mathematics

MDPI

Article

An Optimization Problem of Distributed Permutation Flowshop Scheduling with an Order Acceptance Strategy in Heterogeneous Factories

Seung Jae Lee and Byung Soo Kim *

Department of Industrial and Management Engineering, Incheon National University, 119, Academy-ro, Yeonsu-gu, Incheon 22012, Republic of Korea; lsj0804@inu.ac.kr
* Correspondence: bskim@inu.ac.kr

Abstract: This paper addresses a distributed permutation flowshop scheduling problem with an order acceptance strategy in heterogeneous factories. Each order has a related revenue and due date, and several flowshop machines are operated in each factory, and they have a distinct sequence-dependent setup time. We select/reject production orders, assign the selected orders to the factories, and determine the permutation manufacturing sequence in each factory to maximize the total profit. To optimally solve the scheduling problem, we formulate the scheduling problem as a mixed integer linear programming model to find an optimal solution for small-sized experiments. Then, we propose two population-based algorithms, a genetic algorithm and particle swarm optimization for large-sized experiments. We proved that the proposed genetic algorithm effectively and efficiently solves the problem to guarantee a near optimal solution through computational experiments. Finally, we conduct a sensitivity analysis of the genetic algorithm to observe the relationship between order selection, revenue, and order tardiness cost.

Keywords: scheduling; distributed manufacturing; order acceptance; mixed-integer linear programming; genetic algorithm

MSC: 90B30

check for updates

Academic Editors: Javier Alcaraz and Frank Werner

Received: 5 February 2025
Revised: 1 March 2025
Accepted: 4 March 2025
Published: 6 March 2025

Citation: Lee, S.J.; Kim, B.S. An Optimization Problem of Distributed Permutation Flowshop Scheduling with an Order Acceptance Strategy in Heterogeneous Factories. *Mathematics* **2025**, *13*, 877. https://doi.org/10.3390/math13050877

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Distributed manufacturing (DM) is a manufacturing policy that produces production orders by decentralizing the orders to several production plants [1]. The DM strategy helps to effectively enhance productivity, resilience, etc., since it is practically able to use decentralized manufacturing resources and capabilities. Therefore, real-world manufacturing systems have been increasingly adopting decentralized approaches, a trend that has been further accelerated by the COVID-19 pandemic [2]. Furthermore, according to the existing literature, various scheduling studies have been conducted to operate manufacturing environments applying the DM strategy [3–12]. A distributed permutation flowshop scheduling problem (DPFSP) addressed by [13] is one of the scheduling topics attracted in the DM scheduling studies. To maximize/minimize the related objective function, the main decision contents of the DPFSP are the factory assignment and permutation manufacturing sequence in each factory.

In an order acceptance and scheduling (OAS) problem, a decision maker determines the selection or rejection of orders [14]. After selection, the decision maker determines the schedule of the selected orders. In the OAS, the decision maker rejects some orders

depending upon the trade-off between revenue and cost, the business direction of the company, etc. By using the order acceptance (OA) strategy, it helps to enhance the profit of the company. The OA strategy has been used in diverse scheduling studies to maximize profit or minimize cost.

In this paper, we address a distributed permutation flowshop scheduling problem with an order acceptance strategy in heterogeneous factories (DPFSP-OAHF) to maximize the total profit. The problem addressed is an integrated scheduling problem of both the DPFS and OAS. There are several orders and heterogeneous permutation flowshops. Each order has related revenue and due date, and each flowshop considers infinite buffers between operation stages and distinct sequence-dependent setup times within operation stage. To optimally solve the addressed problem, we simultaneously determine the selection of production orders, the assignment of the selected orders to factories, and the sequencing of manufacturing the assigned orders in each factory, respectively. Total profit contains the total revenue and the order tardiness cost. We have to select production orders by considering the trade-off between revenue and tardiness cost since the order tardiness cost is multiplied with the related revenue of an order. To determine the assignment, we assign each order to one factory without considering the order splitting. Furthermore, we determine the permutation sequence of each factory that minimizes the total order tardiness cost.

In Section 2, we present a literature survey related to the addressed problem. We formulate a mixed-integer linear programming (MILP) to optimally solve the problem in Section 3. In Section 4, we propose a genetic algorithm (GA) and particle swarm optimization (PSO). In Section 5, we show that the GA outperforms the PSO through numerical experiments. Also, we conduct a sensitivity analysis of the GA to observe the relationship between order selection, revenue, and order tardiness cost. We conclude this paper in Section 6.

2. Related Work

As mentioned in Section 1, the addressed problem in this study is an extended scheduling study of the DPFSP addressed by [13]. In this section, we conduct a literature survey of the extended scheduling studies of the DPFS published in the last five years.

In [15,16], the extended scheduling studies considering sequence-dependent setup times were addressed. They studied a homogeneous flowshops framework to minimize the makespan, (C_{max}). A mathematical model and a variation of an iterated greedy (IG) algorithm, which was IGR, were proposed by [16]. Constraint programming (CP) and MILP models were developed, and to efficiently solve the problem, an evolution strategy algorithm (ES_en) was proposed in [15]. The DPFSP was addressed by [17]. They also studied the homogeneous flowshops framework and presented an evolution algorithm memetic, called MDDE algorithm, to minimize C_{max}. The DPFSP was also studied [18] to minimize C_{max}. A novel heuristic was proposed to obtain lower bounds, and the CP model was developed to solve the DPFSP. In [19], a DPFSP was extended by considering flexible assembly and batch delivery, and the problem was defined as DAPFSP-FABD to minimize delivery and tardiness costs (DTC). They studied a manufacturing framework that combined distributed permutation homogeneous flowshops and identical assembly factories. Furthermore, they considered batch delivery (BD) as a transportation policy after processing. To solve DAPFSP-FABD, they developed single solution-based metaheuristics. In [20], they addressed a DPFSP considering assembly process and multi-objective function. They minimized total flowtime (TF) and tardiness (TT) with a two-phase evolution algorithm (TEA). A DPFSP considering a lot-streaming (DLPFSP) was presented by [21]. They minimized C_{max} with a mathematical model harmony search (HS), Jaya, artificial

bee colony (ABC), GA, and PSO. They assumed that the processing times included setup times, and it means that they considered sequence-independent setup times. A DPFSP with heterogeneous factories was studied in [22,23]. In [22], they minimized multi-objective criteria C_{max} and total energy consumption (TEC). In [23], they considered a lot-stream and carry-over sequence-dependent setup time (CSDST). They developed the MILP model to find optimal C_{max} and proposed constructive heuristics and a variation of artificial bee colony algorithm, called NEABC. To minimize total tardiness (TD), a DPFSP was studied by [24]. They studied a homogeneous flowshops framework and proposed the MILP model, Harris hawks optimization (HHO), and IG algorithms. A DPFSP considering tardiness and rejection costs was studied [25]. They defined their scheduling problem as DPTR and rejected some production orders that were not able to be produced within a common deadline. To solve DPTR, they developed the MILP model and IG algorithm (IG_TR). Zhang and Zhen [26] studied the energy-efficient distributed heterogeneous permutation flowshop scheduling problem. They proposed a competitive multilevel Jaya algorithm with a SPA-based multi-directional local search. Song et al. [27] studied the worker fatigue dual-resource-constrained distributed hybrid flowshop scheduling. They presented a Q-learning-driven multi-objective evolutionary algorithm to solve the problem. Zhu et al. [28] dealt with the distributed heterogeneous mixed no-wait flowshop scheduling problem. They proposed a cooperative learning-aware dynamic hierarchical hyper-heuristic.

We summarize the literature survey in Table 1. Moreover, the taxonomy review for the DPFSP was conducted by [29,30].

Table 1. Literature survey.

	OA	Framework			Setup		Etc.	Method	Objective
		Homo	Hetero	Etc.	SD	SI			
[16]		√			√			Mathematical model, IGR	Min. C_{max}
[15]		√			√			MILP, CP, ES_en	Min. C_{max}
[17]		√						MMDE	Min. C_{max}
[18]		√						Heuristic, CP	Min. C_{max}
[19]		√		Assembly factories			Batch Delivery	Heuristic, VND, IG	Min. DTC
[20]		√		Assembly line				TEA	Min. {TF, TT}
[21]		√				√	Lot-Streaming	Mathematical model, GA, PSO, ABC, HS, Jaya	Min. C_{max}
[22]			√					KMOEA/D	Min. $\{C_{max}, TEC\}$
[23]			√		√		Lot-streaming	MILP, Constructive Heuristic, NEABC	Min. C_{max}
[24]		√						MILP, HHO, IG	Min. TD
[25]	√	√					Deadline	MILP, IG_TR	Max. Total profit
[26]			√					Jaya	Min. $\{C_{max}$, TEC$\}$
[27]			√				Worker fatigue	Q-learning driven multi-objective evolutionary algorithm	Min. C_{max}
[28]			√		√		No-wait	Hyper-heuristic	Min. C_{max}
This	√		√		√			MILP, GA, PSO	Max. Total profit

Homo: homogeneous; Hetero: heterogeneous; SD: sequence-dependent; SI: sequence-independent, TEC: total energy consumption.

Based on the presented literature survey, we have two major contributions in terms of the problem domain and methodology in this paper. For the problem domain, we address the problem with novelty by adopting heterogeneous factories, sequence-dependent setup times, and an order acceptance strategy to extend the DPFSP. To our best knowledge, there is no comprehensive integration study of DM and OAS despite their importance for enhancing productivity, agility, and profitability. For the methodology, we derive a MILP

model to find an optimal solution. Also, we propose a GA with a solution representation and decoding procedure to find a near-optimal solution effectively and efficiently for the addressed problem.

3. Mathematical Model

3.1. Assumptions of a Mathematical Model

The assumptions of the mathematical model are as follows:

(1) There are several distributed factories.
(2) Each factory has one set of flowshop machines.
(3) Each factory has different processing times for orders and statuses.
(4) Pre-emption of orders is not allowed.
(5) The lead time from when an order is completed on the machines of the factories to when it is delivered to the customer is not considered.

3.2. MILP Model

In this section, we describe an MILP model to formulate the addressed problem.

Parameters & Sets

F	Set of distributed factories, $F = \{1, 2, \ldots, n_F\}$
S	Set of operation stage, $S = \{1, 2, \ldots, n_S\}$
I	Set of orders, $I = \{1, 2, \ldots, n_I\}$
ID	Set of orders and dummy order, $ID = \{0, 1, 2, \ldots, n_I\}$
R_i	Revenue related to order $i \in I$
D_i	Due date related to order $i \in I$
PT_{si}^{f}	Processing times of production order $i \in I$ at stage $s \in S$ in distributed factory $f \in F$
CO_{ij}^{fs}	Sequence-dependent setup times between orders $i, j \in ID$ at stage $s \in S$ in factory $f \in F$
α	Scaling parameter of tardiness costs
M	Large number

Decision Variables

z_i	If order $i \in I$ is selected, 1; Otherwise, 0
y_i^f	If order $i \in I$ is produced in factory $f \in F$, 1; Otherwise, 0
x_{ij}^f	If order $j \in ID$ is produced immediately after order $i \in ID$ in factory $d \in F$, 1; Otherwise, 0
st_{ti}^f	Start time of order $i \in I$ at operation stage $s \in S$ in factory $f \in F$
ct_{ti}^f	Completion time of order $i \in I$ at operation stage $s \in S$ in factory $f \in F$
c_i	Manufacturing completion time of production order $i \in I$
um_i^d	Manufacturing sequence of production order $i \in ID$ in factory $d \in F$
t_i	Tardiness of production order $i \in I$

We describe the MILP model with objective function and constraints as follows:

$$Maximize \sum_{i \in I} z_i \times R_i - \alpha \times \sum_{i \in I} t_i \times R_i \tag{1}$$

$$\sum\nolimits_{d \in DF} y_i^d = z_i \qquad \forall i \in I \tag{2}$$

$$x_{ii}^d = 0 \qquad \forall d \in F, i \in I \tag{3}$$

$$\sum\nolimits_{\substack{j \in ID \\ i \neq j}} x_{ij}^d = y_i^d \qquad \forall d \in F, i \in I \tag{4}$$

$$\sum_{\substack{j \in ID \\ i \neq j}} x_{ji}^{d} = y_i^d \qquad \forall d \in F, i \in I \tag{5}$$

$$\sum_{j \in ID} x_{ij}^{d} \leq 1 \qquad \forall d \in F, i \in ID \tag{6}$$

$$\sum_{j \in ID} x_{ji}^{d} \leq 1 \qquad \forall d \in F, i \in ID \tag{7}$$

$$um_0^d = 0 \qquad \forall d \in F \tag{8}$$

$$um_i^d \leq M \times y_i^d \qquad \forall d \in F, i \in I \tag{9}$$

$$um_i^d \leq M \times \sum_{j \in I} y_j^d \qquad \forall d \in F, i \in I \tag{10}$$

$$um_i^d + x_{ij}^d \leq um_j^d + M \times \left(1 - x_{ij}^d\right) \qquad \forall d \in F, i \in ID, j \in I : i \neq j \tag{11}$$

$$st_{si}^d \leq M \times y_i^d \qquad \forall d \in F, s \in S, i \in I \tag{12}$$

$$st_{si}^d + PT_{si}^d \leq ct_{si}^d + M \times \left(1 - y_i^d\right) \qquad \forall d \in F, s \in S, i \in I \tag{13}$$

$$ct_{si}^d \leq st_{s+1,i}^d + M \times \left(1 - y_i^d\right) \qquad \forall d \in F, s \in S : \mathrm{s} \neq n_s, i \in I \tag{14}$$

$$ct_{si}^d + CO_{ij}^{ds} \leq st_{sj}^d + M \times \left(1 - x_{ij}^d\right) \qquad \forall d \in F, s \in S, i, j \in I : i \neq j \tag{15}$$

$$CO_{0i}^{d1} \leq st_{1i}^d + M \times \left(1 - x_{0i}^d\right) \qquad \forall d \in F, i \in I \tag{16}$$

$$ct_{ti}^d \leq c_i + M \times \left(1 - y_i^d\right) \qquad \forall d \in F, s \in S : \mathrm{s} = n_s, i \in I \tag{17}$$

$$c_i - D_i \leq t_i + M \times (1 - z_i) \qquad \forall i \in I \tag{18}$$

$$z_i, y_i^d \in \{0, 1\} \qquad \forall i \in I, d \in F \tag{19}$$

$$x_{ij}^d \in \{0, 1\} \qquad \forall i, j \in ID, d \in F \tag{20}$$

$$st_{ti}^d, ct_{ti}^d, c_i, t_i \geq 0 \qquad \forall i \in I, s \in S, d \in F \tag{21}$$

$$um_i^d \geq 0 \qquad \forall i \in I, d \in F \tag{22}$$

The objective function, constraint (1), is maximizing the total profit consisting of total revenue from the selected orders and the sum of order tardiness costs, where each order tardiness cost is obtained by multiplying the related revenue and tardiness. To adjust the sum of order tardiness costs, we use a scaling parameter α. The motivation of this objective function is adopted from [31] to observe the variation of the rejection ratios and sub-objective function values. We adopt this objective function to observe the relationship among order acceptance, revenue, and order tardiness cost. The order selection and factory assignment of the selected order are determined by constraint (2). Constraints (3)–(11) determine the permutation manufacturing sequence within each factory. Constraints (3)–(5) determine the immediate production sequence between orders $i, j \in I$. If y_i^d equals 1, in the distributed factory d, there must be a sequence from order i to any order j or the dummy order. Constraints (6)–(11) prevent sub-tour sequences and infeasible tours. Constraints (12)–(16) determine the start and completion times of order $i \in I$ at stage $s \in S$ in distributed factory $d \in F$. Constraints (17) and (18) calculate the manufacturing completion time and tardiness of the order $i \in I$. Constraints (19) and (20) mean that z_i, y_i^d, and x_{ij}^d are binary variables. Constraints (21) and (22) mean that st_{ti}^d, ct_{ti}^d, c_i, t_i, and um_i^d are non-negative variables.

4. Meta-Heuristic Algorithms

The addressed problem is an extension of the flowshop scheduling problem. Since the flowshop scheduling problem is NP-hard [32], the addressed problem is also NP-hard.

Since the addressed problem is NP-hard, the MILP model does not find an optimal solution for the large-sized instances within a limited time. Thus, to efficiently and effectively find a near-optimal solution, we propose a genetic algorithm (GA), which shows good performance in various scheduling problems [31,33]. In Section 4.1, we describe the solution structure and decoding process and the procedure of the proposed GA and PSO. In Sections 4.2 and 4.3, we propose the entire procedure of the GA and PSO with the solution structure and decoding procedure described in Section 4.1, respectively.

4.1. Solution Structure and Decoding Process

We propose a GA and PSO with two one-dimensional lists as a solution structure. Each list is sized with n_I and represented by integer $U\{1,\ n_F\}$ and random-key $U(0,\ 1)$, respectively. List 1 indicates the order selection and factory assignment of each order $i \in I$. List 2 indicates the permutation manufacturing sequence of each order $i \in I$ within each factory $f \in F$. We determine the order selection and factory assignment by using the integer of List 1. The integer 0 means that we reject an order. On the other hand, the integer $1 \leq x \leq n_F$ means that we select the order and assign it to Factory x. We determine the permutation sequence of the selected orders within each factory by sorting List 2 in increasing order.

We illustrate the decoding process in Figure 1 with a toy example. In Figure 1, there are eight orders and two heterogeneous flowshops with three stages. The revenues and due dates of the orders are (20, 24, 30, 17, 28, 15, 29, 18) and (21, 25, 19, 13, 16, 13, 12, 18), respectively. In addition, the weight of the tardiness cost, α, is 0.01. In Figure 2, we illustrate an encoded solution structure. Lists 1 and 2 are [1, 2, 2, 0, 2, 0, 1, 0] and [0.87, 0.73, 0.67, 0.02, 0.13, 0.77, 0.63, 0.25], respectively. In Figure 1b, we show the result of the order selection and factory assignment. From List 1, Orders 4, 6, and 8 are rejected. Orders 1 and 8 are assigned to Factory 1, and Orders 2, 3, and 5 are assigned to Factory 2.

We determine the manufacturing sequence of each factory by sorting List 2 in increasing order. Factories 1 and 2 produce the assigned orders with the related sequence (Order 8 $\rightarrow$ Order 1) and (Order 5 $\rightarrow$ Order 3 $\rightarrow$ Order 2), respectively. The permutation sequence of Factory i is represented as π_i, and π_1 and π_2 correspond to $\pi_1 = \{8,\ 1\}$ and $\pi_2 = \{5,\ 3,\ 2\}$, respectively.

We decide the start and completion times of the selected order i at stage s in each factory f, st_{ti}^{f} and ct_{ti}^{f}, without delay to minimize the manufacturing completion time C_i and tardiness T_i of the selected order i. In Figure 1c, we illustrate the Gantt chart of each factory and present the tardiness of the selected orders. The manufacturing completion time and tardiness of Orders 1, 2, 3, 5, and 8 are (23, 27, 18, 11, 15) and (2, 2, 0, 0, 0), respectively. Thus, we are able to derive the objective value as 119.12 by following constraint (1).

Order	1	2	3	4	5	6	7	8
List 1	1	2	2	0	2	0	0	1

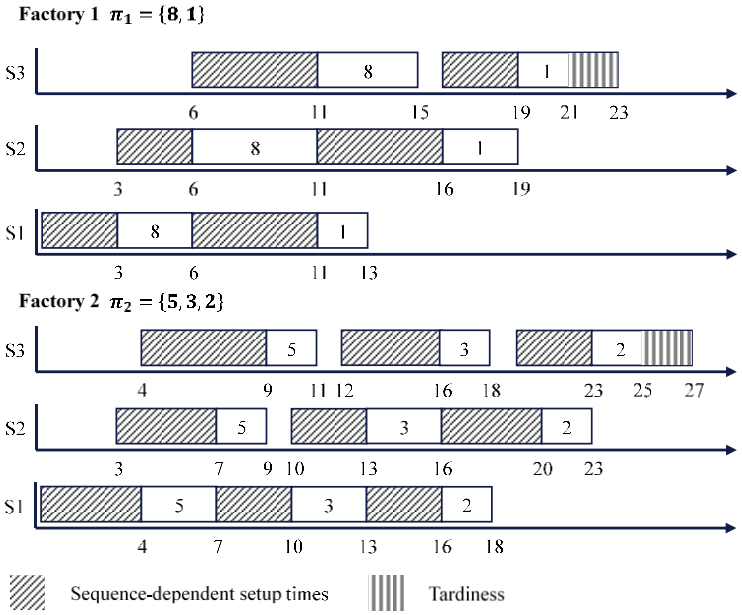
Order	1	2	3	4	5	6	7	8
List 2	0.87	0.73	0.67	0.02	0.13	0.77	0.63	0.25

(**a**) Encoded solution structure

Figure 1. *Cont.*

Rejected Orders	4, 6, 7
Factory 1	1, 8
Factory 2	2, 3, 5

(**b**) Result of the order selection and factory assignment.



(**c**) Gantt-chart of each factory.

Figure 1. An Example of solution representations, decoding procedures, and scheduling results for the proposed GA and PSO ((**a**) Encoded solution structure; (**b**) Result of the order selection and factory assignment; (**c**) Gantt-chart of each factory; S1, S2, and S3 in subfigure (**c**) means the stage 1, 2, and 3, respectively; The numbers in the boxes in subfigure (**c**) indicate processing time).

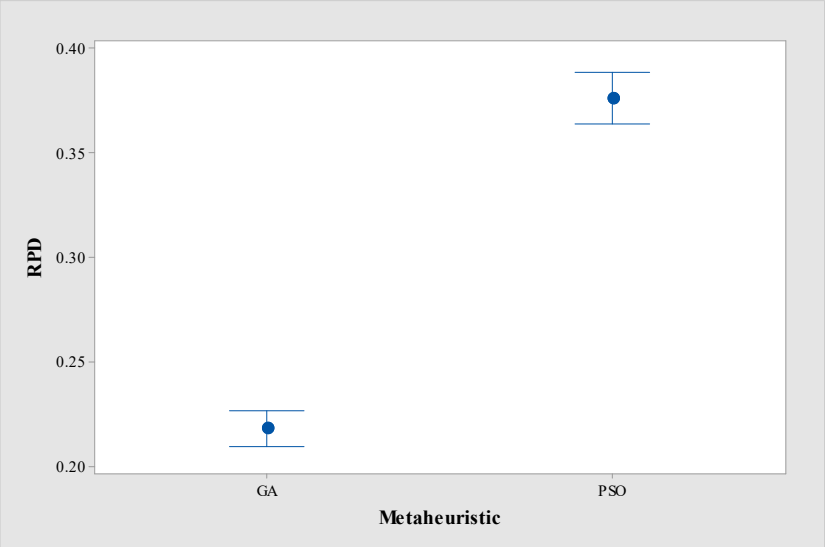


Figure 2. Interval graph showing statistically significant differences in RPD values of GA and PSO (blue dots means the average value).

4.2. Genetic Algorithm (GA)

For the proposed GA, we define a population as a set of chromosomes. The population is denoted as $P = \left[c_{[1]}, c_{[2]}, \ldots, c_{[n_P]}\right]$, where $c_{[i]}$ and n_P indicate ith chromosome and population size, respectively. We compose the proposed GA with Initialization, Evaluation,

Selection, Crossover, and Mutation. The GA randomly generates an initial population by following the probabilistic distributions of Lists 1 and 2 in the Initialization. The Initialization is represented as the operator Initialization() in the pseudo code. In the Evaluation, the GA decodes each chromosome $c_{[i]}$ and evaluates its fitness $f(c_{[i]})$ as the total profit derived by decoding ith chromosome by following Section 4.1. The operator *Evaluate* $(c_{[i]})$ represents the evaluation of the chromosome $c_{[i]}$ to measure its fitness. In the *Selection*, with a roulette wheel method, the GA selects the next population from the current population. It is represented as the operator $Selection(P)$. To swap the chromosomes in the population, the GA adopts one cutting point crossover in *Crossover* $(c_{[i]}, c_{[j]})$. The operator *Crossover* $(c_{[i]}, c_{[j]})$ means that the chromosomes $c_{[i]}$ and $c_{[j]}$ are swapped by the one cutting point crossover. To tweak the chromosome in the parent population, the GA adopts the uniform mutation in *Mutation* $(c_{[i]})$. The operator *Mutation* $(c_{[i]})$ means the chromosome $c_{[i]}$ is tweaked by the uniform mutation. Each List in the chromosome is independently swapped and tweaked in the crossover and mutation processes. We describe the procedure of the GA in Algorithm 1.

Algorithm 1: Genetic Algorithm
Input : Generation size $Gmax$, Population size n_P, Crossover rate Cr, Mutation rate Mr

Initialization ()
$g \leftarrow 1$
While $g \leq Gmax$
  $i \leftarrow 1$
  While $i \leq n_P$
    $f\left(c_{[i]}\right) \leftarrow Evaluate\left(c_{[i]}\right)$ *//Calculate the objective function*
    $i \leftarrow i + 1$
  End While
  $P \leftarrow Selection(P)$ *//Conduct the selection procedure*
  $i \leftarrow 1$
  While $i \leq n_P$
    $r1 \leftarrow U(0, 1)$
    If $r1 \leq Cr$
      Randomly select chromosome $c_{[x]}$ from P
      $c_{[i]} \leftarrow Crossover\left(c_{[i]}, c_{[x]}\right)$ *//Conduct the crossover operator*
    End If
    $r2 \leftarrow U(0, 1)$
    If $r2 \leq Mr$
      $c_{[i]} \leftarrow Mutation\left(c_{[i]}\right)$ *//Conduct the mutation operator*
    End If
    $i \leftarrow i + 1$
  End While
  $g \leftarrow g + 1$
End While

4.3. Particle Swarm Optimization (PSO)

For the proposed PSO, we define a swarm (population) as a set of particles. The swarm at iteration t denote as $S^t = \left[p_1^t, p_2^t, \ldots, p_{n_S}^t\right]$, where p_i^t and n_S indicate particle i and swarm size, respectively. We compose the procedures of the proposed PSO as Initialization, Evaluation, and Update. We apply the identical Initialization and Evaluation processes of

the GA into the proposed PSO. In the Update, there are two sub-update processes, which are (i) updating the personal and global best position vectors of each list and (ii) updating the position and velocity vectors of each list. The personal best position vector PB_i^t and global best vector GB^t mean the historical best position vectors derived by individual particle i and every particle at iteration t. Based on the fitness of each particle i, $f(p_i^t)$, PSO updates personal and global best vectors as follows:

$$PB_i^{t+1} \leftarrow \begin{cases} X_i^t, & f(p_i^t) > f(PB_i^t) \\ PB_i^t, & Otherwise \end{cases} \tag{23}$$

$$GB^{t+1} \leftarrow \begin{cases} argmax X_i^t, & max f(p_i^t) > f(GB^t) \\ \quad f(p_i^t) & \\ GB^t, & Otherwise \end{cases} \tag{24}$$

where X_i^t means the position vector of the particle i at iteration t. The update of the personal and global best position vectors PB_i^{t+1} and GB^{t+1} are represented as the operator $Update\ PB\ (PB_i^t)$ and $Update\ GB\ (GB^t)$, respectively. The position and velocity vectors of each list are independently updated updating position and velocity vectors. Since List 1 is represented by integer $U\{1,\ n_F\}$, then, PSO updates the position vector X_i^t of List 1 by following the update equation of the discrete PSO (DPSO) [33]. The equation is as follows:

$$X_i^{t+1} \leftarrow c_2 \bigotimes F_3(c_1 \bigotimes F_2(w \bigotimes F_1(X_i^t), PB_i^t), GB^t) \tag{25}$$

where F_1 represents mutation operator, and F_2 and F_3 are crossover operators. In the proposed PSO, we adopt F_1 as the operator $Mutation\ (X_i^t)$ used in the proposed GA, and a vector λ_i^{t+1} is derived by $w \otimes F_1(X_i^t)$. The equation of $w \otimes F_1(X_i^t)$ is as follows:

$$\lambda_i^{t+1} \leftarrow \begin{cases} Mutation(X_i^t), & r1 < w \\ X_i^t, & Otherwise \end{cases} \tag{26}$$

where $r1$ and w are the random value $U(0,\ 1)$ and the algorithm parameter, weight, respectively. We adopt F_2 as the operator $Crossover\ (\lambda_i^{t+1}, PB_i^{t+1})$ used in the proposed GA, and a vector δ_i^{t+1} is derived by $c_1 \otimes F_2(\lambda_i^{t+1})$. The equation of $c_1 \otimes F_2(\lambda_i^{t+1})$ is as follows:

$$\delta_i^{t+1} \leftarrow \begin{cases} Crossover(\lambda_i^{t+1}, PB_i^{t+1}), & r2 < c_1 \\ \lambda_i^{t+1}, & Otherwise \end{cases} \tag{27}$$

where $r2$ and c_1 are the random value $U(0,\ 1)$ and the algorithm parameter, cognitive coefficient, respectively. We adopt F_3 as the operator $Crossover\ (\delta_i^{t+1}, GB^{t+1})$ used in the proposed GA, and a vector X_i^{t+1} is derived by $c_2 \otimes F_3(\delta_i^{t+1})$. The equation of $c_1 \otimes F_2(\lambda_i^{t+1})$ is as follows:

$$X_i^{t+1} \leftarrow \begin{cases} Crossover\left(\delta_i^{t+1}, GB^{t+1}\right), & r3 < c_2 \\ GB^{t+1}, & Otherwise \end{cases} \tag{28}$$

where $r3$ and c_2 are the random value $U(0,\ 1)$ and the algorithm parameter, social coefficient, respectively.

Since List 2 is represented by random-key $U(0,\ 1)$, Then, the proposed PSO updates the position vector X_i^t of List 2 by following the update equation of the standard SPSO (SPSO) [30]. The position and velocity vectors X_i^t and V_i^t of List 2 are updated by Equations (29) and (30).

$$V_i^{t+1} \leftarrow w \times V_i^t + c_1 \times U(0,1) \times \left(PB_i^t - X_i^t\right) + c_2 \times U(0,1) \times \left(GB^t - X_i^t\right) \tag{29}$$

$$X_i^{t+1} \leftarrow X_i^t + V_i^{t+1} \tag{30}$$

where w, c_1, and c_2 are the algorithm parameters of the proposed PSO. We describe the procedure of PSO in Algorithm 2.

Algorithm 2: The procedure of PSO
Input : Iteration size $Imax$, Swarm size n_S, PSO parameter w, c_1, c_2

Initialization ()
For $1 \leq t \leq Imax$
  For $1 \leq i \leq n_S$
    $f(p_i^t) \leftarrow Evaluate\ (p_i^t)$ *//Calculate the objective function*
    $PB_i^{t+1} \leftarrow Update\ PB\ (PB_i^t)$ *//Update the particle best*
  End For
  $GB^{t+1} \leftarrow Update\ GB\ (GB^t)$ *//Update the global best*
  For $1 \leq i \leq n_S$
    Update the position vector X_i^t of List 1 by following Equation (25)
    Update the velocity vector V_i^t of List 2 by following Equation (29)
    Update the position vector X_i^t of List 1 by following Equation (30)
  End For
End For

5. Computational Experiments

In this section, we evaluate and verify the performances of the GA through computational experiments. To generate experimental instances, we refer to several testbeds [29].

5.1. Design of Experiments

The complexity of the addressed problem is impacted by the number of factories n_F, stages n_S, and orders n_I. Hence, we generate experimental small- and large-sized instances by referring to TB 1 [29]. For the small-sized experiments, we set n_F, n_S, and n_I as {3, 4}, {3, 4}, and {8, 10, 12}, respectively. For the large-sized experiments, we set n_F, n_S, and n_I as {5, 6, 7}, {5, 10, 15}, and {100, 150, 200}, respectively. We generate revenue R_i by following the discrete uniform distribution $unif\{15, 30\}$. We also randomly generate sequence-dependent setup times CO_{ij}^{fs} and processing times PT_{si}^{f} by following the discrete uniform distribution $unif\{2, 5\}$. To generate the due date D_i, we refer to TB 3 in [29]. The equation for D_i is as follows:

$$D_i = \left(n_S \times med\left(CO_{ij}^{fs}\right) + med\left(\sum_{s \in S} PT_{si}^{f}\right)\right) \times \left(1 + \frac{r}{2n_F}\right) \tag{31}$$

where $med(x)$ and r mean the median of x and random value $U(0,\ 1)$, respectively.

In the small-sized experiments, to verify the absolute performances of the GA, we compare it with the particle swarm optimization (PSO) and the MILP model developed in Section 3. Since the MILP model is unable to find the optimal solution in a limited computing time in large-sized experiments, we relatively compare the GA with PSO, which showed a good performance in the distributed flowshop scheduling problem [29,30]. In this experiment, to verify the performance of the GA and PSO, we compare them with a base heuristic (BH). The pseudo-code of the BH is described in Appendix A.

We verify and evaluate the performance by using a scaling measure, relative percentage deviation (RPD), which is calculated as follows:

$$RPD(\%) = \frac{Z_{best} - Z}{Z_{best}} \times 100\% \quad (32)$$

where Z_{best} means the best total profit among solutions obtained by the GA, PSO, MILP model, or BH, and Z means the total profit obtained by a current heuristic or model.

In the small-sized experiment, we adopt the parameters of the GA and PSO as $(Gmax, n_P, Cr, Mr)$ and $(Imax, n_S, w, c_1, c_2)$ as (500, 50, 0.32, 0.15) and (500, 50, 0.40, 0.35, 0.25), respectively. In large-sized experiments, we adopt the parameters of the GA and PSO as $(Gmax, n_P, Cr, Mr)$ and $(Imax, n_S, w, c_1, c_2)$ as (1000, 50, 0.32, 0.15) and (1000, 50, 0.40, 0.35, 0.25), respectively. For each instance, we repeated the GA and PSO 30 times each and compared the average RPD. The GA and PSO were tested under optimal parameters through pre-repetitive experiments. We implement the BH, GA, and PSO by Python 3.11, and MILP is implemented by CPLEX 22.1.1 on a 2.90 GHz Intel Core i7-10700F CPU. We describe the design of experiments in Table 2.

Table 2. Design of experiments.

	Small-Sized Instances	**Large-Sized Instances**
n_F	3, 4	5, 6, 7
n_S	3, 4	5, 10, 15
n_I	8, 10, 12	100, 150, 200
R_i	$unif\{15, 30\}$ (1000 $)	
D_i	$\left(n_S \times med\left(CO_{ij}^{fs}\right) + med\left(\sum_{s \in S} PT_{si}^{f}\right)\right) \times \left(1 + \frac{r}{2n_F}\right)$ (min)	
PT_{sj}^{f}	$unif\{2, 5\}$ (min)	
CO_{ij}^{fs}	$unif\{2, 5\}$ (min)	
α	0.01	0.001
Fitness Measure	$RPD = \frac{Z_{best} - Z}{Z_{best}} \times 100\%$	

5.2. Results of Experiments

For the small-sized experiments, we define the maximum computational time (CPU) of the MILP model as 1800 s. We compare the MILP model with the proposed GA and PSO described in Section 4 by using the RPD measure. We summarize the results of the small-sized experiment in Table 3. We describe the best solution among the obtained solutions in Z_{best} for each instance. We mark Z_{best} as a bold font if the CPLEX solver finds the optimal solution within the maximum CPU. Otherwise, we describe the RPD value derived by the best feasible solution derived by the CPLEX solver and 1800++ in RPD and CPU, respectively. For the GA and PSO, we describe the median of RPD and CPU in each instance in Table 3. The GA shows the average of RPD and CPU as 0.58 and 3.20, respectively. PSO shows the average of RPD and CPU as 1.08 and 2.52, respectively.

For the large-sized experiments, we compare the proposed GA and PSO with the BH described in Appendix A. In Table 4, we describe the average of RPD and CPU in each large-sized experiment. The BH shows the average RPD as 57.23. The GA shows the average of RPD and CPU as 0.22 and 149.35, respectively. PSO shows the average of RPD and CPU as 0.38 and 131.87, respectively. The GA has a lower RPD than the BH and PSO.

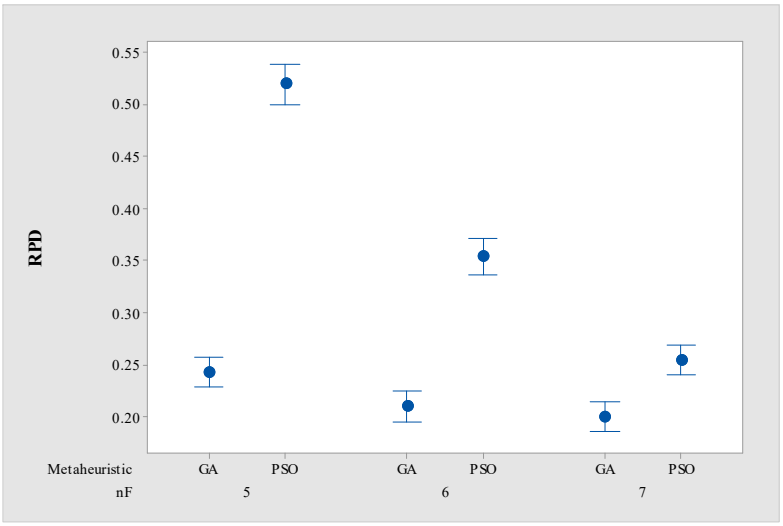
Table 3. Result of small-sized experiments.

Ins.	n_F	n_S	n_I	Z_{best}	MILP		PSO		GA	
					RPD	CPU	RPD	CPU	RPD	CPU
1	3	3	8	173.00	173.00	0.38	0.30	2.01	0.00	2.57
2			10	219.40	219.40	81.89	1.99	2.26	0.54	2.97
3			12	228.10	228.10	1800++	4.54	2.32	2.63	2.98
4		4	8	181.00	181.00	0.19	0.00	2.30	0.00	2.92
5			10	239.00	239.00	0.38	0.00	2.68	0.00	3.35
6			12	265.00	265.00	65.61	2.83	2.90	1.87	3.60
7	4	3	8	175.00	175.00	0.47	0.00	2.09	0.00	2.74
8			10	232.00	232.00	2.95	0.00	2.41	0.00	3.11
9			12	266.80	266.80	1800++	3.29	2.72	1.95	3.42
10		4	8	184.00	184.00	0.23	0.00	2.45	0.00	3.13
11			10	228.00	228.00	0.45	0.00	2.85	0.00	3.57
12			12	302.00	302.00	4.34	0.00	3.22	0.00	4.03
				Mean	224.44	313.09	1.08	2.52	0.58	3.20

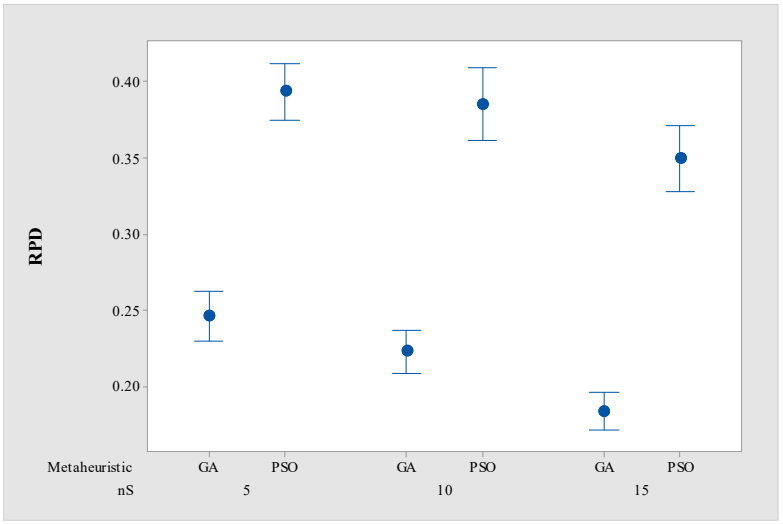
Table 4. Result of large-sized experiment.

Ins.	n_F	n_S	n_I	Z_{best}	BH		PSO		GA	
					RPD	CPU	RPD	CPU	RPD	CPU
1	5	5	100	2085.28	58.54	<0.01	0.43	51.16	0.18	60.14
2			150	3117.11	73.51	<0.01	0.50	76.09	0.24	88.90
3			200	4089.50	79.56	<0.01	0.52	100.32	0.31	117.30
4		10	100	2141.00	52.19	<0.01	0.50	88.11	0.25	99.32
5			150	3199.08	66.30	<0.01	0.52	130.91	0.22	147.78
6			200	4057.96	75.97	<0.01	0.67	173.42	0.34	195.25
7		15	100	2110.56	41.32	<0.01	0.47	124.58	0.20	138.82
8			150	3247.79	63.01	<0.01	0.56	184.35	0.25	206.29
9			200	4158.53	71.23	<0.01	0.51	246.16	0.21	273.49
10	6	5	100	2199.86	49.42	<0.01	0.33	52.12	0.19	61.29
11			150	3094.24	67.66	<0.01	0.45	77.01	0.24	90.54
12			200	4158.09	76.40	<0.01	0.41	101.81	0.29	119.55
13		10	100	2134.06	38.61	<0.01	0.30	88.97	0.13	100.68
14			150	3265.71	60.56	<0.01	0.34	132.08	0.20	149.32
15			200	4132.82	70.42	<0.01	0.41	175.08	0.29	197.79
16		15	100	2285.39	30.82	<0.01	0.25	124.52	0.16	140.00
17			150	3263.94	54.20	<0.01	0.27	186.84	0.10	208.31
18			200	4289.71	65.40	<0.01	0.43	247.83	0.28	276.32
19	7	5	100	2246.76	39.55	<0.01	0.19	52.83	0.10	62.43
20			150	3267.91	61.30	<0.01	0.32	78.12	0.27	92.08
21			200	4233.85	72.09	<0.01	0.39	103.20	0.40	122.14
22		10	100	2301.37	28.26	<0.01	0.23	89.61	0.17	101.81
23			150	3244.54	54.28	<0.01	0.18	133.15	0.17	151.24
24			200	4229.25	67.08	<0.01	0.33	176.52	0.24	200.70
25		15	100	2230.94	22.31	<0.01	0.18	126.08	0.12	141.21
26			150	3301.48	45.13	<0.01	0.24	186.80	0.17	210.39
27			200	4261.60	60.05	<0.01	0.24	252.75	0.16	279.27
				Mean	3198.09	57.23	<0.01	0.38	131.87	0.22	149.35

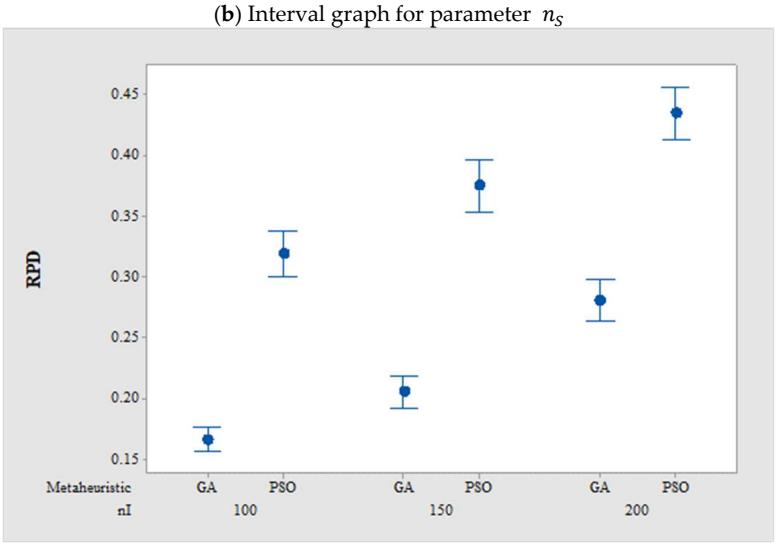
To compare the performance between the GA and PSO, we present confidence interval graphs in Figures 2 and 3. In Figure 2, we show interval graphs for the results of the large-sized experiment with a 0.05 confidence level. As shown in Figure 2, the confidence intervals of the GA and PSO do not overlap. Furthermore, the GA shows a lower mean RPD value and a narrower confidence interval than PSO. We present confidence interval graphs for experimental parameters n_F, n_S, and n_I in Figure 3. For graphs in Figure 3a–c, the GA shows smaller mean RPD values and confidence intervals than PSO even when parameters change. Figures 2 and 3 indicate that the GA has a statistically significant difference in performance compared to PSO. In addition, the GA and PSO show larger mean RPD values and confidence intervals as n_I increases in Figure 3c. It implies that n_I is the experimental parameter that most affects the complexity of the addressed problem.



(**a**) Interval graph for parameter n_F



(**b**) Interval graph for parameter n_S

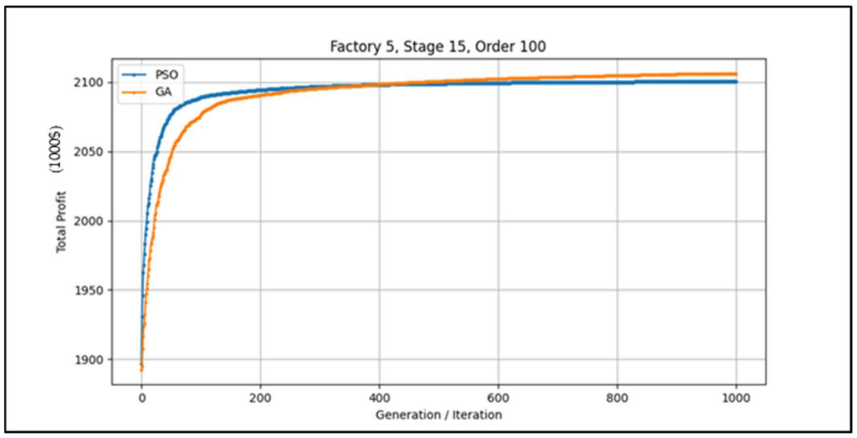


(**c**) Interval graph for parameter n_I

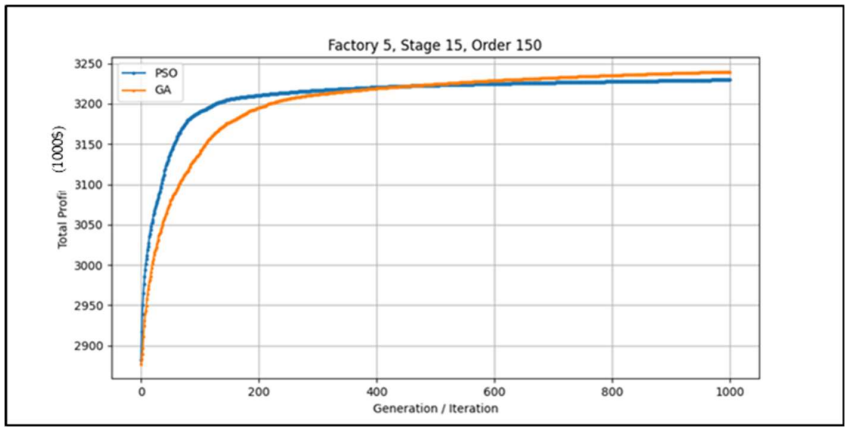
Figure 3. Interval graph showing statistically significant differences in the RPD values of the GA and PSO for parameters n_F, n_S, and n_I ((**a**) Interval graph for parameter n_F; (**b**) Interval graph for parameter n_S; (**c**) Interval graph for parameter n_I; Blue dots means the average value).

To analyze the convergence of the GA and PSO based on changes in n_I, we address average convergence graphs in Figure 4. We present the convergence graphs when n_F and n_S are 5 and

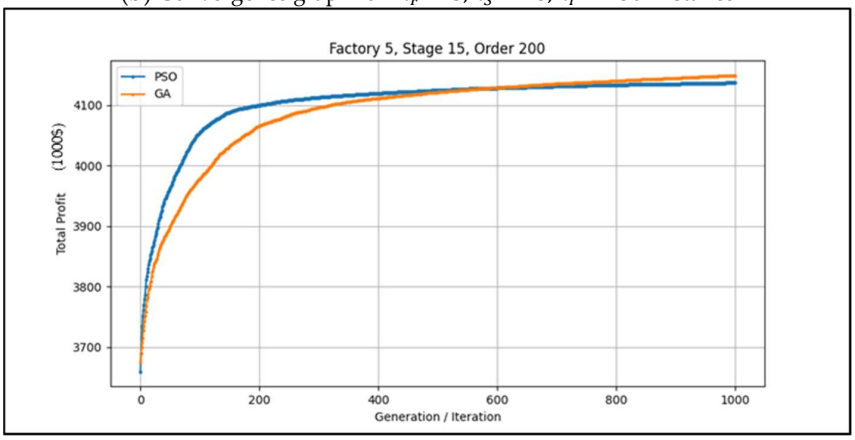
15, respectively. As shown in graphs (a)–(c) of Figure 4, PSO converges faster than the GA in the early stage of the search. However, the GA ultimately outperforms PSO through the continuous improvement of the solution. It implies that genetic and evolutionary operators of the enable the search for a more spacious solution space and continuously improve the solution. In contrast, PSO appears to converge quickly, potentially getting trapped in a local optimum. In addition, the intersection points of the GA and PSO are right-shifted with increasing n_I. This indicates that increasing n_I requires more search steps to sufficiently converge the performance of the GA and PSO. From the results of the large-sized experiment, Figures 2–4, the GA statistically outperforms PSO.



(**a**) Convergence graph for $n_F = 5, n_S = 15, n_I = 100$ instance



(**b**) Convergence graph for $n_F = 5, n_S = 15, n_I = 150$ instance



(**c**) Convergence graph for $n_F = 5, n_S = 15, n_I = 200$ instance

Figure 4. Convergence graph for instance.

5.3. Results of Sensitivity Analysis

To observe the relationship among order selection, revenue, and order tardiness cost, we conduct sensitivity analysis with the GA. We re-test the large-sized experiments described in Section 5.1 and define the scaling parameter of tardiness costs, α, as 0.01. To confirm the reason for the order selection, we analyze a relationship between the slack and revenue of the order. We calculate the slack of each order i, SL_i, and the relative gap of the SL_i, GAP_{SL_i} as follows:

$$SL_i = max\left(D_i - med\left(\sum_{s \in S} PT_{si}^f\right), 0\right) \tag{33}$$

$$GAP_{SL_i}(\%) = \frac{\underset{j \in I}{maxSL_j} - SL_i}{\underset{j \in I}{maxSL_j}} \times 100 \tag{34}$$

Figure 5 shows the interval graphs of GAP_{SL_i} according to the revenue of the order i. The blue and red graphs indicate the interval graphs for the rejected and selected orders, respectively. For each revenue value, we are able to observe that there is no overlap between the blue and red graphs, and the red graph shows a lower mean value than the blue graph. It indicates that the orders with sufficient slacks are statistically significantly selected regardless of the revenue of the order to reduce the potential order tardiness costs.

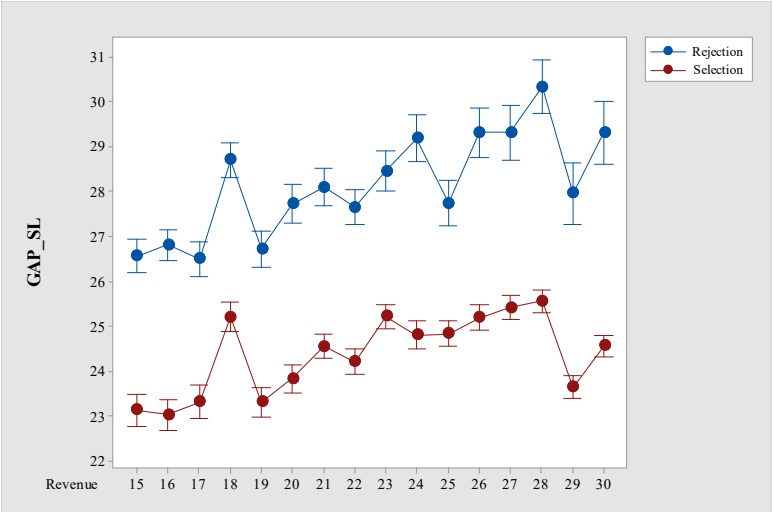


Figure 5. Interval graph for GAP_{SL_i}.

To observe the relationship between revenue and order tardiness cost, we analyze the relative gap of the tardiness of selected order i, GAP_{T_i}, and manufacturing completion time of selected order i, GAP_{C_i}, which are calculated as

$$GAP_{T_i}\ (\%) = \frac{\underset{j \in I}{maxT_j} - T_i}{\underset{j \in I}{maxT_j}} \times 100 \tag{35}$$

$$GAP_{C_i}\ (\%) = \frac{\underset{j \in I}{maxC_j} - C_i}{\underset{j \in I}{maxC_j}} \times 100 \tag{36}$$

where C_i and T_i are the manufacturing completion time and tardiness of selected order i.

Figures 6 and 7 show the interval graphs of GAP_{T_i} and GAP_{C_i} according to the revenue of order i, respectively. In both Figures 6 and 7, we are able to observe the increasing mean

values of GAP_{T_i} and GAP_{C_i} as the revenue of the order increases. It indicates that the orders with high revenue are produced before the orders with low revenue are produced to minimize the actual order tardiness costs.

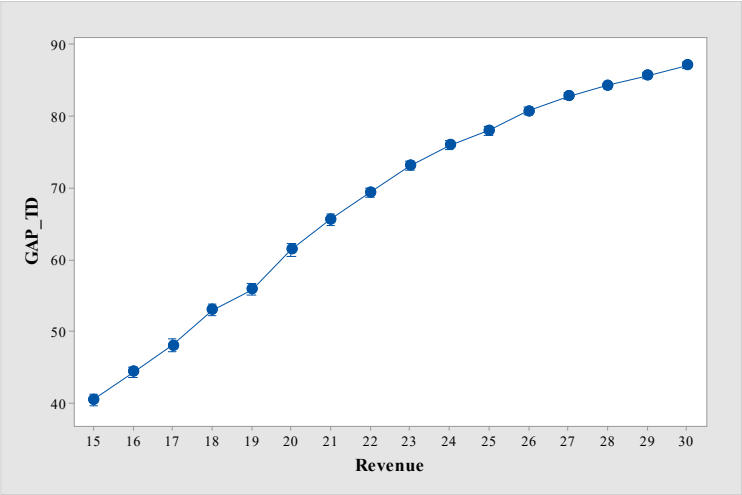


Figure 6. Interval graph for GAP_{T_i} (blue dots means the value of GAP_{T_i}).

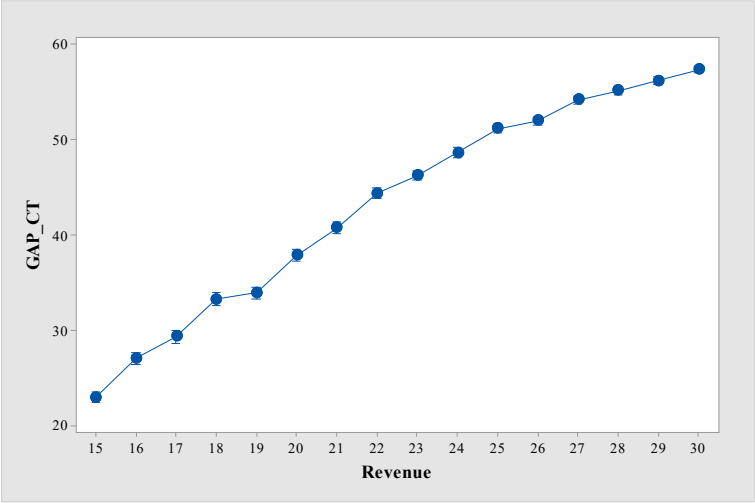


Figure 7. Interval graph for GAP_{C_i} (blue dots means the value of GAP_{C_i}).

Based on the results of the sensitivity analysis, we are able to suggest managerial insights for the selection of production orders and the sequencing of the assigned orders in each factory. To reduce the potential order tardiness costs, we have to determine the order selection by considering orders with sufficient slack among orders with equal revenue. Furthermore, we have to determine the manufacturing sequence by considering the related revenue to minimize the actual order tardiness costs.

6. Conclusions

We addressed the DPFSP-OAHF that firstly considered distributed heterogeneous permutation flowshop, sequence-dependent setup times, and order acceptance strategy, simultaneously. To find an optimal solution, we developed the MILP model of the addressed problem. Furthermore, we proposed the GA to effectively and efficiently find a near-optimal solution. We conducted numerical experiments to verify and evaluate the performances of the GA. In the small-sized experiments, we verified the GA by comparing it with the developed MILP model and PSO. We compared the GA with PSO and the

BH to evaluate the performance of the GA, and the GA showed a statistically significant RPD value in the large-sized experiments. Through the sensitivity analysis with the proposed GA, we suggest managerial insights for the selection of production orders and the sequencing of manufacturing the assigned orders in each factory.

The limitations of this study are as follows: First, the lead time from the factory to the customer is not considered. Second, it is assumed that there is only one flowshop in each factory. In future studies, this study can be extended to multi-factory and multi-flowshop scheduling problems that include the lead time from the factory to the customer. Finally, there are no experiments with real-world data. We will conduct experiments using real-world data in future studies.

Author Contributions: Conceptualization, S.J.L. and B.S.K.; methodology, S.J.L.; software, S.J.L.; validation, S.J.L. and B.S.K.; formal analysis, S.J.L.; investigation, S.J.L.; writing—original draft preparation, S.J.L.; writing—review and editing, S.J.L. and B.S.K.; visualization, S.J.L.; supervision, B.S.K.; funding acquisition, B.S.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Incheon National University Research Grant in 2023 (Grant number: 2023-0181).

Data Availability Statement: The data are available from the corresponding author on reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Base Heuristic

We propose a base heuristic (BH) to efficiently solve the addressed problem. The proposed BH consists of sorting, assigning, and acceptance/rejection phases. In the sorting phase, production orders are sorted in increasing order based on the priority of the order. The priority of order i is equal to R_i/D_i. We calculate the priority by referring to a weighted earliness due date (WEDD) rule that shows good performance to minimize total tardiness in the DPFS and permutation flowshop scheduling problem [33–37]. In the second phase, we virtually assign each sorted order i to each factory f and calculate the tardiness of order i at factory f T_i^f. In the acceptance/rejection phases, we reject an order i if the tardiness cost is higher than its revenue. Otherwise, we assign order i to factory f with minimum T_i^f. Algorithm A1 of the BH is as follows:

Algorithm A1: Base Heuristic
Input : Order list L, Revenue of order i R_i, Due date of order i D_i, Weight α

Sort L in increasing order based on R_i/D_i
For i $\in$ L
  For f $\in$ F
    Let T_i^f as tardiness of order i at factory f
    Virtually assign order i to factory f
    Calculate T_i^f
  End For
  If $R_i \leq \alpha \times R_i \times \min_{1 \leq f \leq n_F} T_i^f$
    Reject order i
  Else
    Assign order i to factory $f^* = \operatorname{argmin}_{1 \leq f \leq n_F} T_i^f$
    Update factory f^*
  End If
End For

References

1. Zhou, L.; Zhang, L.; Laili, Y.; Zhao, C.; Xiao, Y. Multi-task Scheduling of Distributed 3D Printing Services in Cloud Manufacturing. *Int. J. Adv. Manuf. Technol.* **2018**, *96*, 3003–3017. [CrossRef]
2. Leng, J.; Zhong, Y.; Lin, Z.; Xu, K.; Mourtzis, D.; Zhou, X.; Zheng, P.; Liu, Q.; Zhao, J.L.; Shen, W. Towards Resilience in Industry 5.0: A Decentralized Autonomous Manufacturing Paradigm. *J. Manuf. Syst.* **2023**, *71*, 95–114. [CrossRef]
3. Chang, H.C.; Liu, T.K. Optimisation of Distributed Manufacturing Flexible Job Shop Scheduling by Using Hybrid Genetic Algorithms. *J. Intell. Manuf.* **2017**, *28*, 1973–1986. [CrossRef]
4. De Giovanni, L.; Pezzella, F. An Improved Genetic Algorithm for the Distributed and Flexible Job-shop Scheduling Problem. *Eur. J. Oper. Res.* **2010**, *200*, 395–408. [CrossRef]
5. Shao, W.; Pi, D.; Shao, Z. Optimization of Makespan for the Distributed No-Wait Flow Shop Scheduling Problem with Iterated Greedy Algorithms. *Knowl. Based Syst.* **2017**, *137*, 163–181. [CrossRef]
6. Wang, S.Y.; Wang, L.; Liu, M.; Xu, Y. An Effective Estimation of Distribution Algorithm for Solving the Distributed Permutation Flow-Shop Scheduling Problem. *Int. J. Prod. Econ.* **2013**, *145*, 387–396. [CrossRef]
7. Mönch, L.; Shen, L. Parallel Machine Scheduling with the Total Weighted Delivery Time Performance Measure in Distributed Manufacturing. *Comput. Oper. Res.* **2021**, *127*, 105126. [CrossRef]
8. Lei, D.; Yuan, Y.; Cai, J.; Bai, D. An Imperialist Competitive Algorithm with Memory for Distributed Unrelated Parallel Machines Scheduling. *Int. J. Prod. Res.* **2020**, *58*, 597–614. [CrossRef]
9. Behnamian, J.; Fatemi Ghomi, S.M.T. A Survey of Multi-Factory Scheduling. *J. Intell. Manuf.* **2016**, *27*, 231–249. [CrossRef]
10. Bagheri Rad, N.; Behnamian, J. *Recent Trends in Distributed Production Network Scheduling Problem*; Springer: Berlin/Heidelberg, Germany, 2022.
11. Marin-Garcia, J.A.; Garcia-Sabater, J.P.; Miralles, C.; Villalobos, A.R. Profile and Competences of Spanish Industrial Engineers in the European Higher Education Area (EHEA). *J. Ind. Eng. Manag.* **2008**, *1*, 269–284. [CrossRef]
12. Renna, P. Coordination Strategies to Support Distributed Production Planning in Production Networks. *Eur. J. Ind. Eng.* **2015**, *9*, 366–394. [CrossRef]
13. Naderi, B.; Ruiz, R. The Distributed Permutation Flowshop Scheduling Problem. *Comput. Oper. Res.* **2010**, *37*, 754–768. [CrossRef]
14. Slotnick, S.A. Order Acceptance and Scheduling: A Taxonomy and Review. *Eur. J. Oper. Res.* **2011**, *212*, 1–11. [CrossRef]
15. Karabulut, K.; Öztop, H.; Kizilay, D.; Tasgetiren, M.F.; Kandiller, L. An Evolution Strategy Approach for the Distributed Permutation Flowshop Scheduling Problem with Sequence-Dependent Setup Times. *Comput. Oper. Res.* **2022**, *142*, 105733. [CrossRef]
16. Huang, J.P.; Pan, Q.K.; Gao, L. An Effective Iterated Greedy Method for the Distributed Permutation Flowshop Scheduling Problem with Sequence-Dependent Setup Times. *Swarm Evol. Comput.* **2020**, *59*, 100742. [CrossRef]
17. Zhao, F.; Hu, X.; Wang, L.; Li, Z. A Memetic Discrete Differential Evolution Algorithm for the Distributed Permutation Flow Shop Scheduling Problem. *Complex Intell. Syst.* **2022**, *8*, 141–161. [CrossRef]
18. Gogos, C. Solving the Distributed Permutation Flow-Shop Scheduling Problem Using Constrained Programming. *Appl. Sci.* **2023**, *13*, 12562. [CrossRef]
19. Yang, S.; Xu, Z. The Distributed Assembly Permutation Flowshop Scheduling Problem with Flexible Assembly and Batch Delivery. *Int. J. Prod. Res.* **2021**, *59*, 4053–4071. [CrossRef]
20. Huang, Y.Y.; Pan, Q.K.; Gao, L.; Miao, Z.H.; Peng, C. A Two-Phase Evolutionary Algorithm for Multi-Objective Distributed Assembly Permutation Flowshop Scheduling Problem. *Swarm Evol. Comput.* **2022**, *74*, 101128. [CrossRef]
21. Pan, Y.; Gao, K.; Li, Z.; Wu, N. Improved Meta-Heuristics for Solving Distributed Lot-Streaming Permutation Flow Shop Scheduling Problems. *IEEE Trans. Autom. Sci. Eng.* **2023**, *20*, 361–371. [CrossRef]
22. Luo, C.; Gong, W.; Li, R.; Lu, C. Problem-Specific Knowledge MOEA/D for Energy-Efficient Scheduling of Distributed Permutation Flow Shop in Heterogeneous Factories. *Eng. Appl. Artif. Intell.* **2023**, *123*, 106454. [CrossRef]
23. Meng, T.; Pan, Q.K. A Distributed Heterogeneous Permutation Flowshop Scheduling Problem with Lot-Streaming and Carryover Sequence-Dependent Setup Time. *Swarm Evol. Comput.* **2021**, *60*, 100804. [CrossRef]
24. Khare, A.; Agrawal, S. Effective Heuristics and Metaheuristics to Minimise Total Tardiness for the Distributed Permutation Flowshop Scheduling Problem. *Int. J. Prod. Res.* **2021**, *59*, 7266–7282. [CrossRef]
25. Lin, Z.; Jing, X.L.; Jia, B.X. An Iterated Greedy Algorithm for Distributed Flowshops with Tardiness and Rejection Costs to Maximize Total Profit. *Expert Syst. Appl.* **2023**, *233*, 120830. [CrossRef]
26. Zhang, Q.; Zhen, T. Improved Jaya Algorithm for Energy-Efficient Distributed Heterogeneous Permutation Flow Shop Scheduling. *J. Supercomput.* **2025**, *81*, 434. [CrossRef]
27. Song, H.; Li, J.; Du, Z.; Yu, X.; Xu, Y.; Zheng, Z.; Li, J. A Q-learning Driven Multi-Objective Evolutionary Algorithm for Worker Fatigue Dual-Resource-Constrained Distributed Hybrid Flow Shop. *Comput. Oper. Res.* **2025**, *175*, 106919. [CrossRef]
28. Zhu, N.; Zhao, F.; Yu, Y.; Wang, L. A Cooperative Learning-Aware Dynamic Hierarchical Hyper-Heuristic for Distributed Heterogeneous Mixed No-Wait Flow-Shop Scheduling. *Swarm Evol. Comput.* **2024**, *90*, 101668. [CrossRef]

29. Perez-Gonzalez, P.; Framinan, J.M. A Review and Classification on Distributed Permutation Flowshop Scheduling Problems. *Eur. J. Oper. Res.* **2023**, *312*, 1–21. [CrossRef]
30. Mraihi, T.; Driss, O.B.; El-Haouzi, H.B. Distributed Permutation Flow Shop Scheduling Problem with Worker Flexibility: Review, Trends and Model Proposition. *Expert Syst. Appl.* **2023**, *238*, 121947. [CrossRef]
31. Kapadia, M.S.; Uzsoy, R.; Starly, B.; Warsing, D.P. A Genetic Algorithm for Order Acceptance and Scheduling in Additive Manufacturing. *Int. J. Prod. Res.* **2021**, *60*, 1–18. [CrossRef]
32. Kan, A.R. General Flow-Shop and Job-Shop Problems. In *Machine Scheduling Problems*; Springer: Boston, MA, USA, 1976; pp. 106–130.
33. Pan, Q.K.; Tasgetiren, M.F.; Liang, Y.C. A Discrete Particle Swarm Optimization Algorithm for the No-Wait Flowshop Scheduling Problem. *Comput. Oper. Res.* **2008**, *35*, 2807–2839. [CrossRef]
34. Marini, F.; Walczak, B. Particle Swarm Optimization (PSO). A Tutorial. *Chemom. Intell. Lab. Syst.* **2015**, *149*, 153–165. [CrossRef]
35. Cerveira, M.I.F. Heuristics for the Distributed Permutation Flowshop Scheduling Problem with the Weighted Tardiness Objective. 2019. Available online: https://www.proquest.com/openview/109fa27057466f781cfeac17a0b99ee1/1?pq-origsite=gscholar&cbl=2026366&diss=y (accessed on 3 March 2025).
36. Ruiz, R.; Stützle, T. An Iterated Greedy Heuristic for the Sequence Dependent Setup Times Flowshop Problem with Makespan and Weighted Tardiness Objectives. *Eur. J. Oper. Res.* **2008**, *187*, 1143–1159. [CrossRef]
37. Molina-Sánchez, L.P.; González-Neira, E.M. GRASP to Minimize Total Weighted Tardiness in a Permutation Flow Shop Environment. *Int. J. Ind. Eng. Comput.* **2016**, *7*, 161–176. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.