jobs increased. In addition, BB_{LB+} solved all instances, whereas BB_{LB-} did not solve 36 instances with 20 jobs within 3600 seconds. This indicates that using

TABLE 2: ANOVA for the effect of due date tightness on CPU time of the B&B algorithm.

Sources	SS	DF	MS	<i>F</i>	<i>p</i> value
<i>T</i>	28237.9	2	14119.0	9.3	0.000
<i>R</i>	9393.6	2	4696.8	3.1	0.047
Interaction (<i>T</i> × <i>R</i>)	11227.5	4	2806.9	1.8	0.119
Error	533384.4	351	1519.6		
Total	582243.4	359			

Significance level = 0.05.

TABLE 3: Effectiveness of the lower bounds.

<i>n</i>	ACPUT [†]		ANNG [‡]		NI [#]	
	BB _{LB+}	BB _{LB-}	BB _{LB+}	BB _{LB-}	BB _{LB+}	BB _{LB-}
8	<0.01	0.01	68	542	90	90
12	0.03	0.58	1008	26498	90	90
16	0.75	111.93	13855	3326995	90	90
20	33.72	1967.48	423917	39077867	90	54

[†] Average CPU time.[‡] Average number of nodes generated in the branch and bound procedure.[#] Number of instances (out of 90 instances) that were solved within 3600 seconds.

the lower bounds significantly reduced the computation time of the B&B algorithm and fathomed a larger number of nodes generated in the B&B procedure. Furthermore, the effectiveness of the lower bounds became more significant as the problem size increased.

Next, in order to verify the effectiveness of the dominance properties, seven B&B algorithms were tested: BB, which is the B&B algorithm with all dominance properties; BB_{D_x}, where $x = 1, 2, \dots, 5$, which denotes the B&B algorithm with all dominance properties except for that given in Proposition x ; and BB_{D₋}, which is the B&B algorithm with no dominance property. In these algorithms, the lower bounds were used. Table 4 presents the results of these tests. As seen from the table, using the dominance properties, the computation time and the number of generated nodes were reduced considerably. In particular, when $n = 20$, BB_{D₋} did not solve 19 instances within 3600 seconds. This shows that a large portion of the nodes in the B&B tree could be fathomed; hence, the use of dominance properties assisted in enhancing the performance of the B&B algorithm, even though checking those in each node required the computation times. On the other hand, BB_{D₅} required longer computation time than the other BB_{D_x} algorithms. This indicates that the dominance property given in Proposition 5 can be applied more often.

Finally, the heuristic algorithm (mNEH-PI) used in the B&B algorithm was tested in order to examine its effectiveness and performance. As in the above tests, a B&B algorithm with the heuristic was compared with a B&B algorithm without the heuristic. All lower bounds and dominance properties were used in both algorithms. The results of this test are summarized in Table 5. In the table, BB_{H+} and BB_{H-} denote the B&B algorithms with and without mNEH-PI, respectively. As shown in Table 5, the performance of BB_{H+} was better than that of BB_{H-}. That is, when the heuristic was

embedded in the B&B algorithm, the computation time and the number of nodes were reduced significantly. On the other hand, the percentage gap of the solutions of mNEH-PI from the optimal solutions was 2.09% on average. Note that the computation time required for mNEH-PI to solve an instance was less than 0.01 seconds regardless of the instance size and parameter values.

6. Concluding Remarks

This paper considered a two-stage assembly-type flowshop scheduling problem with the objective of minimizing the total tardiness, in which the first stage consists of two independent machines and the second stage consists of a single machine. It is assumed that the processing in the second stage can start only after the processing in the first stage is completed. The dominance properties and lower bounds on the tardiness of jobs associated with a partial schedule were developed, and a branch and bound (B&B) algorithm using these was presented. The results for randomly generated instances showed that the proposed B&B algorithm could find optimal solutions for instances with up to 24 jobs within a reasonable amount of computation time, except for a few cases, and the proposed algorithm worked better than CPLEX.

This research can be extended in several ways. In this study, a simple assembly-type flowshop case was considered, but problems with various characteristics could have been considered. For example, there is a case in which machines in the first stage can produce several types of components; that is, two components of a job can be fabricated on the same machine. Also, a case might include time constraints between the two stages; that is, the (assembly) operation of each job in the second stage should be started within a certain time period after the operations of the job in the first stage are completed. On the other hand, as mentioned earlier, this study has been the first attempt to develop an exact algorithm for the problem considered in this paper. Thus, better dominance properties and tighter bounds might need to be developed, or other exact algorithms to solve larger problems in which there are two stages and more than two machines in each stage might need to be developed.

Notation

The Notation Used in the Formulation

J :	Set of jobs, $J = \{1, \dots, n\}$
i, j :	Job indices
a, b, l :	Machine indices
O_{ia} (O_{ib}):	Operation of job i to be processed on machine a (b) in the first stage
O_{il} :	Operation of job i to be processed on machine l in the second stage
p_{ik} :	Processing time of operation O_{ik} , $k = a, b, l$
d_i :	Due date of job i
$[r]$:	Index for the r th job in a sequence
x_{ir} :	A binary variable that is equal to 1 if job i is scheduled in the r th position in a sequence, and 0 otherwise

TABLE 4: Effectiveness of the dominance properties.

n	$n = 8$		$n = 12$		$n = 16$		$n = 20$	
	ACPUT [†]	ANNG [‡]	ACPUT	ANNG	ACPUT	ANNG	ACPUT	ANNG
BB	<0.01	68	0.03	1008	0.75	13855	33.72	423917
BB _{D1}	<0.01	91	0.07	2141	1.96	36077	95.20	1239968
BB _{D2}	<0.01	78	0.05	1506	1.35	25266	69.23	852062
BB _{D3}	<0.01	73	0.05	1561	1.65	31665	84.22	1085409
BB _{D4}	<0.01	71	0.04	1309	1.13	21147	53.62	684952
BB _{D5}	<0.01	75	0.10	3222	4.64	104141	252.84	3666210
BB _{D-}	<0.01	192	0.51	34250	82.38	4176444	976.07	26509066

^{†,‡}See footnotes in Table 3.

TABLE 5: Effectiveness and performance of the heuristic algorithm.

n	BB _{H+}		BB _{H-}		mNEH-PI
	ACPUT [†]	ANNG [‡]	ACPUT	ANNG	APG [§]
8	<0.01	68	<0.01	106	1.44
12	0.03	1008	0.05	1565	2.67
16	0.75	13855	1.71	35412	1.98
20	33.72	423917	64.04	879744	2.25

^{†,‡}See footnotes in Table 3.

[§]Average percentage gap between heuristic solutions and optimal solutions.

$s_{[r]k}$: Starting time of the r th job on machine k ,
 $k = a, b, l$

$C_{[r]}$: Completion time of the r th job in a sequence

$T_{[r]}$: Tardiness of the r th job in a sequence,
 $T_{[r]} = \max\{C_{[r]} - \sum_{i=1}^n d_i \cdot x_{ir}\}.$

The Notation Used in the Statements and Proofs of the Properties and for the Description of the Algorithms

σ : A (partial) schedule

$s_{ik}(\sigma)$: Start time of operation O_{ik} in σ , $k = a, b, l$

$c_{ik}(\sigma)$: Completion time of operation O_{ik} in σ ,
 $k = a, b, l$

$C_i(\sigma)$: Completion time of job i in σ ; that is,
 $C_i(\sigma) = c_{il}(\sigma)$

$T_i(\sigma)$: Tardiness of job i in σ ; that is,
 $T_i(\sigma) = \max\{C_i(\sigma) - d_i, 0\}$

$\Gamma_k(\sigma)$: Completion time of the last job on machine k in σ , $k = a, b, l$

$S(\sigma)$: Set of jobs scheduled (or included) in σ

$n^s(\sigma)$: Number of jobs scheduled in σ , that is,
 $|S(\sigma)|$

$n^u(\sigma)$: Number of jobs that are not included in σ ,
that is, $n - |S(\sigma)|$

$S_{ij}(\sigma)$: Set of jobs scheduled from job i to j in σ .

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

- [1] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.
- [2] C. Koulamas, "The total tardiness problem: review and extensions," *Operations Research*, vol. 42, no. 6, pp. 1025–1041, 1994.
- [3] 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.
- [4] 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.
- [5] C. N. Potts, S. V. Sevast'yanov, 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.
- [6] 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.
- [7] M. Haouari and T. Daouas, "Optimal scheduling of the 3-machine assembly-type flow shop," *RAIRO Operations Research*, vol. 33, no. 4, pp. 439–445, 1999.
- [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] C. Koulamas and G. J. Kyparisis, "The three-stage assembly flowshop scheduling problem," *Computers and Operations Research*, vol. 28, no. 7, pp. 689–704, 2001.
- [10] C.-S. Sung and J.-H. Juhn, "Makespan minimization for a 2-stage assembly scheduling problem subject to component available time constraint," *International Journal of Production Economics*, vol. 119, no. 2, pp. 392–401, 2009.
- [11] 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.
- [12] F. S. Al-Anzi and A. Allahverdi, "A hybrid tabu search heuristic for the two-stage assembly scheduling problem," *International Journal of Operations Research*, vol. 3, no. 2, pp. 109–119, 2006.
- [13] 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.

- [14] 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 and Operations Research*, vol. 33, no. 4, pp. 1056–1080, 2006.
- [15] 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.
- [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] J. Schaller, "Note on minimizing total tardiness in a two-machine flowshop," *Computers and Operations Research*, vol. 32, no. 12, pp. 3273–3281, 2005.
- [18] C. N. Potts and L. N. Van-Wassenhove, "Branch and bound algorithm for the total weighted tardiness problem," *Operations Research*, vol. 33, no. 2, pp. 363–377, 1985.
- [19] 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.
- [20] C. Chu, "A branch-and-bound algorithm to minimize total tardiness with different release dates," *Naval Research Logistics*, vol. 39, no. 2, pp. 265–283, 1992.
- [21] S. Kondakci, Ö. Kirca, and M. Azizoglu, "An efficient algorithm for the single machine tardiness problem," *International Journal of Production Economics*, vol. 36, no. 2, pp. 213–219, 1994.
- [22] Y.-D. Kim, "A new branch and bound algorithm for minimizing mean tardiness in two-machine flowshops," *Computers and Operations Research*, vol. 20, no. 4, pp. 391–401, 1993.
- [23] Y.-D. Kim, "Minimizing total tardiness in permutation flowshops," *European Journal of Operational Research*, vol. 85, no. 3, pp. 541–555, 1995.

Hindawi

Submit your manuscripts at
<http://www.hindawi.com>