Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 6982631, 17 pages
https://doi.org/10.1155/2018/6982631

Research Article

Two-Stage Assembly Scheduling with Batch Setup Times, Time-Dependent Deterioration, and Preventive Maintenance Activities Using Meta-Heuristic Algorithms

Sunwoong Jung,[1] Young-Bin Woo,[1] Shiegheun Koh,[2] and Byung Soo Kim [1]

[1]*Department of Industrial and Management Engineering, Incheon National University, 119 Academy-ro, Songdo-dong, Yeonsu-gu, Incheon 406-772, Republic of Korea*
[2]*Department of Systems Management and Engineering, Pukyong National University, 45 Yongsoru, Namgu, Busan 608-737, Republic of Korea*

Correspondence should be addressed to Byung Soo Kim; bskim@incheon.ac.kr

Received 6 July 2018; Accepted 3 December 2018; Published 24 December 2018

Academic Editor: Mauro Gaggero

This article considers a two-stage assembly scheduling problem (TSASP) with batch setup times, time-dependent deterioration, and preventive maintenance activities (PMAs). The objective of this problem is to simultaneously determine the optimal component-manufacturing sequence (CMS), product-assembly sequence (PAS), number of setups, and number and position of PMAs (PPMA). First, to determine the optimal solution, a novel mixed integer linear programming model (MILP) for the proposed problem is derived. Then, a standard genetic algorithm (SGA), hybrid genetic algorithm (HGA), standard harmony search (SHS), hybrid harmony search (HHS), and harmony-search-based evolutionary algorithm (HSEA) were proposed owing to the intractability of the optimal solution for large-scale problems. SGA and SHS provide a chromosome to represent a complete solution including three decisions (CMS, PAS, and PPMA). HGA, HHS, and HSEA provide a chromosome to represent a partial solution including PAS. CMS and PPMA are found by an effective local search heuristic based on the partial solution. A computational experiment is then conducted to evaluate the impacts of the factors on the performance of the proposed algorithms.

1. Introduction

The two-stage assembly scheduling problem (TSASP) is one of the most widely studied scheduling problems in the literature and has many applications in the industry. This problem has been determined under many different settings reflecting a wide range of application environments [1–3]. However, most previous studies for TSASP did not consider the delay of tasks due to deterioration and maintenance activities. In this article, we consider TSASP with batch setup times, time-dependent deterioration, and preventive maintenance activity (PMA). In the first stage, a single machining machine produces components to assemble products. During the machining process, a batch setup time occurs whenever a new component is processed, or the component type is switched on the machining machine. Each component has a different deterioration rate and the deterioration is continuously accumulated during the production of the component. The PMA restores the deteriorated processing times of tasks to the original processing time by the cleaning or maintenance process on the machines. In the second stage, a single assembly machine can start assembling several required components for an ordered product when the components are available from the first stage. The target manufacturing system is described in Figure 1. In the figure, I_m and J_n represent m types of products and n types of components, respectively.

TSASP, like the other scheduling problems, has widely received attention in the literature [4–7]. Due to complex schemes of the problem, many studies proposed metaheuristic algorithms. Allahverdi and Al-Anzi [3] proposed a particle-swarm optimization algorithm (PSO), tabu-search algorithm (TS), and local search heuristic in a two-stage distributed database application. Their computational analysis

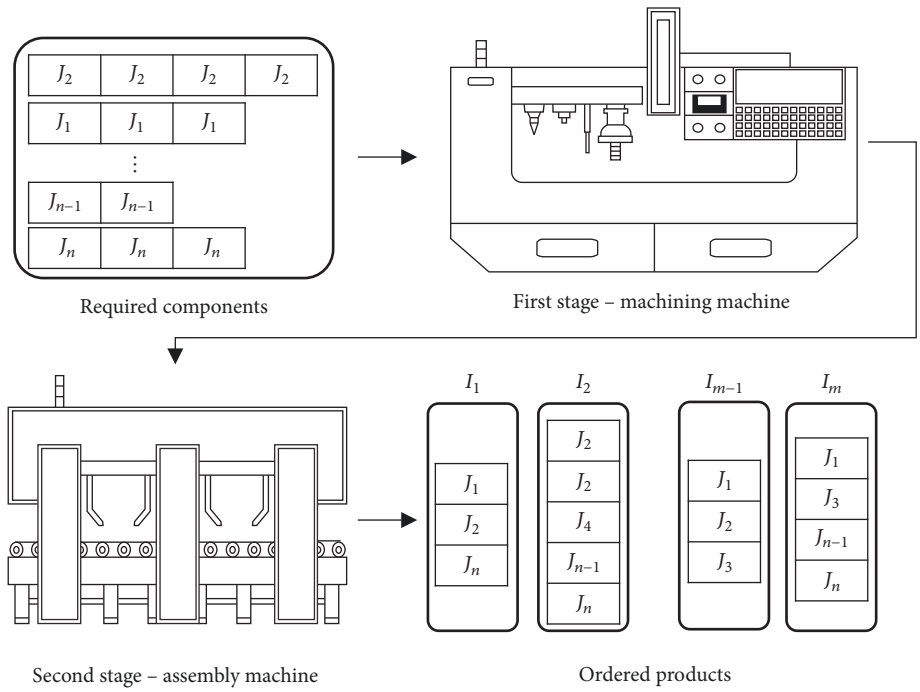

Figure 1: Overall structure of targeting manufacturing system.

indicates that the difference between the average errors of PSO is the best. Al-Anzi and Allahverdi [8] proposed a self-adaptive differential evolution heuristic (SADEH) for TSASP with respect to maximum lateness criterion where setup times are treated as separate from processing times. Yan et al. [9] proposed a hybrid variable neighborhood search (VNS) algorithm to minimize the weighed sum of the maximum makespan, earliness, and lateness on parallel-manufacturing machines and a single assembly machine in TSASP. Komaki et al. [10] proposed an artificial immune system algorithm (AIS) to solve a two-stage hybrid flow shop followed by a single assembly machine to minimize the makespan and Deng et al. [11] proposed a competitive memetic algorithm (CMA) to minimize the makespan in a parallel-manufacturing machine and a single assembly machine in TSASP. Komaki and Kayvanfar [12] proposed a grey wolf optimizer algorithm to solve the TSASP with release time in a parallel-manufacturing machine and a single assembly machine. Jung et al. [13] proposed two hybrid genetic algorithms (HGAs) for two-stage assembly scheduling problem (TSASP) for processing products with dynamic-component sizes and a setup time. They had the same two-stage assembly structure with our study. The main differences are that they considered no job deterioration and preventive maintenance activities in the manufacturing stage.

Meanwhile, many scheduling studies related to single or parallel machines considered not only a simple manufacturing environment but also various manufacturing environments such as situations involving deterioration and preventive maintenance [14–18]. According to Gupta and Gupta [19] and Browne and Yechiali [20], the processing times of tasks are not constant but increase over time depending on the sequence of the tasks or their starting times in many practical environments. This phenomenon is generally known as the *deterioration* of resources in the scheduling problem. As mentioned earlier, various types of deterioration occur in practical environments owing to workers' fatigue, mal-position of tools, and scraps of operations. Therefore, scheduling problems with deterioration are universal in a number of industrial applications. Meanwhile, the deteriorated processing times of tasks can be restored to the original processing times by the cleaning or maintenance process on the machines. An activity that changes the production efficiency of a machine is called a *preventive maintenance activity* (PMA) [21]. According to these previous researches, many researches were conducted for scheduling problem with deterioration and PMA. Mosheiov and Sarig [14] considered a single-machine scheduling and due-window assignment problem with PMA. The objective of this problem is to schedule the jobs, due-window and PMA, so as to minimize the total cost consisting of earliness, tardiness and due-window starting time and size. Cheng et al. [15] considered a two-agent scheduling problem in which they assumed that, given a schedule, the actual processing time of a task of the first agent is a function of position-based learning, whereas the actual processing time of a task of the second agent is a function of position-based deterioration for minimizing the makespan. Lee et al. [16] considered the total earliness and tardiness penalty when scheduling simultaneously with the deterioration effects and maintenance activities on an unrelated parallel machine setting. Yang [17] considered the unrelated parallel machines with simultaneous considerations of the deterioration effects and multimaintenance activities for minimizing the makespan. They examined two

models of scheduling with the deterioration effect, namely, the job-dependent and position-dependent deterioration model and the time-dependent deterioration model. Ji et al. [18] considered a single-machine scheduling with a common due-window and a deteriorating PMA. They assumed that the processing time of a task is a function of the amount of a resource allocated to it, its position in the processing sequence, and its aging effect. Cheng [22] introduced a single-machine two-agent scheduling problem with a truncation learning effect. The objective is to find an optimal schedule to minimize the makespan. He used a branch-and-bound algorithm and three heuristic-based genetic algorithms to solve the problem. Wu et al. [23] proposed a branch-and-bound algorithm and some ant colony algorithms for two-agent single-machine scheduling problem involving with the sum-of-processing-times-based learning and deteriorating effects. Wu et al. [24] proposed to integrate the two-agent and time-dependent processing times scheduling problem to minimize the total weighted number of tardy jobs of the first agent with the restriction that the maximum lateness of a job of the second agent is allowed to have an upper bound. They also proposed TS for searching near-optimal solutions. Ruiz-Torres et al. [25] proposed a set of list scheduling algorithm and simulated annealing algorithm (SA) for unrelated parallel machines with deterioration and multimaintenance activities. The results of this paper indicate that the SA is capable of producing high quality solutions for a wide range of instances. Joo and Kim [26] and Chung and Kim [27] considered a single-machine scheduling problem with various deteriorations and multiple PMAs for minimizing the makespan. To solve the problem, they proposed a mathematical model and hybrid genetic algorithms.

To the best of our knowledge, no available effective metaheuristic algorithms are capable of generating a near-optimal solution while the TSASP simultaneously considering batch setup times, time-dependent deterioration, and PMAs. The remainder of this article is organized as follows. In Section 2, we describe a mathematical model of this problem. Section 3 describes five metaheuristic algorithms that are proposed in this article. First, the standard genetic algorithm (SGA) and standard harmony search (SHS) are proposed with a complete solution indicating component-manufacturing sequence (CMS), product-assembly sequence (PAS), and position of PMAs (PPMA). Second, hybrid genetic algorithm (HGA) and hybrid harmony search (HHS) with local search heuristic are proposed with a partial solution indicating PAS. The remaining solution of HGA and HHS indicates CMS and PPMA are provided by local search heuristic. Finally, a harmony-search-based evolutionary algorithm (HSEA) that combines evolutionary algorithm, harmony search, and local search heuristic is proposed with a partial solution (CMS and PPMA). In Section 4, a computational experiment is then conducted to evaluate the impacts of the factors on the performance of the proposed algorithms using randomly generated instances. Finally, the conclusion and the future studies are discussed in Section 5.

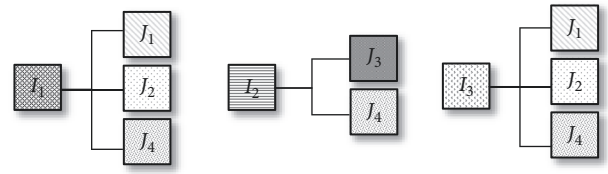

Figure 2: Example for representing dynamic-component size.

2. Mixed Integer Linear Programming (MILP) Model

In this section, a novel MILP model is developed to minimize the makespan for the TSASP with batch setup times, deterioration, and PMAs. The problem statement of this problem has been applied the *dynamic-component size*. This means that the kinds and number of components are different for each product. This is depicted in detail in Figure 2.

According to Figure 2, one unit of component 1, one unit of component 2, and one unit of component 4 are required to assemble product 1. One unit of component 3 and one unit of component 4 are required to assemble product 2. Finally, one unit of component 1, one unit of component 2, and one unit of component 4 are required to assemble product 3. As such, various combinations of components for product types are required. Thus, the size of the components is dynamically changed according to the associated product, which is called a *dynamic-component size*. The TSASP has a realistic production process with the manufacturing stage and assembly stage. In the first stage, a single machining machine produces components to assemble products. During the machining process, a batch setup time occurs whenever the machining machine produces a different component of a new component.

Each component has a different deterioration rate and the deterioration is continuously accumulated during the producing the component. The PMA restores the deterioration when it is most appropriate in order to minimize the makespan. In the second stage, a single assembly machine starts to assemble products whenever the required components of the corresponding the product are available from the first stage.

The notations, decision variables, basic assumptions, and a mathematical model for the proposed problem are as follows:

(i) Sets

I: product types

J: component types

K: indices of product sequence to be assembled

L: indices of component sequence to be manufactured

(ii) Parameters

μ: batch setup time in the machining machine

α_j: manufacturing time for $j \in J$

β_i: assembly time for $i \in I$

ρ_{ij}: the required number of $c \in C$ for $i \in I$

γ_j: the deterioration rate for $j \in J$

σ: PMA time

M: big number

(iii) Decision Variables

loc_{kjl}: the required number of $j \in J$ for assembling to $k \in K$ at $l \in L$

cnc_{jl}: the cumulated number of $j \in J$ at $l \in L$

cnp_{jk}: the cumulated number of used $j \in J$ for assembling to $k \in K$

ctc_l: completion time of component at $l \in L$

ctp_k: completion time of any product assigned to $k \in K$

avt_{kl}: available time for assembling any product assigned to $k \in K$ at $l \in L$

cts_l: cumulative time of setup at $l \in L$

d_k: temporary variable for linearization

e_k: temporary variable for linearization

C_{max}: makespan for the TSASP

$v_{jl} = \{1, \text{If } j \in J \text{ is scheduled at } l \in L; 0, \text{Otherwise}\}$

$w_{ik} = \{1, \text{If } i \in I \text{ is scheduled at } k \in \text{K}; 0, \text{Otherwise}\}$

$x_l = \{1, \text{If } l \in L \text{ and } l-1 \in L \text{ are the same}; 0, \text{Otherwise}\}$

$y_{kl} = \{1, \text{If there is at least one required component for assembling any product assigned to } k \in K \text{ at } l \in L; 0, \text{Otherwise}\}$

$z_l = \{1, \text{If there is a PMA to } l \in L; 0, \text{Otherwise}\}$

$f_{jl} \in B$: temporary variable for linearization

$g_{kl} \in B$: temporary variable for linearization

(iv) Basic Assumptions

(1) All components are available at time zero

(2) The machining machine produces at most one component

(3) The deterioration makes only component in machining machine

(4) The assembly machine assembles at most one assembly operation

(5) All batch setup times are identical

(6) All PMA times are identical

(7) The machining and assembly machines are continuously available

(8) No preemptions are allowed in the machining and assembly operations

(9) There are unlimited buffers between the machining machine and assembly machine

(v) Mixed Integer Linear Programming (MILP) Model

$$\min C_{max} \tag{1}$$

The objective function in (1) is to minimize the makespan of the TSASP.

$$\sum_{c \in C} v_{jl} = 1, \quad \forall l \in L \tag{2}$$

$$\sum_{i \in I} w_{ik} = 1, \quad \forall k \in K \tag{3}$$

$$\sum_{k \in K} w_{ik} = 1, \quad \forall i \in I \tag{4}$$

The constraints in (2) ensure that the machining machine must produce only one component at each $l \in L$. The constraints in (3) and (4) ensure that assembly machine must produce only one product at each $k \in K$.

$$\sum_{l \in L} v_{jl} = \sum_{i \in I} \rho_{ij}, \quad \forall j \in J \tag{5}$$

$$x_l = \max_{j \in J} \left(v_{jl-1} + v_{jl}, 1 \right) - 1, \quad \forall l \in L,\ l > 1\ \forall j \in J \tag{6}$$

The constraints in (5) ensure that the amount in which each component is produced corresponded to the number of the components required to assemble all the products. The constraints in (6) define the batch relationship determining that two consecutive components are the same.

$$ctc_l \geq \sigma \bullet z_l + \mu + \sum_{j \in J} \alpha_j \bullet v_{jl}, \quad l = 1 \tag{7}$$

$$ctc_l \geq ctc_{l-1} + \gamma_j \bullet \left(ctc_{l-1} - cts_{l-1} \right) + \sigma \bullet z_l + \mu \bullet \left(1 - x_l\right) + \alpha_j \bullet v_{jl} - M \bullet \left(1 - v_{jl}\right), \quad \forall l \in L,\ l > 1,\ \forall j \in J \tag{8}$$

The constraints in (7) and (8) define the completion time of each component. The completion time for first $l \in L$ is calculated by adding the batch setup time, the machining time for the component assigned to first $l \in L$, and the PMA time that determines whether PMA is conducted or not through z_l as shown in (7). After that, the constraints in (8) calculate the batch setup time according to different component added to $l \in L$ compared to the component at $l - 1 \in L$ and add a deterioration time of the current component and the PMA time is determined through whether or not the PMA is conducted.

$$ctp_k \geq \min_{l \in L} \left(avt_{kl} \right) + \sum_{i \in I} \beta_i \bullet w_{ik}, \quad k = 1 \tag{9}$$

$$ctp_k \geq \max \left(ctp_{k-1}, \min_{l \in L} \left(avt_{kl} \right) \right) + \sum_{i \in I} \beta_i \bullet w_{ik}, \quad \forall k \in k,\ k > 1 \tag{10}$$

The constraints in (9) and (10) define the completion time of each product. The completion time for first $k \in K$ is calculated by adding the minimum available assembly time for a product and the assembly time for the product as shown in (9). Following this, the constraints in (10) consider the completion time of the previous $k \in K$ due to the sufficient condition for operating the assembly machine.

$$cnc_{jl} = \sum_{a=1}^{l} v_{ja}, \quad \forall j \in J, \ \forall l \in L \tag{11}$$

$$cnp_{jk} = \sum_{i \in I} \left(\rho_{ij} \bullet \sum_{a=1}^{k} w_{ia} \right), \quad \forall j \in J, \ \forall k \in K \tag{12}$$

The constraints in (11) and (12) define the number of manufactured components at each $l \in L$ and the cumulated number of required components, respectively.

$$\min \left(M \bullet z_l + \mu, ctc_l \right) \geq cts_l, \quad l = 1 \tag{13}$$

$$\begin{aligned} &\min \left(M \bullet z_l + cts_{l-1} + \mu \bullet \left(1 - x_l\right), ctc_l - cts_{l-1} \right) \\ &\quad \geq cts_l, \quad \forall l \in L, \ l > 1 \end{aligned} \tag{14}$$

$$\begin{aligned} loc_{kjl} = \max \left(cnp_{jk} - cnc_{jl}, 0 \right), \\ \forall j \in J, \ \forall k \in k, \ \forall l \in L \end{aligned} \tag{15}$$

The constraints in (13) and (14) define the cumulative batch setup time for calculating pure (without batch setup time) manufacturing time of component with deterioration. Both are used to define a lack of components at each $l \in L$ as stated in the constraints in (15). Note that the term for the lack of components determines whether there is the lack of components.

$$avt_{kl} = ctc_l + M \bullet y_{kl}, \quad \forall j \in J, \ \forall l \in L \tag{16}$$

The constraints in (16) determine the available assembly time for a product in each $l \in L$ by assuming a significant penalty value for the lack into account.

$$M \bullet y_{kl} \geq \sum_{j \in J} loc_{kjl}, \quad \forall j \in J, \ \forall l \in L \tag{17}$$

The constraints in (17) ensure that y_{kl} is equal to 1, if sum of loc_{kjl} is nonnegative value.

$$C_{max} \geq ctp_k, \quad \forall k \in K \tag{18}$$

Finally, the constraints in (18) ensure the makespan of the TSASP. As a result, the optimal solution of component-manufacturing sequence (CMS), optimal solution of product-assembly sequence (PAS), and position of preventive maintenance activity (PPMA) are determined by w_{ik}, v_{jl}, and z_l.

For the solving the problem by using CPLEX, some equations containing nonlinear functions such as *min* and *max* functions in (6), (9), (10), (13), (14), and (15) should be linearized. Some substitution variables are additionally defined to linearize the equations. The additional equations and additional variables related to the linearization are as follows:

(vi) Decision Variables and Modified Constraints for Linearization

d_k, e_k: substitution variable for calculating the objective in linearized model

f_{jl} = {1, specific $j \in J$ is produced continuously in a specific $l \in L$; 0, Otherwise}

g_{kl} = {1, specific $k \in K$ is assembled at specific $l \in L$; 0, Otherwise}

Constraints (6), (9), (10), (13), (14), and (15) can be modified as follows:

$$v_{jl} + v_{jl-1} \leq f_{jl} + 1, \quad \forall l \in L, \ l > 1, \ \forall j \in J \tag{6a}$$

$$v_{jl} + v_{jl-1} \geq 2 f_{jl}, \quad \forall l \in L, \ l > 1, \ \forall j \in J \tag{6b}$$

$$x_l = \sum_{j \in J} f_{jl}, \quad \forall l \in L, \ l > 1 \tag{6c}$$

Constraints (6a) and (6b) define f_{jl} according to whether a specific $j \in J$ is produced continuously in a specific $l \in L$. Constraint (6c) defines x_l to 1 if a specific $j \in J$ is produced continuously in a specific $l \in L$.

$$ctp_k \geq d_k + \sum_{i \in I} \beta_i \bullet w_{ik}, \quad k = 1 \tag{9a}$$

$$avt_{kl} \geq d_k, \quad k = 1 \tag{9b}$$

$$avt_{kl} - M \left(1 - g_{kl}\right) \geq d_k, \quad k = 1 \tag{9c}$$

$$\sum_{l \in L} g_{kl} = 1, \quad k = 1 \tag{9d}$$

Constraint (9a) defines the completion time of each product for first $k \in K$. Constraint (9b) limits the upper bound of d_k as a minimum value of avt_{kl}. Furthermore, d_k is searched by its lower bound in the optimization problem solving method because d_k is part of the objective. In order to let value of d_k induce as avt_{kl}, constraints for matching the lower bound with the upper bound of d_k are essential and it can be conducted through Constraint (9c) and Constraint (9d). Note that the variables, g_{kl}, will be set as 1 in order for the feasibility of Constraints (9c) and (9d).

$$ctp_k \geq e_k + \sum_{i \in I} \beta_i \bullet w_{ik}, \quad \forall k \in k, \ k > 1 \tag{10a}$$

$$ctp_{k-1} \leq e_k, \quad \forall k \in k, \ k > 1 \tag{10b}$$

$$d_k \leq e_k, \quad \forall k \in k, \ k > 1 \tag{10c}$$

Constraint (10a) represents that avt_{kl} formulated in Constraint (10) is changed to e_k. Constraints (10b) and (10c)

Table 1: The production parameter for Example 1.

Product types	Batch setup time (s)	PMA time (s)	Assembly time (s)	Required number of components (processing time, deterioration rate)			
				J_1 (21, 0.05)	J_2 (27, 0.10)	J_3 (24, 0.15)	J_4 (26, 0.20)
I_1	8	5	36	1	1	-	1
I_2			42	-	-	1	1
I_3			37	1	1	-	1

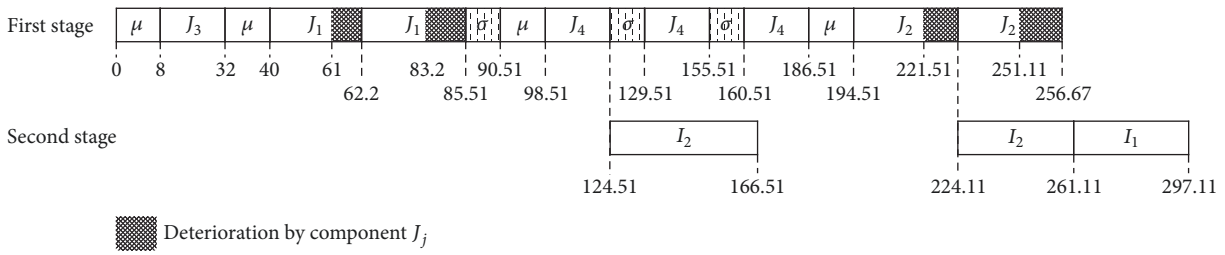

Figure 3: Gantt chart for an example with the optimal schedule.

define the lower bound of e_k as a larger value between ctp_{k-1} and d_k.

$$cts_l \le M \bullet z_l + \mu \quad l = 1 \tag{13a}$$

$$cts_l \le ctc_l, \quad l = 1 \tag{13b}$$

$$cts_l \le M \bullet z_l + cts_{l-1} + \mu \bullet (1 - x_l), \quad \forall l \in L,\ l > 1 \tag{14a}$$

$$cts_l \le ctc_l - cts_{l-1}, \quad \forall l \in L,\ l > 1 \tag{14b}$$

$$loc_{kjl} \ge cnp_{jk} - cnc_{jl}, \quad \forall j \in J,\ \forall k \in k,\ \forall l \in L \tag{15a}$$

$$loc_{kjl} \ge 0, \quad \forall j \in J,\ \forall k \in k,\ \forall l \in L \tag{15b}$$

The constraints in Constraints (13a), (13b), (14a), (14b), (15a), and (15b) can be rewritten as the decomposition constraints of the max or min function included in Constraints (13), (14), and (15) for linearization.

To validate the MILP model, an illustrative example with four components and two product types and detail parameters are provided in Figure 2 and Table 1, respectively. With the optimal CMS, PAS, and PPMA, component-manufacturing completion times, product-assembly completion times, and the makespan are calculated in Figure 3.

The processing times of component 1~4 are 21, 27, 24, and 26, respectively. The assembly times of products 1~3 are 36, 42, and 37, respectively. The batch setup time and PMA time are 8 and 5, respectively. The optimal CMS in the first stage calculated by CPLEX Optimization Studio is 3-1-1-4-4-4-2-2 and the corresponding completion times of the components are 32, 62.2, 85.51, 124.51, 155.51, 186.51, 224.11, and 256.67, respectively. The optimal position of PMAs in the first stage is 0-0-1-1-1-0-0-0. The PAS in the second stage is 2-3-1 and the corresponding available assembly time of the products is 124.51, 224.11, and 261.11, respectively. According to Figure 1, the first PAS, product 2, is composed of a unit each of component 3 and component 4. Product 2 can be assembled when component 3 and component 4 are completely manufactured. In accordance with the optimal CMS, the latest component between the components of product 2 is component 3, and it is completely manufactured at 124.51. Therefore, product 2 can start to be assembled at 124.51. The available assembly time of the remainder products also follows the same procedure. The completion time of the final product in PAS is the makespan of the problem, and it is determined by CPLEX to 297.11. The Gantt chart for the optimal schedule for the example is illustrated in Figure 3.

3. Metaheuristics

The MILP model proposed in this article is not tractable for TSASP because it is a typical combinatorial optimization problem. To solve the large-scale problems, we focus on developing effective metaheuristic algorithms and propose SGA, SHS, HGA, HHS, and HSEA.

3.1. Batching Heuristic Reducing Setup Time and Total Deterioration Time. In this section, *batching heuristic reducing the number of setups and total deterioration time* (BSD) is proposed. The heuristic basically groups the same type of components together to reduce the number of setups and adds PMAs to reduce the total deterioration time of components. Thus, the BSD is guaranteed to minimize the total assembly time in the second stage by reducing the total manufacturing time by constructing an effective CMS and PPMA.

First, BSD is used with the predefined PAS to find the necessary components of each product. Then, the first product in PAS is selected. The component with the highest requirements for the first product is selected. If several of the highest requirements of the components are equal, the component with the longest processing time is selected. When all the components are scheduled, the same process is repeated for

Input: PAS $\longleftarrow$ w
Output: complete CMS $\longleftarrow$ v and PPMA $\longleftarrow$ z
Initialization:
Let requirement of components for each product, $\theta \longleftarrow \emptyset$
Let highest requirement of component for product, $\omega \longleftarrow \emptyset$
Let CMS $v \longleftarrow \emptyset$
Let PPMA z $\longleftarrow \emptyset$
Begin
(1) **Repeat**
(2) **Repeat**
(3) Calculate the θ by requiring number of component types for each product type for w
(4) Select the component with the highest requirement in θ
(5) **If** the number of requirements of component is equal in θ
(6) Select the component with the longest processing time
(7) **End If**
(8) Insert the selected component to ω
(9) **Until** θ
(10) Insert ω to v
(11) **Until** Length of w
(12) Set v as many as number of θ
(13) **Repeat**
(14) **If** completion time of index for current v with a PMA $\leq$ completion time of index for current v
(15) z $\longleftarrow$ 1
(16) **Else**
(17) z $\longleftarrow$ 0
(18) **End If**
(19) **Until** Length of z
End

Algorithm 1: Pseudocode for BSD.

the next product in PAS. The process is repeated until all products in PAS are scheduled. As a result, the corresponding CMS is constructed by the current PAS. Then, to assign the effective PPMA, the PMA time is compared with the total deterioration time of the components from the previous PMA. If PMA time is less than total deterioration time of the previous components, a PMA is added and the current component is manufactured after the PMA. Otherwise, the current component is continuously manufactured without a PMA. The process is repeated until all PPMA are constructed in CMS. The pseudocode of BSD is shown in Algorithm 1.

An illustrative example from Figure 2 and Table 1 describes BSD. Suppose that PAS is $\{I_2\text{-}I_3\text{-}I_1\}$, which means one unit of product 2, one unit of product 3, and one unit of product 1 are sequentially assembled in the second stage. Let CMS initially be $\emptyset$. In order to assemble I_2 at the first position of PAS in the second stage, one unit of J_4 and one unit of J_3 are sequentially scheduled in the first stage, because the longest processing time (LPT) based assignment rule is applied for the CMS rule. Once all components in I_2 are scheduled in the first stage, the current CMS is updated as $\{J_4\text{-}J_3\}$. Next, in order to assemble I_3 at the second position of PAS in the second stage, one unit each of J_2, J_4, and J_1 must be ready by LPT rule. They are sequentially assigned to a position behind the same job to reduce the setup time. The resulting current CMS is updated as $\{J_4\text{-}J_4\text{-}J_3\text{-}J_2\text{-}J_1\}$. At last, in order to assemble I_1 at the third position of PAS in the second stage, one unit each of J_2, J_4, and J_1 must be ready by LPT rule. They are also sequentially assigned to a position behind the same job to reduce the setup time. Thus, the CMS is updated as $\{J_4\text{-}J_4\text{-}J_4\text{-}J_3\text{-}J_2\text{-}J_2\text{-}J_1\text{-}J_1\}$. The PPMA is constructed by comparing the PMA time and total deterioration time of the components from the previous PMA. By the CMS, the corresponding PPMA is 1-1-1-0-1-0-0-0. Each value of PPMA determines the execution of PMAs according to each value. According to Table 1, the deterioration rate and processing time for the first component J_4 are 26 and 0.20, respectively. The deterioration time caused by J_4 is $26 \bullet 0.20 = 5.2$ and the PMA time is 5. At this time, the PMA is executed because the deterioration time caused by J_4 is greater than the PMA time. Then, the first value of the PPMA vector is 1 which means the execution of PMA. The rest of the values are determined in the same way. In particular, the 4^{th} value in PPMA is 0. This value means that it does not execute the PMA. The deterioration time caused by the 4^{th} component, J_3, is $24 \bullet 0.15 = 3.6$. At this time, the PMA is not executed because the deterioration time caused by J_3 is less than the PMA time. This process is repeated until all PPMA corresponding to the CMS are constructed.

3.2. Genetic Algorithms. The genetic algorithm (GA), initially developed by Holland [28], is one of the most powerful and broadly applicable metaheuristic algorithms based on the principles of evolution theory. GA is generally in an effective and efficient algorithm for large-scale combinatorial optimization problems [29]. In this article, two GAs

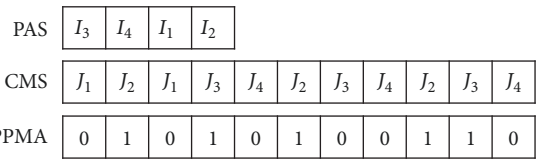

Figure 4: The chromosome structure for genetic algorithms.

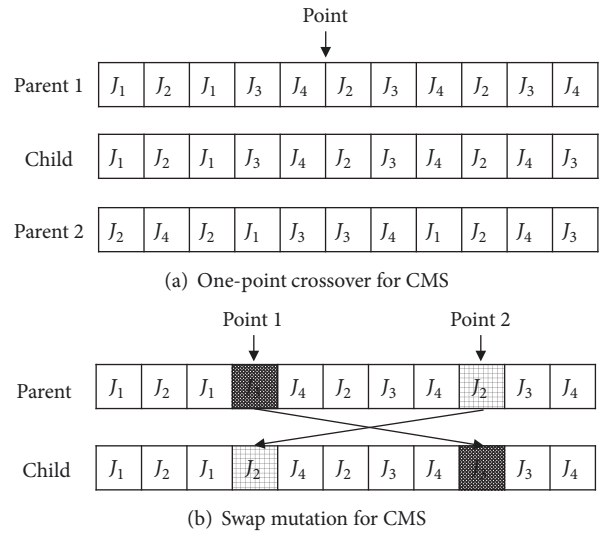

Figure 5: One-point crossover and swap mutation for CMS.

with different solution representations (*chromosomes*) are compared for the two-stage assembly scheduling problem (TSASP). First, SGA uses a chromosome with a complete solution. The SGA using a chromosome with CMS, PAS, and PPMA has a chromosome to represent a complete solution. SGA can explore a wide solution area, because the chromosome of SGA is randomly generated. However, SGA may have a potential vulnerability to find the optimal solution if the optimization problem has optimality characteristics. To improve the vulnerability, the BSD in Section 3.1 must be combined with HGA. Thus, HGA with a partial solution provides a chromosome representing one partial solution (PAS). Then other partial solutions (CMS and PPMA) are determined by BSD using the partial solution decoded from the chromosome.

The structure of the chromosome in HGA for TSASP in this article is required to simultaneously represent a CMS and a PPMA in the first stage, as well as a PAS in the second stage. The CMS represents the manufacturing sequence of the components with consideration of deterioration, PAS represents the assembly sequence of the products, and PPMA represents the position of PMA. CMS and PAS use the actual machining and assembly sequence for the chromosome without additional decoding process. The value of the gene in PPMA is binary and it determines the execution of PMAs according to value of genes (0 or 1). The size of PPMA is equal to size of CMS because the deterioration occurs only for components in the machining machine according to the basic assumption of this problem. The detailed structure of chromosomes is depicted in Figure 4.

According to Figure 4, the PAS is $\{I_3\text{-}I_4\text{-}I_1\text{-}I_2\}$ and the CMS and PPMA are $\{J_1\text{-}J_2\text{-}J_1\text{-}J_3\text{-}J_4\text{-}J_2\text{-}J_3\text{-}J_4\text{-}J_2\text{-}J_3\text{-}J_4\}$ and {0-1-0-1-0-1-0-0-1-1-0}, respectively. An initial population is randomly generated for the first generation. Each chromosome of the population is evaluated through the makespan. New chromosomes for the next generation are generated by using two genetic operations, which are crossover and mutation. In HGA, the one-point crossover and swap mutation are proposed as genetic operations. The main advantage of these operations is able to avoid infeasible solutions during the genetic operations. The one-point crossover and swap mutations are used for all chromosomes equally, and they are illustrated for CMS in Figure 5.

The chromosome of SGA has a three-dimensional string array that represents the CMS, PAS, and PPMA, as shown in Figure 4. The chromosome of HGA has a one-dimensional string array that represents a PAS and the other partial solutions (CMS and PPMA) are determined by BSD using the partial solution decoded from the chromosome. In both SGA and HGA, the chromosomes for the next generation are composed of the top 10% of the chromosomes directly copied from the current generation and the remaining chromosomes are stochastically selected by the roulette-wheel selection from the current generation. The next generation is evaluated, and this process is repeated until the termination criterion is satisfied. The pseudocode of the GA is described in Algorithm 2.

3.3. Harmony Search Algorithms. The harmony search algorithm (HS) was initially studied by Geem et al. [30]. The HS algorithm is inspired by the natural musical performance process, which occurs when a musician searches for a better state of harmony. In the HS algorithm, each solution is called a "*harmony*" which is represented by a m-dimensional real-number vector. An initial population of harmony vectors is randomly generated in the harmony memory (HM). Then a new candidate harmony is generated by three operators, namely, HM consideration, pitch adjustment, and random selection. The new harmony is updated in the memory through a comparison of the candidate and the worst harmony vector.

In this article, two types of HSs with different solution representations are compared for the proposed TSASP. First, SHS uses a solution structure with a complete solution that represents CMS, PAS, and PPMA. Similar to SGA, HHS and HHES combine a HM algorithm with a local search-batching algorithm, BSD. Thus, HHS and HHES construct a solution structure representing one partial solution, PAS. Then other partial solutions, CMS and PPMA are determined by BSD using the PAS. In the HS algorithm, a harmony memory with n harmony vectors, in the jth harmony vector in the HM, $X_j = \{x_j(1), x_j(2), \cdots, x_j(m)\}$, is represented by a m-dimension real-number vector. So, the harmony is necessary to convert it to a solution to evaluate the objective function.

In this article, three solution structures, PAS, CMS, and PPMA, are required to evaluate the makespan. To meet the conversion, a harmony vector is only converted to a PAS. Similar to HGA, other partial solutions which are a CMS

Input: crossover probability, mutation probability, population size, and maximum number of generations, P_c, P_m, N_p, and N_g.
Output: makespan, C_{max}
Initialization:
Let PAS, CMS, and PPMA $w, v, z \longleftarrow \emptyset$
Begin
(1) **If** this algorithm is SGA
(2) Randomly generate an initial population for the first generation based on three single-dimensional arrays
(3) **End If**
(4) **If** this algorithm is HGA
(5) Randomly generate an initial population for the first generation based on a single-dimensional array
(6) **End If**
(7) **Repeat**
(8) **Repeat**
(9) **If** this algorithm is SGA
(10) Calculate the objective value through w, v, and z
(11) **End If**
(12) **If** this algorithm is HGA
(13) Calculate the objective value through v and z derived by MBR and w
(14) **End If**
(15) Randomly select two chromosomes in the current population
(16) **If** random probability $\leq P_c$
(17) Do one-point crossover operation
(18) **End If**
(19) **If** random probability $\leq P_m$
(20) Do swap mutation operation
(21) **End If**
(22) Calculate the fitness value
(23) **Until** N_p
(24) Cloning the value of the top 10% of the chromosomes to the next generation.
(25) Generate the next generation
(26) **Until** N_g.
End

Algorithm 2: Pseudocode of GAs.

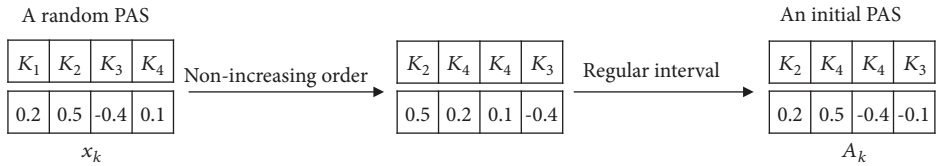

Figure 6: The procedure for generating an initial harmony vector with a regular interval between the minimum and maximum real-numbers.

and a PPMA are determined by BSD using the PAS. Let each index of the vector represent a typical product index from $I = \{1, 2, \cdots, m\}$, where m denotes the number of products assembled in the second stage. Thereafter, the largest position value (LPV) rule is employed to obtain a PAS $\pi = \{\pi(1), \pi(2), \cdots, \pi(K_m)\}$ by ordering the products in their nonincreasing position value of X_j. Then, the harmony vector with a regular interval between the minimum and maximum real-numbers, x_{min} and x_{max}, $A_j = \{a_j(1), a_j(2), \cdots, a_j(m)\}$, is defined by calculating $a(K_i)$ in

$$a\left(K_i\right) = x_{max} - \frac{x_{max} - x_{min}}{m-1} \times (k-1), \quad i = 1, 2, \cdots, m \tag{19}$$

According to Figure 6, we have 4 products and an initial harmony vector is randomly generated as $X_4 = \{0.2, 0.5, -0.4, 0.1\}$ with, $x_{min} = -0.4$ and $x_{max} = 0.5$. A PAS for X_4 is constructed as $\{K_2, K_1, K_4, K_3\}$ by the nonincreasing order of X_4. In accordance with (19), $a(2) = 0.5 - [0.9/3 \times (1-1)] = 0.5$, $a(1) = 0.5 - [0.9/3 \times (2-1)] = 0.2$, $a(4) = 0.5 - [0.9/3 \times (3-1)] = -0.1$, and $a(3) = 0.5 - [0.9/3 \times (4-1)] = -0.4$. As a result, the corresponding initial harmony vector $A_4 = \{0.2, 0.5, -0.4, -0.1\}$.

The initial HM of the first iteration contains a randomly generated harmony vector to represent a PAS. The CMS and PPMA are generated as an actual machining sequence. PAS conducts the decoding process that reverses the order of the encoding process for each iteration. The solution structure of

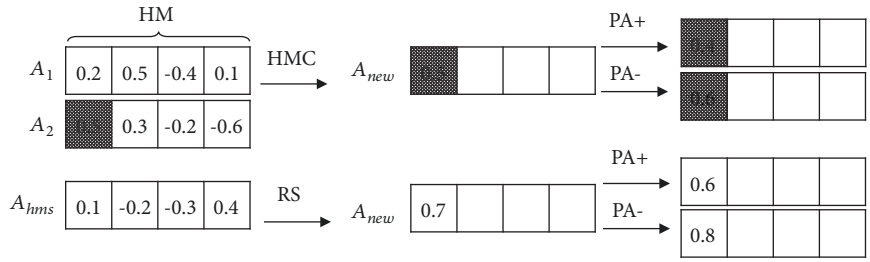

Figure 7: HM, PA, and RS for PAS.

SHS consists a three-dimensional string array that represents a CMS, PAS, and PPMA. Each iteration of SHS improvises a new harmony and an improvising process is composed of HMC, RS, and PA.

For equal-operator effect, HMC, RS, and PA are used for SHS and HHS. First, a uniformly random number is generated. If the random number is less than the operation probability of HM consideration P_{HMC}, $a_{new}(k)$ is generated by the HM consideration and is selected from any harmony in set of $\{1, 2, \cdots, hms(\text{size of HM})\}$. Otherwise, a_{new} is generated by the random selection. The random selection is generated new real number in the search bounds. Second, if the random number is less than operation probability of pitch adjustment P_{PA}, a_{new} is adjusted by a pitch adjustment. Otherwise, proceed to the next step. The pitch adjustment operator is as follows:

$$a_{new}(k) = a_{new}(k) \pm \text{rand}(0, 1) \times bw \tag{20}$$

where rand(0, 1) is a uniformly generated random number between 0 and 1 and bw is the bandwidth.

HMC, PA, and RS are illustrated for PAS in Figure 7.

According to Figure 7, the selected harmony is A_2 and the current variable in the improvising process is 0.5. If the random number is less than P_{HMC}, $a_{new}(1)$ is 0.5. Otherwise, $a_{new}(1)$ is 0.7, the randomly generated variable thorough RS. Next, if the random number is less than P_{PA}, than $a_{new}(1)$ is adjusted as 0.4 or 0.6 and 0.6 or 0.8. The signs are determined by a random number (if the random number > 0.5, then PA+; otherwise, PA−). The HM is updated by the fitness between A_{new} and the worst harmony vector A_w in the HM. Thereafter, the A_w is eliminated in HM and A_{new} is a new member of HM. When the improvement process is finished, and the process is repeated until the termination criterion is satisfied.

Similar to HGA, the harmony vector of HHS consists of a one-dimensional string array that represents a PAS and the other partial solutions (CMS and PPMA) are determined by BSD using the partial solution decoded from the solution structure. The pseudocode of HSs is shown in Algorithm 3.

3.4. Harmony-Search-Based Evolutionary Algorithm. The harmony-search-based evolutionary algorithm (HSEA) combines the advantages of EA and HS in order to effectively solve the large-scale problem of TSASP. Similar to HS, the initial solutions of HSEA (sol_f) for PAS are generated randomly as real-number vectors. To obtain a PAS, a decoding procedure is run to change from a real-number vector to a PAS. Other partial solutions (CMS and PPMA) are determined by BSD using the partial solution decoded from the solution structure. Next, the objective value for each initial solution is calculated by the complete solution. According to random probability, HSEA performs a harmony operator such as HMC, PA, or RS and improves a new solution vector (sv_{new}). If sv_{new} is not applied in HM and the regeneration index (RGI) is not same as the regeneration point (RGP), then RGI is increased by 1. Otherwise, update sol_f and RGI is 0. This process is repeated until the termination criterion (HMS) is satisfied. If RGI is the same as RGP, then the solutions that yield the objective value of the top 10% for sol_f are preserved and the rest solution for sol_f is regenerated randomly.

This process is similar to the selection operator of GA in this article. For the diversity of solutions, another randomly generated solution (sol_g) consisting of the actual sequence and decoded current solution (sol_f) are combined. It performs the genetic operator for the combined solution (sol_{fg}) and calculates the objective value of the combined solution to evaluate the fitness. Thereafter, the dominant solutions are preserved in the size of sol_f and sol_f. In order to continue HSEA, the PAS of the updated sol_f is encoded as real-number vectors. This process is repeated until the termination criterion is satisfied. A flow chart of HSEA is illustrated in Figure 8.

4. Computational Results

To evaluate the performance of the metaheuristic algorithms proposed, computational experiments were conducted by using randomly generated test problems. The computational experiments to test the performance of metaheuristic algorithms were executed with a small-sized problem group and a large-scale problem group. Since the complexity of a problem highly depends upon the length of CMS $|L|$, several instances of two problem groups of small and large-scale problems are randomly generated according to $|L|$. To demonstrate the solvability of the mathematical model, we evaluated the best solution of metaheuristics with the optimal solution using $|L|$ as 8, 10, and 12 for small-sized problem group. The instances were constructed by randomly generating 8 instances within each $|L|$. Since the computing time for CPLEX significantly becomes longer as $|L|$ have increased, we imposed a 7200 (sec.) limit and a particular run was simply terminated even if an optimal solution had not been found. Meanwhile, to evaluate the performance

Input: harmony memory size, iteration, probability of harmony memory consideration, and probability of pitch adjustment, HMS, $Iter$, P_{pa}, and P_{hmc}.
Output: makespan, C_{max}
Initialization:
Let solution vectors for PAS $w \longleftarrow \emptyset$
Begin
(1) **If** this algorithm is SHS
(2) Randomly generate an initial solution for the first iteration based on three single-dimensional arrays
(3) **End If**
(4) **If** this algorithm is HHS
(5) Randomly generate an initial solution for the first iteration based on single-dimensional arrays
(6) **End If**
(7) **Repeat**
(8) **If** this algorithm is SGA
(9) Calculate the objective value through w and, CMS and PPMA generated by actual sequence
(10) **End If**
(11) **If** this algorithm is HGA
(12) Calculate the objective value through CMS and PPMA derived by MBR and w
(13) **End If**
(14) **Until** HMS
(15) Rank by the objective value
(16) **Repeat**
(17) **Repeat**
(18) **If** random probability $\leq P_{hmc}$
(19) Let harmony consideration operation
(20) **End If**
(21) **If** random probability $\leq P_{pa}$
(22) Let pitch adjustment operation
(23) **End If**
(24) **Until** the size of w
(25) Calculate the objective value of new solution vector
(26) **If** the objective value of new solution vector $\leq$ the objective value of harmony with lowest rank
(27) Update the new solution vector
(28) **End If**
(29) **Until** $Iter$.
End

Algorithm 3: Pseudocode of HSs.

Table 2: The generating conditions of experimental data.

Group	Scheduling Period	I	L	α_j	β_i
Small-sized problems	480	2	8	U[60, 80]	U[140, 160]
		3	10	U[40, 60]	U[90, 110]
		4	12	U[30, 50]	U[60, 80]
Large-scale problems	3600	40	160	U[10, 16]	U[30, 40]
		60	400	U[3, 9]	U[20, 30]
		80	640	U[1, 7]	U[15, 25]

between metaheuristic algorithms and gain the insight of the algorithms by altering the parameter $|L|$, we relatively compare the best solution of SGA, SHS, HGA, HHS, and HSEA using $|L|$ as 160, 400, and 640 for large-scale problem group. The instances are constructed by randomly generating 8 instances within each $|L|$.

The generating conditions of the parameters of problem instances for each group were summarized in Table 2. The scheduling periods were assumed to 480 (min.) (12 h × 1day) for the small-sized problem group and 3600 (min.) (12 h × 5day) for large-scale problem group, respectively. The number of products $|I|$ for each $|L|$ was 2, 3, and 4 for the small-sized problem group and 40, 60, and 80 for the large-scale problem group. The batch setup times, preventive maintenance activity times, and deterioration rates were generated as two cases, low-case and high-case. The low-case of batch setup time was fixed as 5 and high-case of batch setup time was fixed as 20. The low-case and high-case of

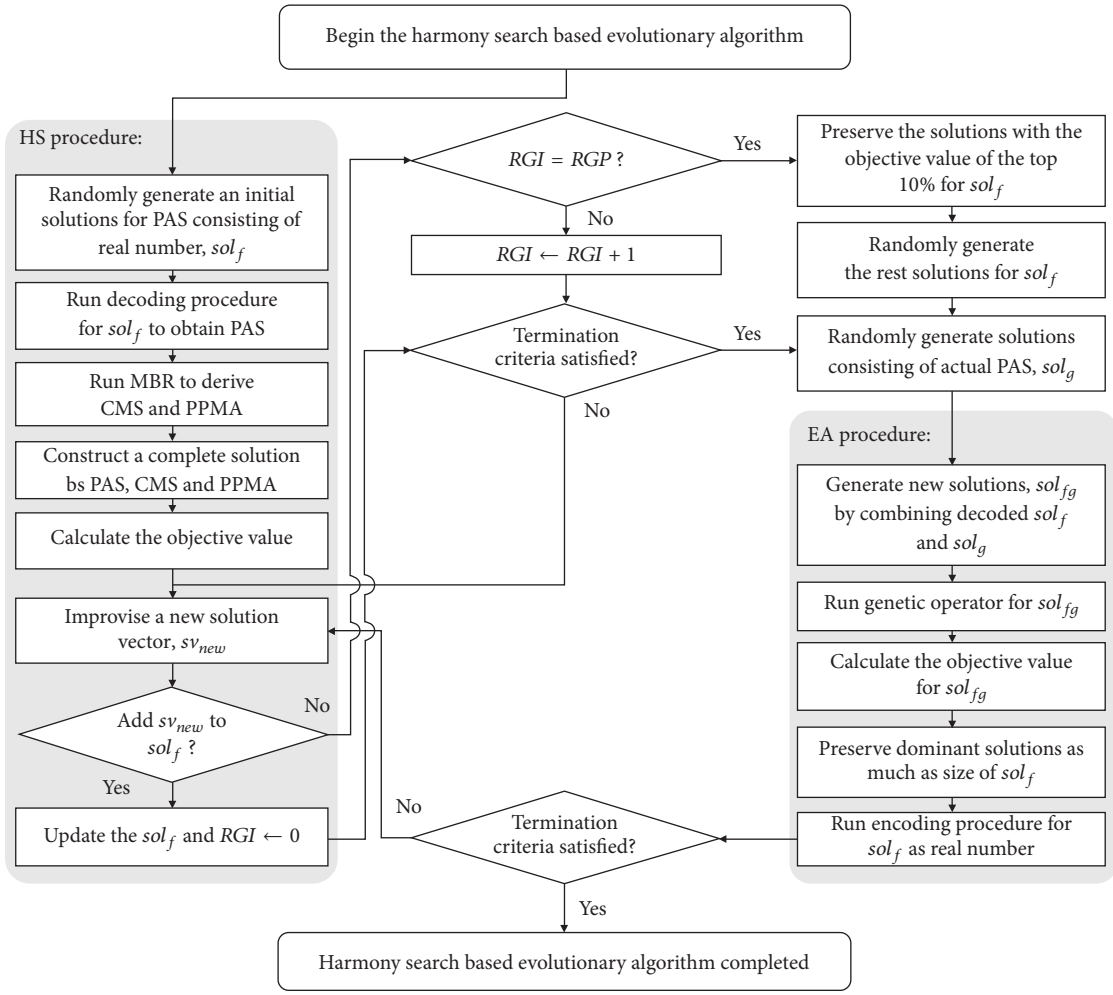

Figure 8: The flow chart of harmony-search-based evolutionary algorithm.

preventive maintenance activity times were also fixed 5 and 20, respectively. The low-case and high-case of deterioration rate were uniformly generated in $[0.01, 0.05]$ and $[0.1, 0.3]$, respectively.

The all metaheuristic algorithms were executed with 30 replications. In order to be equal comparison of SGA, HGA, SHS, HHS, and HSEA, a population size was set as $2 \times I$ and a termination time was set as the elapsed time when SGA as a base heuristic is converged with no-improvement or repeats until 1,000 generations. The crossover, mutation, harmony memory consideration, and pitch adjustment rate were 0.8, 0.2, 0.7, and 0.3, respectively. The parameters of proposed algorithms were predetermined by extensive preliminary experimentations. The MILP model presented in Section 2 was coded in ILOG CPLEX 12.5 and the all metaheuristic algorithms were coded by C#. All the experiments solving each test problem using CPLEX and metaheuristic algorithms were run on a PC with Intel core i7-4770 CPU with 4GB RAM and Windows 7 operating system. The test results of small-sized problem group and large-scale problem group are summarized in Tables 3 and 4. It shows the optimal solution by CPLEX, mean. For performance measures, the mean objective function value of all replications (*Mean*), the relative percentage deviation (*RPD*), and mean absolute deviation (*MAD*) of each replication are calculated. The *RPD* and *MAD* are expressed in

$$RPD\,(\%) = \frac{MH_{sol} - Best}{Best} \times 100, \tag{21}$$

$$MAD\,(\%) = \frac{\left|MH_{sol} - \overline{MH_{sol}}\right|}{\overline{MH_{sol}}} \times 100, \tag{22}$$

where MH_{sol} is the objective function value and $\overline{MH_{sol}}$ is the mean objective function value of all replications obtained by each metaheuristic and *Best* is the best objective function value of all experiments (CPLEX, SGA, HGA, SHS, HHS, and HSEA) for each test problem. *Best* can be the optimal solution if CPLEX obtains the optimal solution.

For the tests of small-sized problem group, CPLEX was unable to derive the optimal solution for most of test problems with $|L| \geq 12$ in 7200 (sec.). We defined the values of *RPDs* and *MADs* of test problems with $|L| \geq 12$ unable to

Table 3: The results of the small-sized problem group.

	CPLEX		SGA			SHS			HGA			HHS			HSEA		
$\|L\|$	Opt.	Time	*Mean*	*RPD*	*MAD*	*Mean*	*RPD*	*MAD*	*Mean*	*RPD*	*MAD*	*Mean*	*RPD*	*MAD*	*Mean*	*RPD*	*MAD*
8	297.1	35.6	297.1	0.0	0.0	297.1	0.0	0.0	297.1	0.0	0.0	297.1	0.0	0.0	297.1	0.0	0.0
8	285.7	21.4	285.7	0.0	0.0	285.7	0.0	0.0	285.7	0.0	0.0	285.7	0.0	0.0	285.7	0.0	0.0
8	278.1	27.5	278.1	0.0	0.0	278.1	0.0	0.0	278.1	0.0	0.0	278.1	0.0	0.0	278.1	0.0	0.0
8	291.6	11.2	291.6	0.0	0.0	291.6	0.0	0.0	291.6	0.0	0.0	291.6	0.0	0.0	291.6	0.0	0.0
8	262.2	17.6	262.2	0.0	0.0	262.2	0.0	0.0	262.2	0.0	0.0	262.2	0.0	0.0	262.2	0.0	0.0
8	268.5	18.4	268.5	0.0	0.0	268.5	0.0	0.0	268.5	0.0	0.0	268.5	0.0	0.0	268.5	0.0	0.0
8	243.4	32.8	243.4	0.0	0.0	243.4	0.0	0.0	243.4	0.0	0.0	243.4	0.0	0.0	243.4	0.0	0.0
8	258.7	26.3	258.7	0.0	0.0	258.7	0.0	0.0	258.7	0.0	0.0	258.7	0.0	0.0	258.7	0.0	0.0
10	332.4	246.8	332.4	0.0	0.0	334.4	1.3	0.6	332.4	0.0	0.0	332.4	0.0	0.0	332.4	0.0	0.0
10	346.3	216.6	346.3	0.0	0.0	346.3	0.0	0.0	346.3	0.0	0.0	346.3	0.0	0.0	346.3	0.0	0.0
10	346.8	125.9	346.8	0.0	0.0	346.8	0.0	0.0	346.8	0.0	0.0	346.8	0.0	0.0	346.8	0.0	0.0
10	329.1	276.2	329.1	0.0	0.0	329.1	0.0	0.0	329.1	0.0	0.0	335.1	2.0	2.8	329.1	0.0	0.0
10	315.8	311.7	315.8	0.0	0.0	315.8	0.0	0.0	315.8	0.0	0.0	315.8	0.0	0.0	315.8	0.0	0.0
10	312.7	592.2	312.7	0.0	0.0	312.7	0.0	0.0	312.7	0.0	0.0	312.7	0.0	0.0	312.7	0.0	0.0
10	328.6	167.4	328.6	0.0	0.0	328.6	0.0	0.0	328.6	0.0	0.0	328.6	0.0	0.0	328.6	0.0	0.0
10	341.9	565.4	358.9	16.3	0.5	341.9	0.0	0.0	343.9	1.3	1.1	341.9	0.0	0.0	341.9	0.0	0.0
12	N/A	7200+	365.4	N/A	N/A	366.4	N/A	N/A	363.4	N/A	N/A	363.4	N/A	N/A	363.4	N/A	N/A
12	N/A	7200+	340.8	N/A	N/A	340.8	N/A	N/A	340.8	N/A	N/A	340.8	N/A	N/A	340.8	N/A	N/A
12	N/A	7200+	419.3	N/A	N/A	419.3	N/A	N/A	419.3	N/A	N/A	419.3	N/A	N/A	419.3	N/A	N/A
12	337.3	974.5	337.3	0.0	0.0	357.6	19.8	1.7	337.3	0.0	0.0	337.3	0.0	0.0	337.3	0.0	0.0
12	N/A	7200+	411.6	N/A	N/A	411.6	N/A	N/A	411.6	N/A	N/A	411.6	N/A	N/A	411.6	N/A	N/A
12	305.9	424.1	309.9	11.5	3.8	305.9	0.0	0.0	305.9	0.0	0.0	305.9	0.0	0.0	305.9	0.0	0.0
12	N/A	7200+	364.5	N/A	N/A	364.5	N/A	N/A	364.5	N/A	N/A	364.5	N/A	N/A	364.5	N/A	N/A
12	N/A	7200+	340.3	N/A	N/A	340.3	N/A	N/A	340.3	N/A	N/A	340.3	N/A	N/A	340.3	N/A	N/A
Avg.				1.3	0.2		1.1	0.1		0.1	0.1		0.1	0.1		0.0	0.0

Table 4: The results of the large-scale problem group.

	SGA			SHS			HGA			HHS			HSEA		
$\|L\|$	*Mean*	*RPD*	*MAD*	*Mean*	*RPD*	*MAD*	*Mean*	*RPD*	*MAD*	*Mean*	*RPD*	*MAD*	*Mean*	*RPD*	*MAD*
160	3313.4	12.2	4.9	3297.3	11.6	0.7	3268.9	10.7	0.6	3262.7	10.5	0.6	3062.0	3.7	1.7
160	3116.7	10.3	1.8	3102.9	9.8	0.5	3121.2	10.5	0.8	3116.0	10.3	0.5	2883.6	2.1	1.0
160	4512.0	10.6	1.9	4433.2	8.7	0.5	4331.5	6.2	0.5	4327.1	6.1	0.4	4133.9	1.3	0.8
160	3632.5	13.4	1.7	3583.4	11.9	0.8	3497.8	9.2	0.6	3502.5	9.4	0.4	3257.9	1.7	1.2
160	6747.1	11.8	2.1	6583.9	9.1	0.2	6383.2	5.8	0.3	6384.5	5.8	0.2	6049.2	0.3	0.1
160	5199.8	17.3	3.6	5003.4	12.9	0.3	4841.6	9.3	0.7	4836.9	9.2	0.4	4533.4	2.3	1.2
160	5829.8	9.4	2.0	5827.3	9.4	0.2	5711.3	7.2	0.2	5720.7	7.4	0.5	5389.5	1.1	0.6
160	5823.8	9.5	2.0	5821.3	9.4	0.2	5705.3	7.2	0.2	5714.7	7.4	0.5	5383.5	1.2	0.6
400	3211.2	14.3	0.9	3219.4	14.6	0.7	3092.8	10.1	0.5	2883.2	2.7	1.3	3072.4	9.4	1.0
400	4348.8	7.2	2.2	4319.1	6.4	0.6	4364.8	7.5	0.2	4356.9	7.4	0.4	4113.9	1.4	0.6
400	3669.8	14.1	2.7	3716.	15.5	0.6	3588.7	11.5	1.2	3588.2	11.5	1.1	3300.2	2.6	1.0
400	4235.3	20.6	4.4	4052.8	15.4	0.4	3900.5	11.0	0.4	3923.1	11.7	0.3	3583.0	2.0	0.8
400	5191.6	12.3	3.3	5074.4	9.8	0.4	4971.4	7.6	0.3	4953.4	7.2	0.6	4706.3	1.8	0.9
400	5372.9	8.5	0.9	5421.7	9.3	0.1	5306.4	7.1	0.3	5307.2	7.1	0.4	5000.9	1.0	0.5
400	5368.6	8.4	0.9	5417.4	9.4	0.1	5302.1	7.1	0.2	5302.9	7.1	0.4	4996.6	0.9	0.5
400	4191.2	14.7	1.3	4092.0	12.0	0.6	3971.0	8.7	0.4	4030.0	10.3	0.2	3760.6	2.9	1.3
640	3366.8	11.8	2.7	3479.	15.5	0.3	3295.1	9.4	0.5	3297.7	9.5	0.4	3075.0	2.1	1.4
640	3489.8	8.5	2.3	3527.6	9.7	0.5	3533.5	9.9	0.5	3481.1	8.3	0.4	3304.1	2.8	1.4
640	3856.9	10.7	1.3	3823.6	9.8	0.8	3788.1	8.8	0.5	3776.6	8.4	0.3	3553.5	2.0	1.4
640	3822.5	18.9	5.9	3791.9	18.0	0.3	3593.7	11.8	0.4	3590.0	11.7	0.4	3286.2	2.3	1.4
640	6318.8	9.2	2.6	6288.3	8.6	0.4	6139.5	6.1	0.3	6138.0	6.0	0.5	5814.6	0.5	0.3
640	4194.8	14.8	1.2	4095.6	12.1	0.6	3974.6	8.8	0.3	4033.6	10.4	0.2	3764.2	3.0	1.3
640	4190.5	14.9	1.2	4091.3	12.2	0.5	3970.3	8.9	0.3	4029.3	10.5	0.2	3759.9	3.1	1.3
640	5825.1	9.5	1.9	5822.6	9.4	0.2	5706.6	7.2	0.2	5716.0	7.4	0.5	5384.8	1.2	0.7
Avg.		12.3	2.3		11.3	0.4		8.7	0.4		8.5	0.4		2.2	0.9

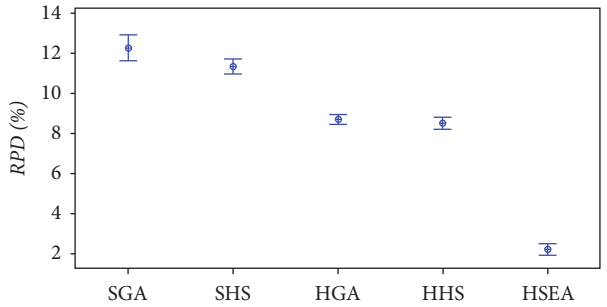

Figure 9: Mean plots and Turkey HSD intervals at the 95% confidence level of each algorithm.

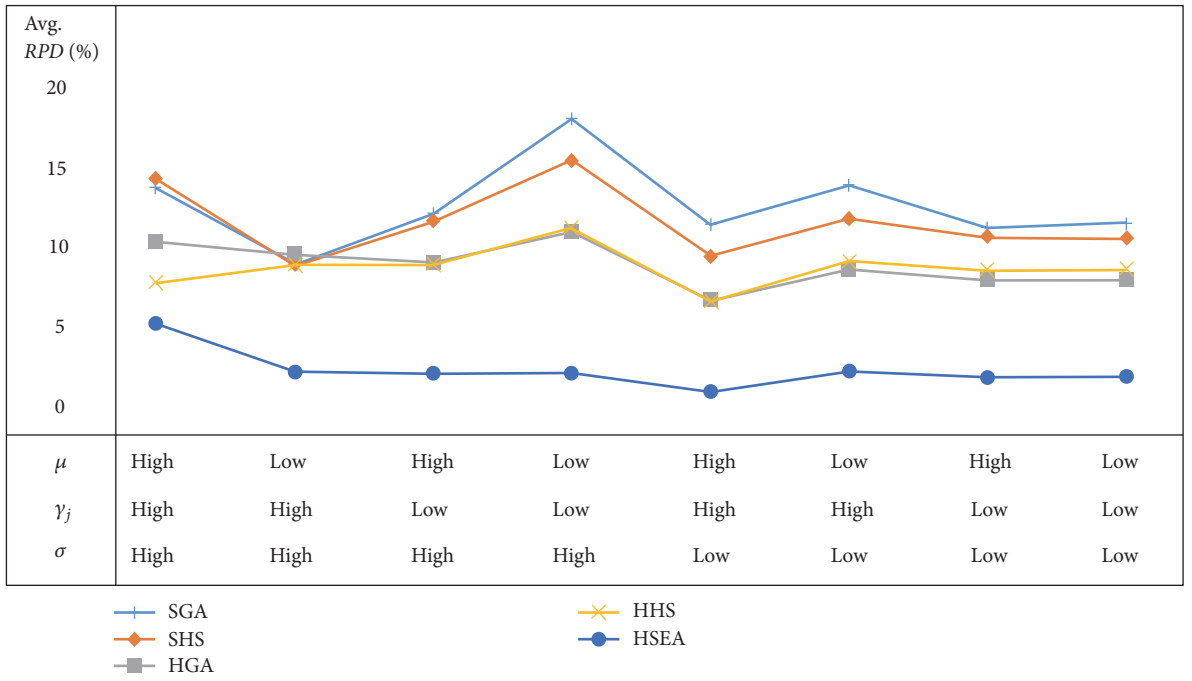

Figure 10: The comparison of average RPD value for five metaheuristic algorithms in three significant factors.

derive the optimal solution. Thus, the average values of *RPDs* and *MADs* were calculated all the values of test problems except for the values of test problems with N/A. For the small-sized problem group, the low values of *RPD*s and *MAD*s in Table 3 indicate that all proposed metaheuristic algorithms perform well. Especially, HSEA was very close to the optimal solution with achieving the highest frequency of finding the optimal solution among other metaheuristic algorithms. The computational times of all metaheuristics algorithms for small-sized problem group were small enough to obtain solutions within reasonable computational times of 1 (sec.). These results mean that HSEA is very effective algorithms with low variation for small-sized problem group.

For large-scale problem group, the *RPD*s and *MAD*s of HSEA were much better than the other metaheuristic algorithms in all the test problem groups. The results indicated that HSEA could significantly improve the performance. Both the *RPD*s and *MAD*s of HSEA are 2.2 and 0.9, which are nearly 0.

Figure 9 showed the mean plots and Tukey HSD intervals at the 95% confidence level for all problems by SGA, HGA, SHS, HHS, and HSEA in Table 4. It clearly illustrated that there were statistically significant differences between the *RPD* values among SGA, HGA, SHS, HHS, and HSEA.

Figure 10 showed the pattern of the average *RPDs* in SGA, HGA, SHS, HHS, and HSEA by changing three significant performance factors, batch setup times (μ), PMA time (σ), and deterioration rate (γ_j). In this figure, HSEA clearly showed a low average *RPD* in all the combinations of the factors. This result indicates that HSEA is robust for various combinations of significant performance factors of the algorithms. The average *RPDs* of SGA, HGA, SHS, and HHS except for HSEA relatively show the highest value (the worst performance) in the case of high PMA time, low deterioration rate, and low batch setup time. The main reason of the occurrence of this phenomenon is that the makespan

becomes sensitive to small change of the number of PMAs as the PMA time is increased.

5. Conclusions

In this article, a novel MILP model was developed for TSASP with batch setup times, time-dependent deterioration, and preventive maintenance activities in dynamic-component size and to minimize the makespan. In TSASP, n products ordered by various customers are scheduled. In the first stage, $|L|$ required number of components to be assembled to the corresponding products in the second stage must be manufactured. When all the required components for one corresponding product ordered were made, an assembly machine in second stage can immediately assemble the components into the product. Since the MILP model was not tractable for the problems $|L| \geq 12$ within a reasonable computing time, metaheuristic algorithms (SGA, HGA, SHS, HHS, and HSEA) were proposed. We executed the computational experiments with two groups, namely, a small-sized problem group and a large-scale problem group. Based on the experimental results, we found that HSEA was very effective algorithm with low *RPD* in reasonable computational time for the TSASP. Furthermore, HSEA showed that the algorithm is robust for various combinations of significant performance factors of the algorithms.

The future research can be divided into three directions. Firstly, the study can be extended on other kinds of setup and preventive maintenance activity such as sequence dependent setup time and position-based preventive maintenance activity. Secondly, the study should be extended to the deterioration of the assembly process in the second stage, because the most manual assembly process with human fatigue is popular. Finally, the study should be extended to apply other manufacturing process with three-stage assembly scheduling problem and flexible flow shop scheduling problems.

Data Availability

All data can be accessed in the Computational Results section of this article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Incheon National University Research Grant in 2018 (Grant no. 2018-0075).

References

[1] 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.

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

[3] 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.

[4] 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.

[5] D. He, A. Babayan, and A. Kusiak, "Scheduling manufacturing systems in an agile environment," *Robotics and Computer-Integrated Manufacturing*, vol. 17, no. 1-2, pp. 87–97, 2001.

[6] D. He and A. Babayan, "Scheduling manufacturing systems for delayed product differentiation in agile manufacturing," *International Journal of Production Research*, vol. 40, no. 11, pp. 2461–2481, 2002.

[7] H.-T. Lin and C.-J. Liao, "A case study in a two-stage hybrid flow shop with setup time and dedicated machines," *International Journal of Production Economics*, vol. 86, no. 2, pp. 133–143, 2003.

[8] 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.

[9] H.-S. Yan, X.-Q. Wan, and F.-L. Xiong, "A hybrid electromagnetism-like algorithm for two-stage assembly flow shop scheduling problem," *International Journal of Production Research*, vol. 52, no. 19, pp. 5626–5639, 2014.

[10] G. M. Komaki and V. Kayvanfar, "Grey Wolf Optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time," *Journal of Computational Science*, vol. 8, pp. 109–120, 2015.

[11] J. Deng, L. Wang, S.-Y. Wang, and X.-L. Zheng, "A competitive memetic algorithm for the distributed two-stage assembly flow-shop scheduling problem," *International Journal of Production Research*, vol. 54, no. 12, pp. 3561–3577, 2016.

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

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

[14] G. Mosheiov and A. Sarig, "Scheduling a maintenance activity and due-window assignment on a single machine," *Computers & Operations Research*, vol. 36, no. 9, pp. 2541–2545, 2009.

[15] T. C. Cheng, W.-C. Lee, and C.-C. Wu, "Single-machine scheduling with deteriorating jobs and past-sequence-dependent setup times," *Applied Mathematical Modelling*, vol. 35, no. 4, pp. 1861–1867, 2011.

[16] H.-T. Lee, D.-L. Yang, and S.-J. Yang, "Multi-machine scheduling with deterioration effects and maintenance activities for minimizing the total earliness and tardiness costs," *The International Journal of Advanced Manufacturing Technology*, vol. 66, no. 1–4, pp. 547–554, 2013.

[17] S.-J. Yang, "Unrelated parallel-machine scheduling with deterioration effects and deteriorating multi-maintenance activities for minimizing the total completion time," *Applied Mathematical Modelling: Simulation and Computation for Engineering and Environmental Systems*, vol. 37, no. 5, pp. 2995–3005, 2013.

[18] M. Ji, J. Ge, K. Chen, and T. C. E. Cheng, "Single-machine due-window assignment and scheduling with resource allocation, aging effect, and a deteriorating rate-modifying activity," *Computers & Industrial Engineering*, vol. 66, no. 4, pp. 952–961, 2013.

[19] J. N. D. Gupta and S. K. Gupta, "Single facility scheduling with nonlinear processing times," *Computers & Industrial Engineering*, vol. 14, no. 4, pp. 387–393, 1988.

[20] S. Browne and U. Yechiali, "Scheduling deteriorating jobs on a single processor," *Operations Research*, vol. 38, no. 3, pp. 495–498, 1990.

[21] C.-Y. Lee and V. J. Leon, "Machine scheduling with a rate-modifying activity," *European Journal of Operational Research*, vol. 128, no. 1, pp. 119–128, 2001.

[22] S.-R. Cheng, "A single-machine two-agent scheduling problem by GA approach," *Asia-Pacific Journal of Operational Research*, vol. 29, no. 2, Article ID 12500133, p. 22, 2012.

[23] W.-H. Wu, S.-R. Cheng, C.-C. Wu, and Y. Yin, "Ant colony algorithms for a two-agent scheduling with sum-of processing times-based learning and deteriorating considerations," *Journal of Intelligent Manufacturing*, vol. 23, no. 5, pp. 1985–1993, 2012.

[24] W.-H. Wu, J. Xu, Y. Yin, I.-F. Cheng, and C.-C. Wu, "A tabu method for a two-agent single-machine scheduling with deterioration jobs," *Computers & Operations Research*, vol. 40, no. 8, pp. 2116–2127, 2013.

[25] A. J. Ruiz-Torres, G. Paletta, and E. Pérez, "Parallel machine scheduling to minimize the makespan with sequence dependent deteriorating effects," *Computers & Operations Research*, vol. 40, no. 8, pp. 2051–2061, 2013.

[26] C. M. Joo and B. S. Kim, "Genetic algorithms for single machine scheduling with time-dependent deterioration and rate-modifying activities," *Expert Systems with Applications*, vol. 40, no. 8, pp. 3036–3043, 2013.

[27] B. D. Chung and B. S. Kim, "A hybrid genetic algorithm with two-stage dispatching heuristic for a machine scheduling problem with step-deteriorating jobs and rate-modifying activities," *Computers & Industrial Engineering*, vol. 98, pp. 113–124, 2016.

[28] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, University of Michigan Press, Oxford, UK, 1975.

[29] M. Gen and R. Cheng, "Genetic algorithms and engineering optimization," *Technometrics*, vol. 44, no. 1, p. 95, 2000.

[30] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.