AIMS Mathematics, 10(12): 28524–28555.
DOI: 10.3934/math.20251256
Received: 23 October 2025
Revised: 23 November 2025
Accepted: 26 November 2025
Published: 03 December 2025

https://www.aimspress.com/journal/Math

Research article

An improved iterated greedy algorithm for scheduling distributed permutation flowshop problems with weighted total completion time criterion

Yuan-Zhen Li*, Lei-Lei Meng and Biao Zhang

School of Computer Science, Liaocheng University, Shandong Liaocheng, 252000, China

* **Correspondence:** Email: liyuanzhen@lcu.edu.cn.

Abstract: In this paper, distributed permutation flowshop problems with a weighted total completion time criterion (DPFSP-WTC) were addressed to minimize the completion time of all factories. First, the completion time of all factories were converted to a single one by a novel strategy, and a mixed integer programming model was developed. Second, an improved iterated greedy (IIG) algorithm was proposed. Based on features of the concerned problems, a simple heuristic is designed to improve the quality of initialization solutions. A local search operation was developed to improve the convergence performance of the proposed algorithm. Finally, numerous experiments were carried out for solving 720 instances with different scales. The proposed IIG was compared with five state-of-the-art algorithms. The comparisons and discussions showed that the proposed IIG has superior performance compared to its peers.

Keywords: distributed permutation flowshop scheduling; iterated greedy algorithm; makespan; weighted total completion time
Mathematics Subject Classification: 68T40, 68W20

1. Introduction

With the deepening of economic globalization and increasingly severe market competition, the traditional single flowshop manufacturing mode has been unable to meet the production needs of the manufacturing industry, and the distributed manufacturing mode has become an important tool for

enterprises to improve their production competitiveness [1]. The distributed manufacturing mode can make full use of the production resources of multiple factories (workshops) or machining centers, and realize the rapid production of products at a reasonable cost through the reasonable planning and sharing of resources [2,3]. Distributed shop scheduling problems are based on distributed manufacturing, such as cooperative production among different companies or collaborative production among different factories, to study the allocation of jobs among factories and the processing sequence in each factory, to achieve the optimization of scheduling indicators [4].

In distributed manufacturing, a widely studied problem is the distributed permutation flowshop scheduling problem (DPFSP). In the DPFSP, there are f factories, and each factory contains a flowshop. The existing research on the DPFSP includes multi-objective and single optimization models. The research objectives include completion time [5,6], total flow time [7,8], total tardiness [9], and other indicators.

Indeed, in this paper, another interesting indicator is observed. For instance, a distributed system has three factories. Suppose a scheduling scheme 1, where the completion time of each factory is (166,155,145), and the completion time (makespan) of this scheduling scheme is *max* (166,155,145) = 166. In another scheduling scheme 2, the completion time of each factory is (166,153,144), and the completion time of this scheduling scheme is *max* (166,153,144), which is also equal to 166. If the completion time is used as the evaluation standard, the completion time of the two scheduling schemes is equal. In fact, it can be seen that scheduling scheme 2 is better than the scheduling scheme 1. In two considered scheduling schemes, although the completion time of the first factory is the same, the completion time of the second factory and the third factory in the second scheduling scheme are smaller. The majority of papers in the literature [5,10–16] with a makespan criterion only optimize the factory with the largest completion time, without considering the completion time of other factories. To the best of our knowledge, there is no research on minimizing the completion time of all factories simultaneously.

This paper takes into account the DPFSP as the research object and aims to minimize the completion time of all factories. Indeed, we transformed seemingly a multi-objective problem into a single objective optimization problem using the weighted total completion time criterion (named the DPFSP-WTC), and proposed a mixed-integer linear programming model for the DPFSP-WTC. After studying the characteristics of the DFPSP-WTC, we designed a heuristic algorithm and proposed an improved iterated greedy (IIG) algorithm to solve the DPFSP-WTC. The obtained numerical optimization results verified the good performance of the proposed IIG algorithm compared to other considered optimizers.

The remainder of this paper is organized as follows. Section 2 represents a literature review of related works for single- and multi-objective optimization models for the DPFSP. In Section 3, the DPFSP with makespans criterion for all factories is formulated in detail. The relevant characteristics of the DPFSP-WTC are also described in this section. Section 4 provides the proposed algorithm based on the iterated greedy algorithm in detail. Section 5 describes the parameter settings of the proposed algorithm for optimal solving of the DPFSP. Obtained computational and numerical optimization results and comparisons are reported in Section 6. Finally, Section 7 provides the concluding remarks and suggests some future works.

2. Literature review

The research on the DPFSP can be divided into single-objective and multi-objective optimization

problems given in the following sections. For more detailed information, interested readers can refer to the comprehensive review papers in the literature [17,18].

2.1. Single-objective optimization for the DPFSP

In 2010, Naderi and Ruiz published the first paper on the DPFSP with makespan criterion [5]. Since then, the research of the DPFSP has been in full swing. Many scholars have proposed different methods to study the DPFSP with minimizing the makespan criterion, including the tabu search algorithm [10], hybrid immune algorithm [11], iterated greedy algorithm [12–14], scatter search algorithm [15], novel chemical reaction optimization [16], and so forth. Other optimization indicators of the DPFSP also have been studied. The most important indicators are the total flow time [7,8], cumulative payoffs [19], and cycle time [20].

Some scholars have considered realistic constraints to study the DPFSP, such as distributed no-idle (makespan) [21], distributed mixed no-idle (total flow time) (Li, Pan, and Li et al., 2021), distributed blocking (makespan) [22–24], distributed no-wait (makespan) [25], distributed no-wait with due windows (total weighted earliness and tardiness) [26], customer order constraint (makespan) [27], distributed limited-buffer (makespan) [6], sustainability criteria (makespan) [28], robust optimization (makespan) [29], and sequence-dependent setup times (makespan) [30]. To sum up, as shown in Table 1, we can see that there are many single-objective researches on the distributed flowshop. They mainly focus on the completion time.

Table 1. Literature about single-objective optimization for the DPFSP.

Ref.	Constraints	Objectives	Methods
[5]	-	Makespan	Variable neighborhood descent algorithms
[10]	-	Makespan	Tabu search algorithm
[11]	-	Makespan	Hybrid immune algorithm
[12]	-	Makespan	Iterated greedy
[13]	-	Makespan	Iterated greedy
[14]	-	Makespan	Iterated greedy
[15]	-	Makespan	Scatter search
[16]	-	Makespan	Novel chemical reaction optimization
[7]	-	Total flow time	Iterated local search, artificial bee colony, scatter search, iterated greedy
[8]	-	Total flow time	Evolutionary algorithm
[19]	-	Cumulative payoffs	Iterated greedy
[20]	-	Cycle time	Tabu search algorithm
[21]	No-idle	Makespan	Iterated reference greedy
[22]	Blocking	Makespan	Iterated greedy
[23]	Blocking	Makespan	Differential evolution
[24]	Blocking	Makespan	Evolutionary algorithm
[25]	No-wait	Makespan	Iterated greedy algorithms

Continued on next page

Ref.	Constraints	Objectives	Methods
[26]	No-wait, due windows	Total weighted earliness and tardiness	A fruit fly algorithm
[27]	Customer order constraint	Makespan	Variable neighborhood descent, artificial bee colony, iterated greedy
[28]	Sustainability criteria	Makespan	Simulated annealing and tabu search, adaptive large neighborhood search
[29]	Robust optimization	Makespan	
[6]	Limited-buffer	Makespan	Differential evolution
[30]	Sequence-dependent setup times	Makespan	Evolution strategy
This study	-	Weighted total completion time	Iterated greedy

2.2. Multi-objective optimization for the DPFSP

As multi-objective optimization has become a research hotspot, some researchers began to study multiple objectives of the DPFSP simultaneously. In order to achieve the carbon peak and carbon neutralization, many studies regarded total energy consumption as an important indicator, including the energy-efficient DPFSP (total energy consumption and makespan) [31], green scheduling (total energy and total flowtime) [32], distributed blocking flowshop (total energy consumption and makespan) [33], carbon-efficient scheduling (total carbon emissions and makespan) [34], energy-conscious DPFSP with the total tardiness constraint (total energy consumption and expected makespan) [35], energy-efficient DPFSP with limited buffers (total energy consumption and makespan) [36], and sustainable DPFSP (total energy consumption, makespan, and the social factor) [37].

The multi-objective research on other indicators of the DPFSP include the multi-objective DPFSP (makespan and the total tardiness) [38], distributed heterogeneous manufacturing systems (makespan, total machine load, and the maximum machine load) [39], no-wait DPFSP with sequence-dependent setup time (makespan and maximum tardiness) [40], reentrant DPFSP with sequence-dependent setup time (makespan, tardiness, and production cost) [41], DPFSP with total processing time and makespan [42], disassembly line balancing model [43], and multi-objective distributed fuzzy permutation flowshop problem (fuzzy completion time and agreement index) [44]. The multi-objective research of the DPFSP has been a wide concern for researchers, as shown in Table 2. However, there is no research report on the DPFSP minimizing the completion time of all factories.

Table 2. Literature study about multi-objective optimization for the DPFSP.

Ref.	Constraints	Objectives	Methods
[31]	-	Total energy consumption and makespan	Knowledge-based cooperative algorithm
[32]	-	Total energy and total flowtime	NSGA-II
[33]	Blocking	Total energy consumption and makespan	Pareto-based estimation of distribution algorithm

Continued on next page

Ref.	Constraints	Objectives	Methods
[34]	-	Total carbon emissions and makespan	Competitive memetic algorithm
[35]	Stochastic	Total energy consumption and expected makespan	Brain storm optimization
[36]	Limited buffers	Total energy consumption and makespan	Pareto-based collaborative multi-objective optimization algorithm
[37]	-	Total energy consumption, makespan, and social factor	Novel multi-objective learning-based heuristic
[38]	-	Makespan and the total tardiness	Competitive memetic algorithm
[39]	Heterogeneous manufacturing systems	Makespan, total machine load, and maximum machine load	Multi-objective memetic algorithm
[40]	No-wait, sequence-dependent setup time	Makespan and maximum tardiness	Genetic algorithm, artificial bee colony, migratory bird optimization
[41]	Reentrant, sequence-dependent setup time	Makespan, tardiness, and production cost	Multi-objective adaptive large neighborhood search
[42]	-	Total processing time and makespan	Multi-objective particle swarm optimization, differential evolution
[43]	Disassembly line balancing model	Idle rate, smoothness, labor cost, and energy consumption	Adaptive large neighborhood search
[44]	Fuzzy processing time	Fuzzy completion time and agreement index	Artificial bee colony algorithm

In summary, there have been numerous studies on distributed flowshop scheduling problems, including both single-objective optimization and multi-objective optimization. However, research on minimizing the completion time of all factories has not yet been reported. This paper adopts the weighted total completion time criterion to minimize the completion time of all factories.

3. Problem description

In this section, we introduce the mathematical model of the DPFSP-WTC. The indices, parameters, and decision variables are listed in Table 3. There are some assumptions for the problem at hand as given in the following. There are multi-factories in which all factories are the same and are flowshop. There are m machines in the fixed permutation in each factory. There are n jobs to be processed. All jobs must be processed following the same route of machines. The processing capacity of each factory is the same. That is, p_{ij} of all factories is equal. The parameters n, m, f, and p_{ij} are known in advance. All jobs are ready at the beginning. Each job is assigned to one factory for processing, and

cannot be processed in two or more factories. When a job is being processed in a factory, it cannot be interrupted and preempted. There are no facility malfunctions or maintenance issues. The setup time has been included in the processing time of each job.

Table 3. Indices, parameters, and decision variables used in the DPFSP model.

f	Number of factories
n	Number of jobs needed to be processed
m	Number of machines in each factory
k	Position index
$J = \{J_1, J_2, \ldots, J_n\}$	Set of n jobs to be processed, where J_j is the j^{th} job to be processed, $J_j \in J$
$M = \{M_1, M_2, \ldots, M_m\}$	Set of m machines, where M_i is the i^{th} machine used to process the job, $M_i \in M$
$F = \{F_1, F_2, \ldots, F_f\}$	The set of f factories, where F_l is the l^{th} factory from set F, $F_l \in F$
$O_{i,j}$	Operation of job J_j on machine M_i
p_{ij}	The processing time of job J_j on machine M_i
C_{ikl}	Completion time of the job in position k of machine M_i in shop F_l
C_l	The makespan in the factory l
D	The sum of the processing time for all jobs on all machines, $D = \sum_{ij} P_{ij}$
x_{jkl}	A binary index, which equals one if job J_j is assigned for processing in position k of the shop F_l, and otherwise equals 0

3.1. The mixed-integer linear programming models

Based on the assumptions and defined notations given in Table 3, the mathematical programming model is extended from the models addressed by Naderi and Ruiz [5], which can be formulated as given follows:

$$Minimize \quad WTC = \sum_{l=1}^{f} D^{f-l} \cdot C_l, \tag{1}$$

subject to:

$$\sum_{k=1}^{n} \sum_{l=1}^{f} x_{jkl} = 1, j = 1, \ldots, n, \tag{2}$$

$$\sum_{j=1}^{n} x_{jkl} \leq 1, k = 1, \ldots, n, l = 1, \ldots, f, \tag{3}$$

$$C_{ikl} \geq \sum_{j=1}^{n} x_{jkl} \cdot p_{ij}, i = 1, k = 1, l = 1, \ldots, f, \tag{4}$$

$$C_{ikl} \geq C_{(i-1)kl} + \sum_{j=1}^{n} x_{jkl} \cdot p_{ij}, i = 2, \ldots, m, k = 1, \ldots, n, l = 1, \ldots, f, \tag{5}$$

$$C_{ikl} \geq C_{i(k-1)l} + \sum_{j=1}^{n} x_{jkl} \cdot p_{ij}, i = 1,2, \ldots, m, k = 2, \ldots, n, l = 1, \ldots, f, \tag{6}$$

$$C_l \geq C_{mkl}, k = 1, \ldots n, l = 1, \ldots, f, \tag{7}$$

$$C_l \geq C_{l+1}, l = 1, \ldots, f - 1, \tag{8}$$

$$C_l \geq 0, l = 1, \ldots, f, \tag{9}$$

$$C_l \leq D, l = 1, \ldots, f, \tag{10}$$

$$C_{ikl} \geq 0, k = 1, \ldots n, i = 1, \ldots, m, l = 1, \ldots, f, \tag{11}$$

$$x_{jkl} \in \{0,1\}, j = 1, \ldots n, k = 1, \ldots, n, l = 1, \ldots, f. \tag{12}$$

Equation (1) is the objective function of the considered problem. It will be proved in the following that this single-objective function will optimize the completion time of all factories. Constraint (2) is the restriction that every job must be exactly at one position and only at one factory. Constraint (3) ensures that at most one job is placed in each position. Constraint (4) gives the completion time of the first job on the first machine in factory l. Constraint (5) gives the relationship between the completion time of a job on two adjacent machines, indicating that a job can be processed on this machine only after it is finished on the previous machine. Similarly, constraint (6) defines the relationship among the completion times of adjacent jobs on the same machine, indicating that a job can be processed on a machine only after the previous job is completed on the same machine. Constraint (7) defines the completion time of factory l, that is, the value of C_l. Constraint (8) represents that the completion time of the first factory is the largest, the completion time of the second factory is the second largest, and so forth, and the completion time of the f^{th} factory is the smallest. Constraints (9) and (10) define the value range of C_l. Obviously, C_l is greater than zero. Since $D = \sum_{ij} P_{ij}$, the completion time of all factories cannot be greater than D. Constraints (11) and (12) define the value range of C_{ikl} and x_{jkl}, respectively.

To clarify further, there are several points in the mathematical model stated in Eqs (2) to (12) that need to be explained. Constraint (7) defines the completion time of each factory, which is greater than the completion time of all jobs processed in this factory on the last machine. Constraint (8) defines that the completion time of the first factory is greater than that of the second factory; the completion time of the second factory is greater than that of the third factory; and so forth. Because all factories are homogeneous, constraint (8) is reasonable. Similarly, in the later description, the vector of the completion time of each factory for a scheduling scheme is $(C_1, C_2, \ldots, C_l, C_{l+1}, \ldots, C_f)$, which means $C_1 \geq C_2 \geq \cdots \geq C_l \geq C_{l+1} \geq \cdots \geq C_f$. Our optimization goal is to minimize $(C_1, C_2, \ldots, C_l, C_{l+1}, \ldots, C_f)$, which means minimizing C_1 first, then minimizing C_2 on the basis of minimizing C_1, and so forth.

Theorem 1. Suppose that the completion time of two adjacent factories in scheduling scheme 1 is (C_l^1, C_{l+1}^1), where $C_l^1 \geq C_{l+1}^1$, and the completion time of the corresponding two factories of another scheduling scheme 2 is (C_l^2, C_{l+1}^2), where $C_l^2 \geq C_{l+1}^2$. The completion time of other factories are the same, that is, $C_{l'}^1 = C_{l'}^2, l' \neq l, l+1$. We have the following conclusion: initial cap of the size relationship between C_{l+1}^1 and C_{l+1}^2, $C_l^1 > C_l^2$ and $D^{f-l} \times C_l^1 + D^{f-(l+1)} \times C_{l+1}^1 > D^{f-l} \times C_l^2 + D^{f-(l+1)} \times C_{l+1}^2$ are equivalent, that is, $C_l^1 > C_l^2$ and $\sum_{l=1}^{f} D^{f-l} \cdot C_l^1 > \sum_{l=1}^{f} D^{f-l} \cdot C_l^2$ are equivalent.

Proof. Suppose $C_l^1 = C_l^2 + \Delta, \Delta \geq 1$. $D^{f-l} \times C_l^1 + D^{f-(l+1)} \times C_{l+1}^1 = D^{f-l} \times (C_l^2 + \Delta) + D^{f-(l+1)} \times C_{l+1}^1 = D^{f-l} \times C_l^2 + D^{f-l} \times \Delta + D^{f-(l+1)} \times C_{l+1}^1 \geq D^{f-l} \times C_l^2 + D^{f-l} + D^{f-(l+1)} \times C_{l+1}^1 \geq D^{f-l} \times C_l^2 + D^{f-l} = D^{f-l} \times C_l^2 + D^{f-(l+1)} \times D \geq D^{f-l} \times C_l^2 + D^{f-(l+1)} \times C_{l+1}^2$. Because $C_{l'}^1 = C_{l'}^2, l' \neq l, l+1$, then $D^{f-l'} \times C_{l'}^1 = D^{f-l'} \times C_{l'}^1, l' \neq l, l+1$. Therefore, the following conclusion is drawn: $\sum_{l=1}^{f} D^{f-l} \cdot C_l^1 > \sum_{l=1}^{f} D^{f-l} \cdot C_l^2$. All aforementioned derivations are reversible,

so they are equivalent.

Theorem 1 expresses that as long as $C_l^1 > C_l^2$, scheduling scheme 1 is better than scheduling scheme 2, and the completion time of the subsequent factories has no effect on which solution is better. The completion time of each factory in scheduling scheme 1 is $(C_1^1, C_2^1, \ldots, C_l^1, C_{l+1}^1, \ldots, C_f^1)$, where $C_1^1 \geq C_2^1 \geq \cdots \geq C_l^1 \geq C_{l+1}^1 \geq \cdots \geq C_f^1$. The completion time of each factory in scheduling scheme 2 is $(C_1^2, C_2^2, \ldots, C_l^2, C_{l+1}^2, \ldots, C_f^2)$, where $C_1^2 \geq C_2^2 \geq \cdots \geq C_l^2 \geq C_{l+1}^2 \geq \cdots \geq C_f^2$. To compare scheduling scheme 1 and scheduling scheme 2, first we need to compare C_1^1 and C_1^2. The scheduling scheme with the shortest completion time of the first factory is better. If they are equal, then we compare C_2^1 and C_2^2, and so forth. In this way, the comparison process is consistent with the objective function given in Eq (1).

Theorem 2. The completion time of each factory in scheduling scheme 1 is $(C_1^1, C_2^1, \ldots, C_l^1, C_{l+1}^1, \ldots, C_f^1)$, where $C_1^1 \geq C_2^1 \geq \cdots \geq C_l^1 \geq C_{l+1}^1 \geq \cdots \geq C_f^1$. The completion time of each factory in scheduling scheme 2 is $(C_1^2, C_2^2, \ldots, C_l^2, C_{l+1}^2, \ldots, C_f^2)$, where $C_1^2 \geq C_2^2 \geq \cdots \geq C_l^2 \geq C_{l+1}^2 \geq \cdots \geq C_f^2$. Scheduling scheme 1 is superior to scheduling scheme 2, which is equivalent to Eq (13),

$$\sum_{l=1}^{f} D^{f-l} \cdot C_l^1 < \sum_{l=1}^{f} D^{f-l} \cdot C_l^2. \tag{13}$$

Proof. Theorem 2 can be proved by using Theorem 1 and repeating $f-1$ times under conditions $l = 1$, $l = 2,\ldots, l = f-1$.

Therefore, the mathematical model described by Eqs (1) to (12) is reasonable. Therefore, we transform what seems to be a multi-objective optimization problem into a single-objective optimization problem. In addition, many papers mark the completion time as *C*. Indeed, the optimization goal in this paper is not directly the completion time. In fact, it is named as the *WTC* (weighted total *C*) as shown in Eq (1).

3.2. Characteristic for the insert operation neighborhood

In many papers, the insertion operation is used to select the best position to insert a job [45,46]. Based on the characteristics of the DPFSP-WTC, we have the following theorem about the characteristic for insert operation neighborhood.

Theorem 3. Suppose that the completion time of each factory in the current scheduling scheme is $(C_1, C_2, \ldots, C_{l-1}, C_l, \ldots, C_f)$, where $C_1 \geq C_2 \geq \cdots \geq C_{l-1} \geq C_l \geq \cdots \geq C_f$. Now, consider inserting a new job *J*, and the temporary solution after insertion is optimal. Try to insert job *J* into factory *f*, factory $f-1$,..., and so forth. That is, first try the factory with the smallest completion time, then try the factory with the second smallest completion time, and so forth. The new completion time obtained after inserting job *J* into factory *l* is marked as C_l^*. If $C_l^* < C_{l-1}$, that is, C_l^* is smaller than

the current completion time of the factory $l-1$, there is no need to try to insert J into the following factory ($l-1, l-2, \ldots, 1$), because the result of the latter insertion is certainly worse than the current one.

Proof. After inserting job J into factory l, the new solution obtained is $(C_1, C_2, \ldots, C_{l-1}, C_l^*, \ldots, C_f)$, where $C_1 \geq C_2 \geq \cdots \geq C_{l-1} \geq C_l^* \geq \cdots \geq C_f$. Assuming that the job J is inserted into the optimal position of the next factory $l-1$, the solution obtained is $(C_1, C_2, \ldots, C'_{l-1}, C_l, \ldots, C_f)$, where $C_1 \geq C_2 \geq \cdots \geq C'_{l-1} \geq C_l \geq \cdots \geq C_f$. Inserting a new job will definitely increase the completion time of the factory. The temporary intermediate inequality $C'_{l-1} > C_{l-1}$ is obtained. Then, we will attain $C'_{l-1} > C_{l-1} > C_l^*$. So, $(C_1, C_2, \ldots, C'_{l-1}, C_l, \ldots, C_f)$ is worse.

According to Theorem 3, if $C_l^* < C_{l-1}$, there is no need to try a subsequent insertion. This characteristic can be used in algorithm design, which can reduce the amount of calculations.

4. The proposed algorithm

The iterated greedy (IG) algorithm is a new metaheuristic algorithm proposed by Ruiz and Stützle [47], which shows superior performance in job shop scheduling problems [48]. As a simple iterated search algorithm, the IG first initializes a solution, then continuously improves the solution through deconstruction, construction, and local search, and uses a greedy strategy to accept the new solution. Once the maximum number of iterations is reached, the optimization task is finished.

4.1. Representation of the solution

A two-dimensional vector to represent a solution is used. For instance, a solution can be represented as $\pi = \{\pi_1, \pi_2, \ldots, \pi_f\}$, where $\pi_l = (\pi_{l,1}, \pi_{l,2}, \ldots, \pi_{l,nl})$, $l = 1,2,\ldots,f$. π_l is the sequence of jobs in factory l and n_l is the total number of jobs in factory l. In addition, $WTC(\pi)$ represents the result of substituting the completion time of each factory of π into Eq (1).

4.2. Initialization

The Nawaz, Enscore, and Ham (NEH) heuristic is the best heuristic for the permutation flowshop scheduling problem (PFSP) with the maximum completion time criterion. Based on the NEH algorithm, a modified distributed NEH (MDNEH) heuristic algorithm is proposed to generate an initial solution, which is explained in Algorithm 1. First, all jobs are sorted according to non-descending order with respect to the total processing time (see lines 1 and 2 in Algorithm 1). Then, the first f jobs are inserted into f empty factories (see lines 3 to 4 in Algorithm 1). The characteristic for inserting the operation neighborhood in Section 3.2 is used to insert the rest of the jobs to the optimal positions of the factories in turn (see lines 6 to 19 in Algorithm 1). Especially, if $C_{min} < C_k$, there is no need to try a subsequent

insertion (See line 10 in Algorithm 1).

Although Eq (8) in the proposed mathematical model defines that the completion time of the first factory is the maximum, it is not directly applied in the actual implementation (coding). All factories are the same. Therefore, in the actual implementation, we ensure that the completion time of the first factory is the maximum by sorting (see lines 5 and 18 in Algorithm 1). Since f is not large and the completion time of only one factory changes at a time, the imposed calculation expenses can be ignored.

Algorithm 1. MDNEH.

1: Compute $T_i = \sum_j p_{i,j}$ for each job $i \in N$;
2: Generate job permutation $\lambda = (\lambda_1,\lambda_2,\ldots,\lambda_n)$ according to non-descending order of T_i;
3: $\pi := (\pi_1,\pi_2,\ldots,\pi_f)$, where $\pi_k = (\lambda_k), k = 1,2,\ldots,f$;
4: Remove jobs $\lambda_1,\lambda_2,\ldots,\lambda_f$ from λ;
5: Sort the factories from large to small according to the completion time of the factories
to ensure $C_1 \geq C_2 \geq \cdots \geq C_{l-1} \geq C_l \geq \cdots \geq C_f$;
6: **While** sizeof(λ) **do**
7: Extract the first job j from λ;
8: $C_{min} = INF$; $\Delta_k = INF, k = 1,2,\ldots,f$;
9: **for** $k = f$ to 1 **do**
10: **if** $C_{min} < C_k$ **do** break; **endif**
11: Test job j at all possible positions of π_k;
12: $\Delta_k =$ Minimum increase of WTC received by Eq (1);
13: $\xi_{k^*} =$ The best position;
14: C_{min} = Makespan of factory k after job j is inserted into the best position;
15: **endfor**
16: $k^* := \arg(\min_{k=1,2,\ldots f} \Delta_k)$;
17: Insert job j at position ξ_{k^*} of π_{k^*};
18 Update the order of the factories using the pseudocode in line 5;
19: **endwhile**
20: **return** $\pi = (\pi_1, \pi_2, \ldots, \pi_f)$

4.3. Destruction

In the destruction stage, d jobs are extracted from the current solution. The pseudocode is shown in Algorithm 2. First, randomly extract a job from each factory and place these f jobs into λ. If f is less than d, the remaining jobs are randomly selected and extracted.

Algorithm 2. Destruction (π, d).

$\lambda = \emptyset$;
for $k = 1$ to f **do**
    Randomly extract a job J from the factory k;
    Append job J to λ;
    if $sizeof(\lambda) == d$ **do return** π and λ; **endif**
endfor
While $sizeof(\lambda) < d$ **do**
    Randomly select a factory l;
    Randomly extract a job J from the factory l;
    Append job J to λ;
endwhile
return π and λ

4.4. Construction

In the construction stage, an extracted job is randomly selected and then inserted into the temporary partial solution. The pseudocode of the construction algorithm is shown in Algorithm 3.

Algorithm 3. Construction (π, λ).

Sort the factories from the largest to the smallest according to the completion time of the factories to ensure $C_1 \geq C_2 \geq \cdots \geq C_{l-1} \geq C_l \geq \cdots \geq C_f$;
While size of $(\pi) > 0$ **do**
    Randomly extract job j from λ;
    $C_{min} = INF$; $\Delta_k = INF, k = 1,2, \ldots, f$;
    for $k = f$ to 1 **do**
        if $C_{min} < C_k$ **do** break; **endif**
        Test job j at all possible positions $of\ \pi_k$;
        $\Delta_k =$ Minimum increase of WTC gotten by Eq (1);
        $\xi_{k^*} =$ The best position;
        $C_{\min}$ = Makespan of factory k after job j is inserted into the best position;
    endfor
    $k^* := \arg(\min_{k=1,2,\ldots,f} \Delta_k)$;
    Insert job j at position ξ_{k^*} of π_{k^*};

14: Update the order of the factories using the pseudocode in line 1;

15: **endwhile**

16: **return** π

4.5. Local search

Generally, the local search algorithm starts from an initial solution, generates its neighbor solution through neighborhood action, judges the quality of the neighbor solution, selects the neighbor solution according to a certain strategy, and repeats the aforementioned process until the termination condition is reached [49]. The pseudocode of the proposed local search is shown in Algorithm 4. According to the descending order of the completion time of the factories, all jobs in factories are put into a temporary permutation λ in turn (see line 3 in Algorithm 4). Then, the first job λ_1 in λ is extracted from the current solution, and then the job λ_1 is inserted into the best position of the partial solution. If the new solution is better, the current solution is replaced. Then, we try the next job in λ. If continuous $n/2$ jobs are extracted and no better solution is found, the local search is terminated.

Algorithm 4. Local search (π).

1: Sort the factories from the largest to the smallest according to the completion time of the factories to ensure $C_1 \geq C_2 \geq \cdots \geq C_{l-1} \geq C_l \geq \cdots \geq C_f$;

2: $\lambda = \emptyset$;

3: **for** $k = 1$ to f **do** Append all jobs in factory k to λ; **ed**

4: $Cnt = 0$;

5: $j = 1$;

6: **While** $Cnt < n/2$ **do**

7: Extract job λ_j from π;

8: $C_{min} = INF$;$\Delta_k = INF, k = 1,2, \ldots, f$;

9: **for** $k = f$ to 1 **do**

10: **if** $C_{min} < C_k$ **do** break; **endif**

11: Test job j at all possible positions $of\ \pi_k$;

12: $\Delta_k =$ Minimum increase of WTC gotten by Eq (1);

13: $\xi_{k^*} =$ The best position;

14: C_{min} = Makespan of factory k after job j is inserted into the best position;

15: **endfor**

16: $k^* = \arg(\min_{k=1,2,\ldots,f} \Delta_k)$;

17: Insert job j at position ξ_{k^*} of π_{k^*} and a new solution π' is found;

18: **if** $WTC(\pi') < WTC(\pi)$ **then**

19: $\pi = \pi'$; $Cnt = 0$;
20: Update the order of the factories using the pseudocode in line 1;
21: **else**
22: $Cnt = Cnt + 1$;
23: **endif**
24: $j = (j + 1)\%n + 1$;
25: **endwhile**
26: **return** π

4.6. Acceptance criterion

Pan et al. [7] used simulated annealing (SA) to accept the new solution. In this paper, the same strategy has been applied. The calculation of parameter T is shown in Eq (14).

$$T = \beta \frac{\sum_{j=1}^{n} \sum_{k=1}^{m} p_{kj}}{10nm}. \tag{14}$$

When programming and debugging, the value of the *WTC* is found to be very large. When $\Delta = WTC(\pi') - WTC(\pi)$, then the value of Δ will be very large, and therefore, $\exp(-\Delta/T)$ is an extremely small value. In this case, the probability of accepting the worse solution is very small, which is not conducive to the IG jumping out of the local optimum. Thus, the *WTC* cannot be used directly. In order to overcome this challenge, a method explained in Algorithm 5 is utilized to accept new solutions. In the new acceptance criterion, $\Delta = C'_k - C_k$, in which its value will not be too large. The proposed algorithm will accept the worse solution with a certain probability to jump out of local optima.

Algorithm 5. Acceptance criterion $(\pi', \pi, \pi^*, \beta)$.

1: **If** $WTC(\pi') < WTC(\pi)$ **then**
2: $\pi = \pi'$;
3: **If** $WTC(\pi') < WTC(\pi^*)$ **then** $\pi^* = \pi'$; **endif**
4: **else**
5: **for** $k = 1$ to f **do**
6: **if** $C_k < C'_k$ **then** $\Delta = C'_k - C_k$; break; **endif**
7: **if** rand(0,1) $< \exp(-\Delta/T)$ **then** $\pi = \pi'$; **endif**
8: **endif**

4.7. Procedures of the proposed algorithm

The proposed algorithm is now complete and its details are given in Algorithm 6. It should be noted that in order to make the jobs sequence of each factory evolve, we set d to be greater than or equal to the number of factories, $d = \max(d,f)$ (see line 2 in Algorithm 6).

Algorithm 6. IIG(d, β).

$\pi = MDNEH$;
$d = \max(d, f)$;
Local search (π);
$\pi^* = \pi$; //best solution so far
While Stopping condition not met **do**
    $\pi' = \pi$;
    $[\lambda, \pi'] =$ Destruction (π', d);
    $\pi' =$ Construction(π', λ);
    $\pi' =$ Local Search(π');
    Acceptance criterion $(\pi', \pi, \pi^*, \beta)$
endwhile
return π^*

5. Experimental calibration

In this section, the parameters of the IIG algorithm are fine-tuned and adjusted to achieve its best performance. The IIG has two parameters that may affect its performance, d and β. For these two parameters, we first determine their general range according to the existing literature [7,48]. Then, the final value of each parameter is determined through experiments. For the IIG algorithm, d is at three levels: 4, 5, and 6; β is at three levels: 0.2, 0.3, and 0.4. In this way, the IIG algorithm has 3 × 3 = 9 different configurations.

The instances [5] used in the parameter adjustment are described as follows. The number of factories f is randomly selected from the set {2, 3, 4, 5, 6, 7}. The combinations of n and m are randomly selected from the set {20 × 5, 20 × 10, 20 × 20, 50 × 5, 50 × 10, 50 × 20, 100 × 5, 100 × 10, 100 × 20, 200 × 10, 200 × 20, and 500 × 20}. The processing time is randomly generated with uniform distribution within the range of [1, 99]. In addition, fifty instances for calibration are randomly generated.

The proposed IIG algorithm was coded and implemented using C++ in Visual Studio 2017. We performed configurations on an Inter(R) Xeon(R) CPU E5-2630 2.4 GHz with 16 GB of RAM running the Windows 7 Standard 64-bits Operating System, and stop the process when the termination criterion of the maximum elapsed CPU time of 6mn milliseconds is met (n is the number of jobs and m is the number of machines). For each of the configurations, the 50 calibration instances are solved. For each of the instances, 15 independent replications are carried out. There are 9 × 15 × 50 = 6750 solutions in total. Many scholars used the relative percentage increase (RPI) to evaluate the performance of the algorithms [26,30]. In this paper, the RPI is used with some modifications. The reasons are as follows. First, the value given in Eq (1) is too large to be directly used in RPI calculation. In addition, this paper needs to evaluate the completion time of all factories at the same time. Therefore, the RPI calculation formula is slightly modified, as given in Eq (15):

$$RPI_l = \begin{cases} \frac{100\times(C_l - C_l^*)}{C_l^*}, & l = 1 \\ \frac{100\times(C_l - C_l^*)}{C_l^*}, & l > 1 \text{ and } RPI_{l-1} = 0 \\ -1, & \text{otherwise} \end{cases} \quad (15)$$

In Eq (15), l represents the l^{th} factory of a solution, C_l is the completion time of the l^{th} factory of a solution, and C_l^* is the completion time of the l^{th} factory of the best solution among the 9 × 15 = 135 results. For a solution, there are f factories and f RPIs.

When $RPI_{l-1} > 0$, it means that the completion time of the $(l-1)^{th}$ factory is greater than that of the $(l-1)^{th}$ factory of the optimal solution ($C_{l-1} > C_{l-1}^*$). In this case, the completion time (C_l) of the l^{th} factory is meaningless as described by Theorem 1. Suppose there is a solution, the vector composed of the completion time of each factory is (103,102,99), and the completion time of each factory corresponding to the optimal solution is (103,101,100). It can be seen that although the completion time of the third factory (99) is smaller than thatof the third factory of the optimal solution (100), this value is meaningless, because the completion time of the second factory (102) is already larger than that of the second factory of the optimal solution (101). Hence, the three RPIs are: $RPI_1 = 100 * \frac{103-103}{103} = 0, RPI_2 = 100 * \frac{102-101}{101} = 0.99, RPI_3 = -1$. When $RPI_l = -1$, it is called invalid. The valid RPIs are RPIs whose values are greater than or equal to zero. In the reported statistical results, invalid RPIs are excluded. The RPI of the first factory of each solution must be valid, so there are at least 108 × 5 = 6750 RPIs. The experimental design (DOE) and analysis of variance (ANOVA) are used to analyse all valid RPIs. The mean plot with 95% Tukey's honest significant difference (HSD) confidence intervals are shown in Figure 1. In fact, Figure 1 shows that the best parameter configurations are d = 5 and β = 0.3.

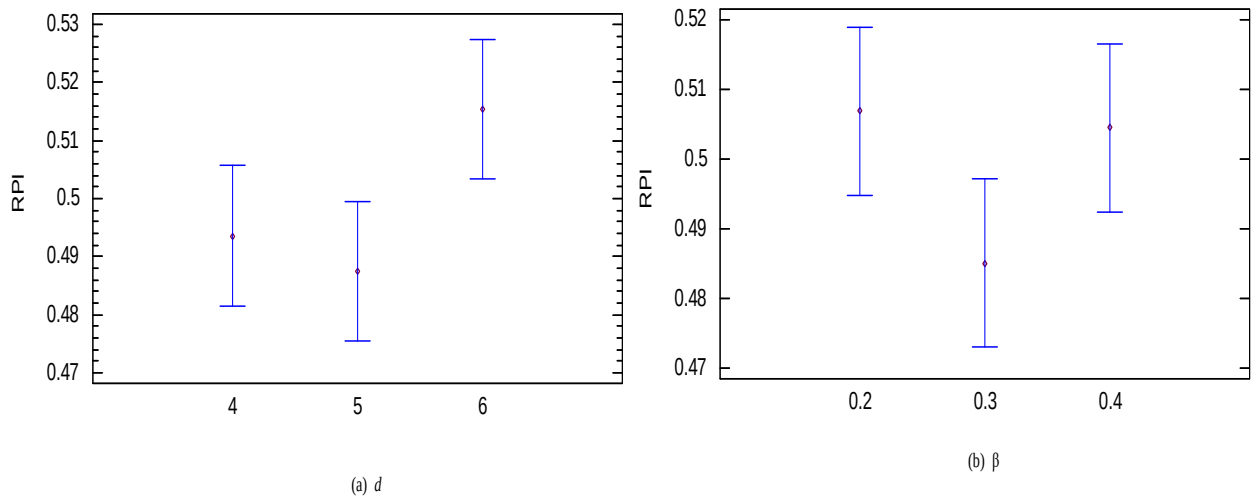


Figure 1. Mean plots for all factories for the IIG calibration.

6. Computational evaluation

Experiments are executed to verify the performance of the proposed IIG. The test instances are from 720 test instances proposed by Naderi and Ruiz [5]. In these instances, the range of factory number f, job number n, and machine number m are $f \in \{2,3,4,5,6,7\}$, $n \in \{20,50,100,200,500\}$, and $m \in \{5,10,20\}$, respectively.

To validate the mathematical model presented in Section 3, we implement it using IBM ILOG CPLEX Optimization Studio 12.7.1 and perform computational tests on several instances. For the smallest instance ($f = 2$, $n = 20$, $m = 5$) mentioned above, CPLEX runs for 12 hours without obtaining any results. We therefore turn to smaller-scale instances. In the case of ($f = 2$, $n = 10$, $m = 5$), CPLEX obtains a result consistent with the IIG algorithm after a runtime of 90 seconds. These experimental findings confirm that while the mathematical model is correct, it requires substantial computational resources for larger instances, highlighting its computational limitations.

All algorithms are implemented in Microsoft Visual Studio 2017 using C++. The experiments are carried out on the same computer mentioned in Section 5. In order to have fair comparison, the termination time of all algorithms is the same. The termination time is $t = vnm$ milliseconds, where n represents the number of jobs and m represents the number of machines. The values of v are 10, 20, and 30 to show the performance of all algorithms more comprehensively. For each of the instances, 10 independent replications are carried out.

In addition to using RPI_l in Section 5, N_l^{alg} is also defined to evaluate the performance of the algorithm. N_l^{alg} represents the number of solutions whose completion times from factory 1 to factory l found by algorithm alg are equal to the completion times from factory 1 to factory l of the optimal solution, where alg is a competition algorithm, such as DABC [7]. The bigger the N_l^{alg}, the better.

6.1. Comparison of results by different optimization objectives

By minimizing makespan or total flowtime, the completion time of all factories will be minimized, which can be considered as an intuitive idea, since makespan is defined as the maximum completion time of all factories and total flowtime is the sum of all jobs' completion times. In this section, the optimization goal of the proposed algorithm is modified to optimize makespan and total flowtime, keeping other parameters and assumptions unchanged, marked as IIG_C and IIG_F, respectively.

Three algorithms (i.e., IIG_C, IIG_F, and IIG) are used to solve the 720 test instances. The average RPI (ARPI) values are tabulated in Table 4. As can be seen from the table, IIG is optimal in all cases. Additionally, in all cases, $N_l^{IIG_F}(l > 1)$ and $N_l^{IIG_C}(l > 1)$ are all 0. IIG_F and IIG_C have no advantage in optimizing the completion time of all other factories (not including the first factory), hence the results of N_l^{alg} are not reported in Table 4.

Table 4. The obtained ARPI values (the minimum ARPI values are highlighted in bold).

Type	10*mn* milliseconds			20*mn* milliseconds			30*mn* milliseconds		
	IIG_C	IG_F	IIG	IIG_C	IIG_F	IIG	IIG_C	IIG_F	IIG
***f* = 2**	4.620	10.384	**0.399**	4.620	10.316	**0.269**	4.620	10.291	**0.194**
***f* = 3**	5.196	12.047	**0.400**	5.196	11.980	**0.253**	5.196	11.978	**0.170**
***f* = 4**	5.349	12.896	**0.425**	5.349	12.771	**0.252**	5.349	12.761	**0.164**
***f* = 5**	5.387	13.628	**0.372**	5.387	13.588	**0.222**	5.387	13.551	**0.144**
***f* = 6**	5.463	14.157	**0.422**	5.463	14.191	**0.241**	5.463	14.113	**0.151**
***f* = 7**	5.259	14.091	**0.428**	5.259	13.948	**0.243**	5.259	13.918	**0.153**
***n* = 20**	5.515	13.060	**0.078**	5.515	13.075	**0.046**	5.515	13.081	**0.033**
***n* = 50**	6.457	16.432	**0.767**	6.457	16.285	**0.472**	6.457	16.253	**0.319**
***n* = 100**	4.859	12.788	**0.661**	4.859	12.734	**0.402**	4.859	12.676	**0.255**
***n* = 200**	4.316	10.187	**0.709**	4.316	10.109	**0.440**	4.316	10.044	**0.238**
***n* = 500**	3.412	7.197	**0.651**	3.412	7.088	**0.393**	3.412	7.107	**0.190**
***m* = 5**	5.587	15.648	**0.341**	5.587	15.584	**0.200**	5.587	15.532	**0.143**
***m* = 10**	5.546	13.424	**0.436**	5.546	13.359	**0.269**	5.546	13.330	**0.176**
***m* = 20**	4.721	10.754	**0.43**	4.721	10.679	**0.256**	4.721	10.662	**0.158**
Mean	5.212	12.867	**0.408**	5.212	12.799	**0.245**	5.212	12.769	**0.160**

We performed a multifactor ANOVA with *f*, *n,* and *m* as factors when the termination time was only 30*mn* milliseconds. The mean plots for different types of algorithms and the interaction plots are shown in Figure 2. Also, 95% confidence leveles and Tukey HSD confidence intervals have been demonstrated in Figure 2. Figure 2(a) shows that the ARPI values generated by the three algorithms are statistically different.

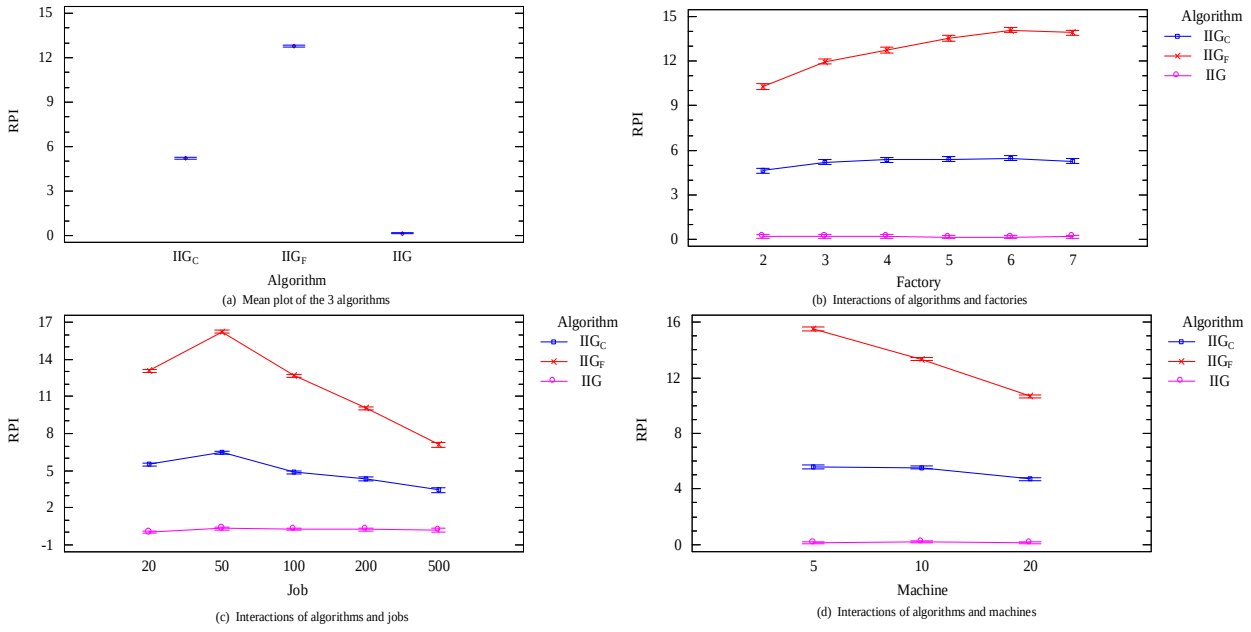


Figure 2. The mean plots, interactions, and 95% Tukey HSD intervals at t = 30*mn* milliseconds.

The results shown in Figure 2 are consistent with those reported in Table 4. The IIG is the most suitable for optimal solving of the DPFSP-WTC. The reasons for the obtained results are analyzed as follows. Taking makespan and total flowtime as optimization objectives will indeed lead to the minimum completion time of all factories, especially in the early period of algorithm execution. However, there are at least two reasons for the unsatisfactory results. First, different optimization objectives lead to the fact that the fitness cannot really reflect the quality of the solution, such as the example given in Section 1, which leads the algorithm stopping evolution. The second reason is that minimizing the completion time of all factories by taking makespan and total flow time as indicators makes it easy to fall into local optima.

6.2. Comparison of metaheuristics

The IIG algorithm is also compared with the evolutionary algorithm (EA) [8], scatter search (SS) [15], discrete artificial bee colony (DABC) [7], iterated local search (ILS) [7], and modified iterated greedy (MIG) [12]. The original competitive algorithms have been appropriately modified, because the optimization objectives of these algorithms need to be modified. The parameter settings of each algorithm are shown in Table 5.

Table 5. Parameter settings for the DABC, EA, SS, ILS, and MIG.

EA	$\gamma = 5$
SS	$b = 10, l = 10, \alpha = 40, p = 0.1$
DABC	$PS = 5, \Xi =$ *pairwise interchange*
MIG	$T_0 = 5, \lambda = 0.9, I_{iter} = 1000, \alpha_{min} = 2, \alpha_{max} = 7$
ILS	α = *DLR_DNEH(0.2)*, $\omega = 20, \tau = 3, \beta = 0.7, \Xi =$ *insertion*

6.2.1. Comparison of the overall performance of the algorithms

In this section, overall performance of all algorithms are compared and analyzed. Table 6 shows the average RPI (APRI) value when $t = 10mn$. The data in Table 6 are summarized according to the number of factories, jobs, and machines. It can be seen that the proposed algorithm IIG is superior in all cases compared to other reported optimizers.

Table 6. The ARPI for $10mn$ milliseconds (the minimum ARPI values are highlighted in bold).

Type	DABC	EA	SS	ILS	MIG	IIG
$f = 2$	0.729	0.734	0.620	0.738	0.579	**0.427**
$f = 3$	0.869	0.845	0.772	1.034	0.585	**0.443**
$f = 4$	0.920	0.841	0.845	1.472	1.208	**0.467**
$f = 5$	0.900	0.829	0.807	1.653	1.056	**0.381**
$f = 6$	0.555	0.371	0.615	0.569	0.379	**0.275**
$f = 7$	0.429	0.292	0.569	0.345	0.351	**0.281**
$n = 20$	0.313	0.114	0.201	0.285	0.194	**0.063**
$n = 50$	1.242	1.428	1.129	2.024	1.270	**0.857**

Continued on next page

Type	DABC	EA	SS	ILS	MIG	IIG
***n* = 100**	0.919	1.167	1.099	1.582	1.403	**0.702**
***n* = 200**	1.117	1.215	1.240	1.494	1.102	**0.740**
***n* = 500**	1.147	1.032	1.055	1.289	1.000	**0.634**
***m* = 5**	0.657	0.567	0.699	1.120	0.681	**0.336**
***m* = 10**	0.805	0.704	0.736	1.166	0.782	**0.409**
***m* = 20**	0.816	0.700	0.740	1.000	0.726	**0.404**
Mean	0.770	0.667	0.728	1.085	0.733	**0.387**
Rank	5	2	3	6	4	**1**

A multifactor ANOVA is applied on the data when $t = 10mn$ milliseconds, and the results are shown in Figure 3. The proposed IIG algorithm is the best among other considered optimizers, which is significantly different from other algorithms, as shown in Figure 3(a). Figure 3(b)-(d) show the interaction plots of factory numbers and algorithms, job numbers, and algorithms, and machine numbers and algorithms, respectively.

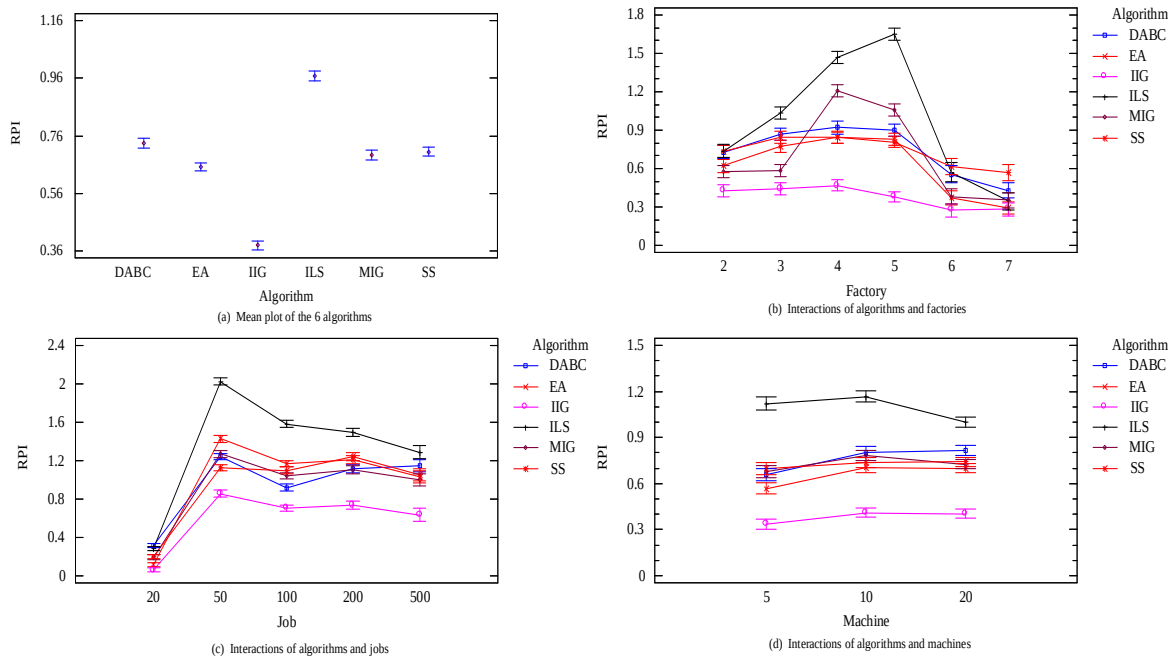


Figure 3. The mean plots, interactions, and 95% Tukey HSD intervals at $t = 10mn$ milliseconds.

As can be seen in Figure 3, when the scale of the instance is relatively small, all algorithms could find best solutions. As the scale of the instance increases, the performance differences of all algorithms begin to appear. The scale of the instance further increases and the instance becomes more difficult. In this case, all algorithms have difficulty solving the instances. It can be seen from Figure 3 that the IIG is the best among the reported algorithms, which is consistent with the conclusion obtained from Table 6.

Tables 7 and 8 show the ARPI values at $t = 20mn$ and $t = 30mn$ milliseconds, respectively.

Figures 4 and 5 are the data analysis for these two cases. It can be seen that the performance of IIG is stable and is the best in most cases (except f = 7, when t = 30mn milliseconds). However, the performance of some algorithms (such as MIG and DABC) is unstable. For instance, the MIG algorithm performed poorly when the number of factories was 4 or 5, while the DABC algorithm performed better when m = 100.

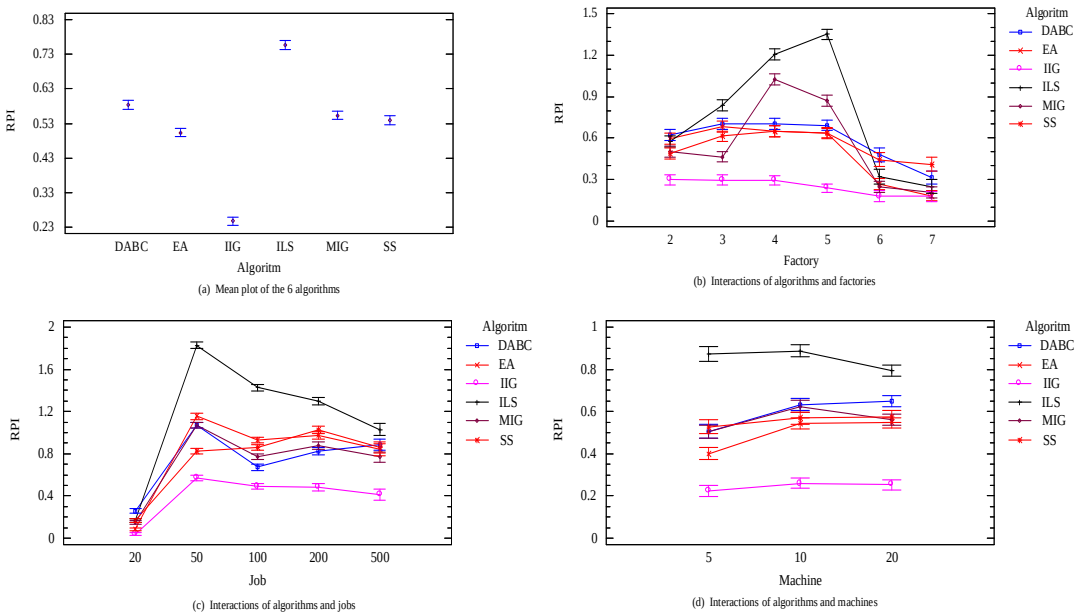


Figure 4. The mean plots, interactions, and 95% Tukey HSD intervals at t = 20mn milliseconds.

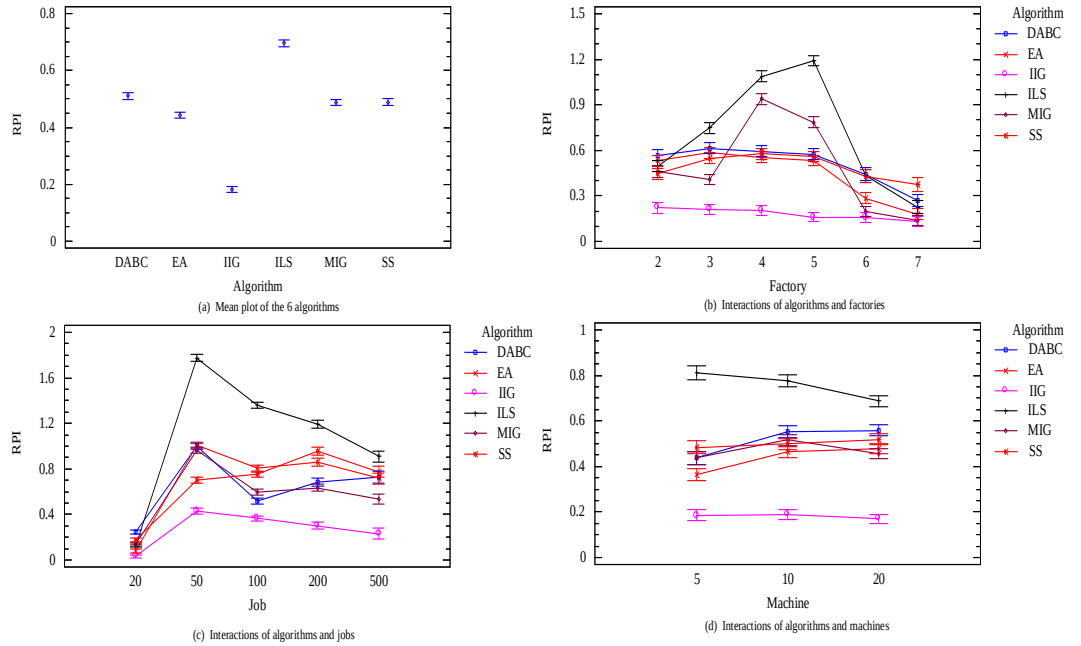


Figure 5. The mean plots, interactions, and 95% Tukey HSD intervals at t = 30mn milliseconds.

Table 7. The obtained ARPI for 20*mn* milliseconds (the minimum ARPI values are highlighted in bold).

Type	DABC	EA	SS	ILS	MIG	IIG
***f* = 2**	0.621	0.597	0.491	0.574	0.504	**0.298**
***f* = 3**	0.702	0.681	0.614	0.836	0.464	**0.296**
***f* = 4**	0.703	0.650	0.648	1.208	1.027	**0.292**
***f* = 5**	0.691	0.639	0.634	1.351	0.869	**0.237**
***f* = 6**	0.479	0.269	0.444	0.320	0.247	**0.178**
***f* = 7**	0.312	0.181	0.410	0.248	0.206	**0.177**
***n* = 20**	0.258	0.083	0.166	0.163	0.150	**0.038**
***n* = 50**	1.073	1.154	0.822	1.828	1.068	**0.570**
***n* = 100**	0.673	0.930	0.861	1.428	0.770	**0.490**
***n* = 200**	0.826	0.976	1.030	1.297	0.877	**0.482**
***n* = 500**	0.883	0.838	0.862	1.029	0.767	**0.412**
***m* = 5**	0.506	0.400	0.527	0.874	0.505	**0.223**
***m* = 10**	0.634	0.543	0.568	0.888	0.624	**0.261**
***m* = 20**	0.649	0.547	0.577	0.793	0.563	**0.253**
Mean	0.606	0.506	0.561	0.845	0.568	**0.247**
Rank	5	2	3	6	4	**1**

Table 8. ARPI for 30*mn* milliseconds (minimum ARPI values are in bold).

Type	DABC	EA	SS	ILS	MIG	IIG
***f* =2**	0.568	0.531	0.444	0.496	0.460	**0.222**
***f* = 3**	0.614	0.584	0.549	0.747	0.407	**0.212**
***f* = 4**	0.594	0.556	0.577	1.087	0.938	**0.204**
***f* = 5**	0.575	0.531	0.558	1.191	0.786	**0.160**
***f* = 6**	0.443	0.284	0.430	0.431	0.196	**0.158**
***f* = 7**	0.092	**0.026**	0.133	0.080	0.111	0.030
***n* = 20**	0.270	0.084	0.179	0.136	0.155	**0.031**
***n* = 50**	1.003	1.000	0.696	1.770	1.023	**0.435**
***n* = 100**	0.488	0.784	0.725	1.331	0.686	**0.341**
***n* = 200**	0.654	0.831	0.936	1.168	0.731	**0.283**
***n* = 500**	0.722	0.706	0.769	0.903	0.619	**0.225**
***m* = 5**	0.422	0.341	0.467	0.786	0.455	**0.168**
***m* = 10**	0.544	0.452	0.482	0.776	0.552	**0.175**
***m* = 20**	0.605	0.530	0.528	0.747	0.532	**0.181**
Mean	0.531	0.447	0.495	0.767	0.518	**0.175**
Rank	5	2	3	6	4	**1**

Figure 6 shows the performance of all algorithms under different termination times. With the increase of the termination time, all algorithms continue to evolve without entering the stagnation state. Figure 6 displays the performance of all algorithms in a panoramic manner instead of a convergence curve. It can be seen that the proposed IIG algorithm achieved good performance at multiple termination times. In summary, it can be concluded with certainty that the proposed IIG is the new state-of-the-art algorithm for solving the DPFSP-WTC.

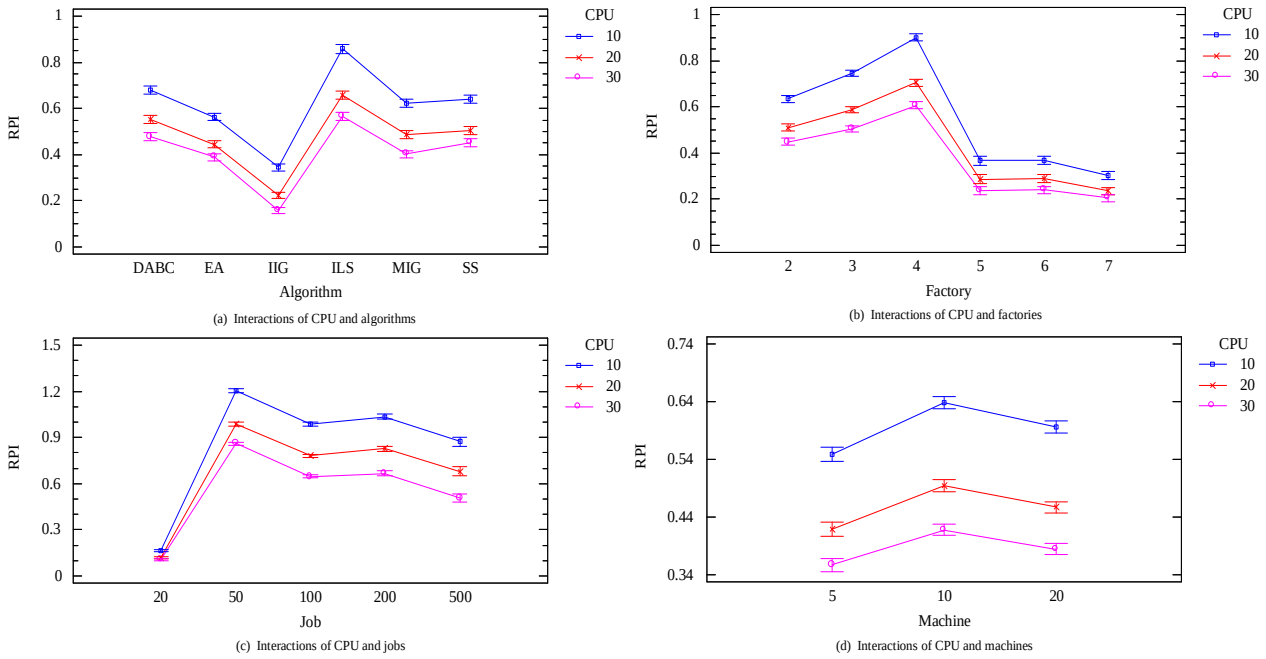


Figure 6. Interactions and 95% Tukey HSD intervals of the CPU times, as well as the type of algorithm, number of factories, number of jobs, and number of machines.

6.2.2. Comparison of the completion time of each factory

In this section, the completion time details of each factory are shown. Tables 9 to 11 give the average valid $ARPI_l$ and N_l^{alg} values of each factory when $t = 10mn$, $t = 20mn$ and $t = 30mn$ milliseconds, respectively. Since there are 120 instances for each factory number (f) and each algorithm runs 10 times, the number of results by each algorithm is 1200 for each factory number. Therefore, the ARPI of the first factory (RPI_1) given in Tables 9 to 11 is the average of 1200 RPIs.

The number of valid RPI_2's (also including $RPI_3, RPI_4, RPI_5, RPI_6$, and RPI_7) is uncertain. For example, in Table 9, when $f = 2$, $ARPI_1^{DABC}$(the superscript DABC is the algorithm name) = 0.75, $ARPI_1^{EA}$ = 0.79, $ARPI_1^{SS}$ = 0.68, $ARPI_1^{ILS}$ = 0.79, $ARPI_1^{MIG}$ = 0.64, and $ARPI_1^{IIG}$ = 0.48 are the average of 1200 RPIs. For the second factory, there are 45 (equal to N_1^{DABC}) valid RPIs for the DABC algorithm. That is, $ARPI_2^{ABC}$ is the average of 45 numbers. Similarly, $ARPI_2^{SS}$ is the average of 112 numbers. $ARPI_2^{ILS}$ is the average of 85 numbers. However, $ARPI_2^{IIG}$ is the average of 164 numbers. N_2^{IIG} = 142 means that 142 out of 164 RPIs are zero. This number 142 is larger than 112 (and 85). That is, when $f = 2$, although $ARPI_2^{SS}$ and $ARPI_2^{ILS}$ are the smallest in Table 9, the makespan of the

first factory and the makespan of the second factory obtained by the IIG are the best. Although the $ARPI_l^{IIG}$ of the IIG algorithm in Tables 9 to 11 is not the smallest in many places, it does not mean that the IIG is not the best. Besides, other data can be analyzed using the proposed method, and it is concluded that the IIG performs well. The performance shown by the indicator N_l^{alg} is relatively intuitive. Generally, large N_l^{alg} means good performance.

Table 9. The $ARPI_l$ and N_l^{alg} for $10mn$ milliseconds (the minimum $ARPI_l$ and maximum N_l^{alg} are highlighted in bold).

f	l	$ARPI_l$						N_l^{alg}					
		DABC	EA	SS	ILS	MIG	IIG	DABC	EA	SS	ILS	MIG	IIG
2	**1**	0.75	0.79	0.68	0.79	0.64	**0.48**	45	102	112	85	142	**164**
	2	0.08	0.04	**0.02**	**0.02**	0.05	0.04	36	89	104	80	123	**142**
3	**1**	0.91	0.96	0.89	1.14	0.7	**0.56**	39	99	108	68	145	**198**
	2	0.12	0.14	**0.08**	0.09	**0.08**	**0.08**	29	75	90	56	121	**167**
	3	**0**	0.01	0.02	**0**	0.03	0.02	29	72	85	55	112	**158**
4	**1**	1	1.01	0.98	1.63	1.31	**0.63**	46	98	88	54	54	**178**
	2	0.13	0.11	0.13	**0.05**	0.24	0.1	31	80	65	41	31	**146**
	3	0.02	**0**	0.04	0.08	0.04	0.01	30	80	61	39	30	**144**
	4	0.01	0.01	**0**	0.01	**0**	0.01	29	77	60	38	30	**134**
5	**1**	1.12	1.17	1.08	1.97	1.25	**0.63**	79	131	117	71	74	**205**
	2	0.04	**0.02**	0.03	0.04	0.05	**0.02**	64	114	99	53	52	**189**
	3	**0**	**0**	0.01	0.01	0.01	0.01	64	114	97	52	50	**185**
	4	**0**	0.02	0.01	0.02	0.08	**0**	64	111	94	49	43	**184**
	5	**0**	0.01	**0**	0.04	**0**	**0**	63	103	91	48	42	**178**
6	**1**	1.37	1.38	1.27	2.03	0.73	**0.83**	130	201	135	141	169	**225**
	2	0.3	**0.15**	0.45	0.36	0.25	**0.15**	64	131	55	55	88	**151**
	3	0.06	**0.04**	0.09	0.08	0.07	0.05	51	119	43	32	62	**125**
	4	0.04	**0.01**	0.04	0.01	0.02	0.04	47	**116**	39	29	57	109
	5	**0**	**0**	**0**	**0**	0.02	0.01	47	**116**	39	29	55	108
	6	**0**	0.03	**0**	0.02	0.03	**0**	47	**112**	39	27	52	107
7	**1**	1.41	1.45	1.29	2.13	0.71	**0.89**	200	244	194	200	212	**253**
	2	0.23	**0.1**	0.46	0.27	0.33	0.16	121	**197**	84	113	104	175
	3	**0.17**	0.08	0.2	0.26	0.24	0.2	60	**151**	41	48	55	102
	4	0.06	0.02	0.06	0.06	0.1	**0.01**	54	**146**	35	40	44	101
	5	0.01	0.01	**0**	0.04	0.01	0.03	52	**133**	35	36	43	93
	6	**0.01**	0.02	0.02	**0.01**	**0.01**	0.03	51	**124**	34	35	42	85
	7	0.1	0.04	**0**	0.08	0.09	0.04	46	**117**	34	32	38	80

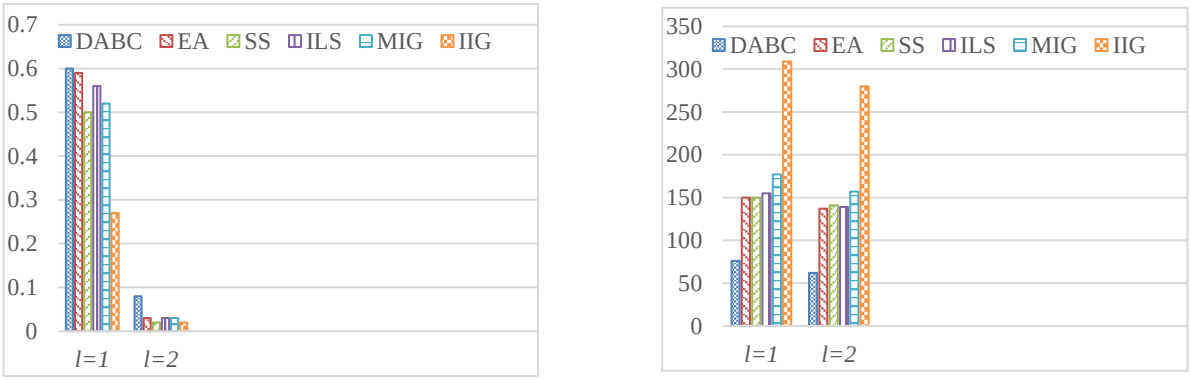
Table 10. The $ARPI_l$ and N_l^{alg} for 10mn milliseconds (the minimum $ARPI_l$ and maximum N_l^{alg} are highlighted in bold).

f	l	$ARPI_l$						N_l^{alg}					
		DABC	EA	SS	ILS	MIG	IIG	DABC	EA	SS	ILS	MIG	IIG
2	**1**	0.65	0.66	0.55	0.63	0.57	**0.35**	62	137	139	125	161	**232**
	2	0.08	**0.02**	**0.02**	**0.02**	0.04	0.03	50	125	131	117	141	**203**
3	**1**	0.76	0.79	0.72	0.97	0.57	**0.39**	57	119	121	112	174	**235**
	2	0.1	0.13	0.07	0.09	0.07	**0.05**	43	90	104	92	146	**208**
	3	**0**	0.01	0.02	0.01	0.03	0.02	43	87	98	88	134	**198**
4	**1**	0.79	0.8	0.78	1.41	1.13	**0.43**	62	112	106	78	64	**231**
	2	0.1	0.1	0.11	**0.04**	0.23	0.06	46	96	82	66	38	**201**
	3	**0**	**0**	0.01	0.04	0.04	0.01	46	96	81	63	36	**198**
	4	0.01	0.01	0.01	0.01	**0**	0.01	44	92	78	61	36	**189**
5	**1**	0.87	0.93	0.86	1.75	1.04	**0.43**	88	148	125	104	82	**264**
	2	0.04	**0.02**	0.03	0.03	0.04	**0.02**	73	131	107	86	61	**241**
	3	**0**	**0**	**0**	0.01	0.01	**0**	71	130	106	85	58	**237**
	4	**0**	0.01	0.01	0.02	0.06	**0**	71	129	103	81	51	**236**
	5	**0**	**0**	**0**	0.02	**0**	**0**	70	121	100	78	50	**230**
6	**1**	1.09	1.12	1.05	1.83	0.54	**0.61**	143	224	149	165	216	**261**
	2	0.26	0.12	0.38	0.24	0.16	**0.09**	74	159	73	87	140	**204**
	3	0.06	0.04	0.06	0.05	0.06	**0.03**	58	146	59	69	107	**184**
	4	0.05	**0.01**	0.02	0.03	0.02	0.04	53	143	56	61	97	**164**
	5	**0**	**0**	**0**	**0**	0.02	**0**	53	143	56	61	94	**163**
	6	**0**	0.02	**0**	0.01	0.02	**0**	53	139	56	59	91	**162**
7	**1**	1.09	1.19	1.06	1.93	**0.53**	0.67	209	254	206	223	256	**280**
	2	0.2	**0.07**	0.39	0.2	0.22	0.1	137	218	97	152	158	**220**
	3	0.14	**0.05**	0.2	0.21	0.21	0.13	78	**184**	50	81	97	145
	4	0.04	**0.01**	0.07	0.04	0.06	**0.01**	72	**180**	43	71	82	144
	5	**0**	0.01	**0**	0.02	**0**	0.01	70	**167**	43	67	81	137
	6	0.02	0.02	**0**	0.02	0.01	0.04	68	**158**	43	64	78	125
	7	0.09	0.04	**0**	0.11	0.09	0.03	61	**149**	43	58	71	118

Table 11. The $ARPI_l$ and N_l^{alg} for 30mn milliseconds (the minimum $ARPI_l$ and maximum N_l^{alg} are highlighted in bold).

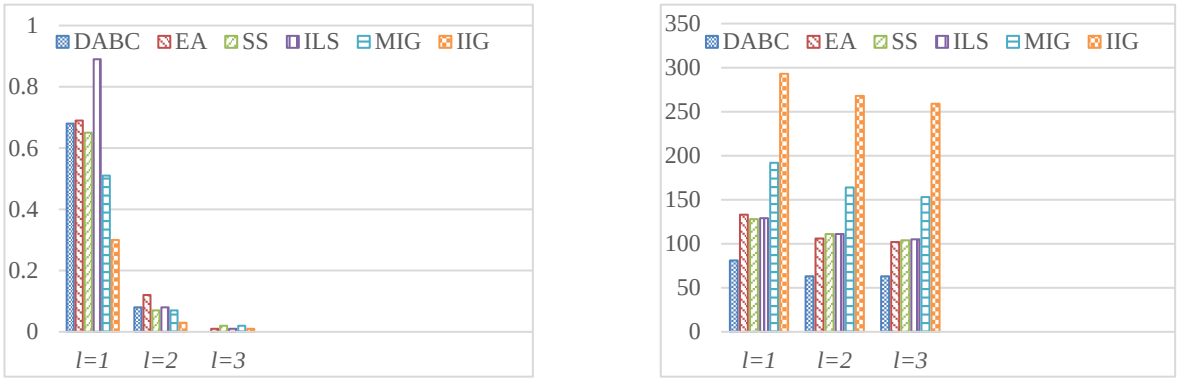
f	l	$ARPI_l$						N_l^{alg}					
		DABC	**EA**	**SS**	**ILS**	**MIG**	**IIG**	**DABC**	**EA**	**SS**	**ILS**	**MIG**	**IIG**
2	**1**	0.6	0.59	0.5	0.56	0.52	**0.27**	76	150	150	155	177	**309**
	2	0.08	0.03	**0.02**	0.03	0.03	**0.02**	62	137	141	139	157	**280**
3	**1**	0.68	0.69	0.65	0.89	0.51	**0.3**	81	133	128	129	192	**293**
	2	0.08	0.12	0.07	0.08	0.07	**0.03**	63	106	111	111	164	**268**
	3	**0**	0.01	0.02	0.01	0.02	0.01	63	102	104	105	153	**259**
4	**1**	0.69	0.7	0.71	1.32	1.05	**0.33**	85	120	114	96	73	**295**
	2	0.09	0.1	0.1	0.03	0.22	**0.04**	66	103	90	87	45	**263**
	3	**0**	**0**	0.02	0.07	0.03	**0**	65	103	88	79	44	**261**
	4	0.01	0.01	**0**	0.01	**0**	0.01	63	99	87	76	44	**252**
5	**1**	0.75	0.78	0.77	1.63	0.95	**0.32**	108	160	135	126	85	**339**
	2	0.04	0.02	0.03	0.03	0.04	**0.02**	88	143	114	109	64	**310**
	3	0	0.01	0	0	0.01	**0**	86	141	113	108	61	**303**
	4	0	0.01	0.01	0.01	0.05	**0**	86	140	110	105	55	**302**
	5	0	0	0	0.03	0	**0**	85	132	107	101	54	**296**
6	**1**	0.94	0.98	0.95	1.73	0.46	**0.5**	152	233	163	186	268	**290**
	2	0.23	0.11	0.33	0.17	0.11	**0.07**	83	168	89	120	202	**239**
	3	0.07	**0.02**	0.05	0.05	0.04	**0.02**	64	156	74	98	168	**220**
	4	0.03	**0.01**	**0.01**	0.02	0.02	0.03	60	152	72	89	158	**202**
	5	**0**	**0**	**0**	**0**	0.01	**0**	60	152	72	89	155	**201**
	6	**0**	0.02	**0**	0.01	0.01	**0**	60	148	72	87	151	**200**
7	**1**	0.95	1.04	0.96	1.83	**0.45**	0.56	216	255	216	240	301	**304**
	2	0.19	**0.07**	0.37	0.18	0.16	0.08	147	223	105	172	214	**241**
	3	0.13	**0.04**	0.18	0.16	0.14	0.1	89	**192**	55	105	157	172
	4	0.04	**0**	0.05	0.03	0.04	**0.01**	83	**191**	49	97	140	171
	5	**0**	0.01	**0**	0.02	0	0.01	81	**177**	49	92	139	163
	6	0.02	0.03	**0**	0.03	0.01	0.03	78	**164**	49	86	135	151
	7	0.09	0.04	**0**	0.12	0.05	0.03	70	**155**	49	78	128	144

Figures 7 to 12 graphically display the data of Table 11 (t = 30mn milliseconds) and show the advantages of the IIG algorithm. It can be seen that the N_l^{IIG} value is the largest in most cases. There are several special cases, such as N_3^{alg}, N_4^{alg}, and N_5^{alg} when f = 6, and N_2^{alg}, N_3^{alg}, N_4^{alg}, N_5^{alg}, N_6^{alg}, and N_7^{alg} when f = 7. In these cases, as long as N_0^{IIG} is the largest, IIG is the best compared to the other considered optimization methods. Overall, it can be inferred that the IIG minimized the makespans of all factories and performs well among the comparison pool.



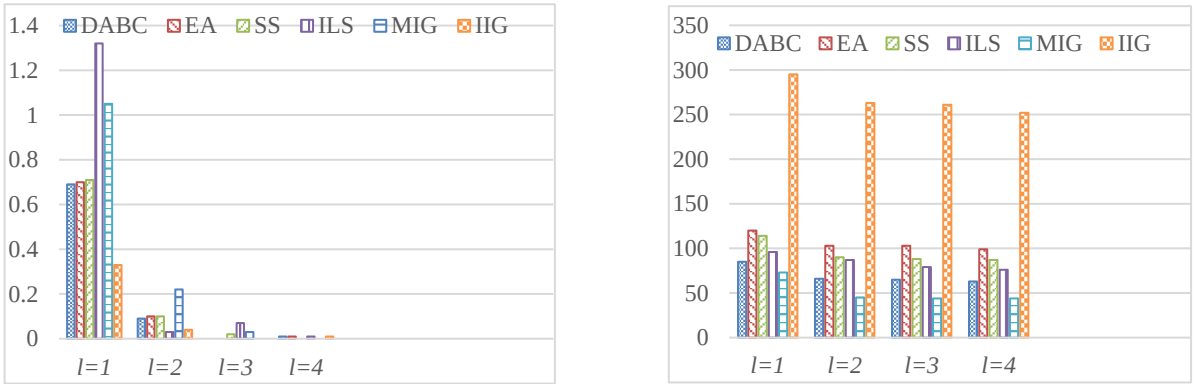
(a) $ARPI_l$ (b) N_l^{alg}

Figure 7. The $ARPI_l$ and N_l^{alg} ($f = 2$) for 30mn milliseconds.



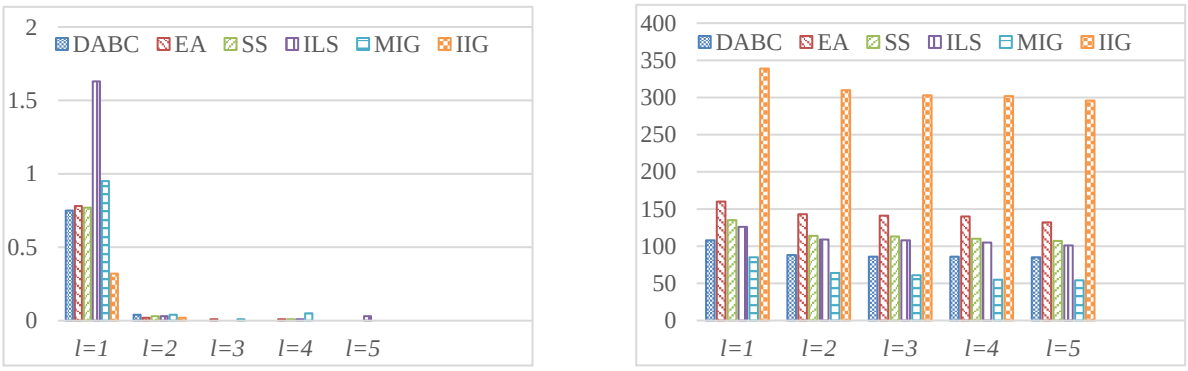
(a) $ARPI_l$ (b) N_l^{alg}

Figure 8. The $ARPI_l$ and N_l^{alg} ($f = 3$) for 30mn milliseconds.



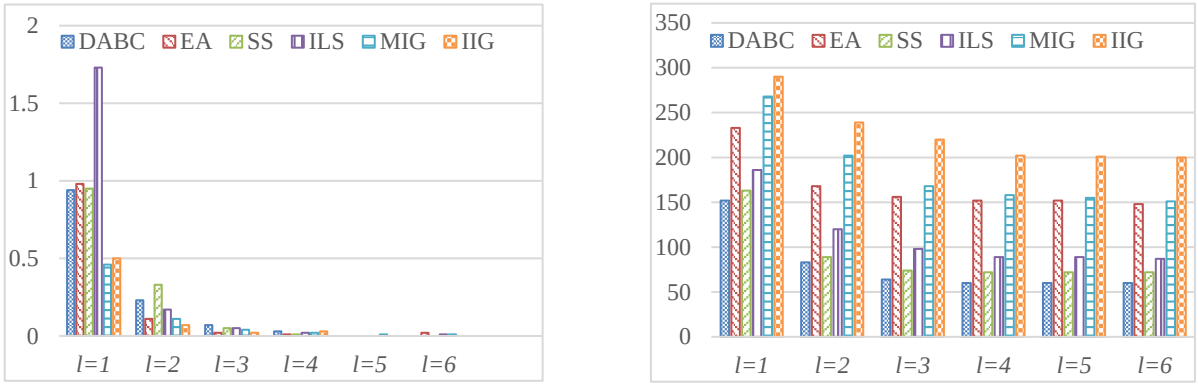
(a) $ARPI_l$ (b) N_l^{alg}

Figure 9. The $ARPI_l$ and N_l^{alg} ($f = 4$) for 30mn milliseconds.



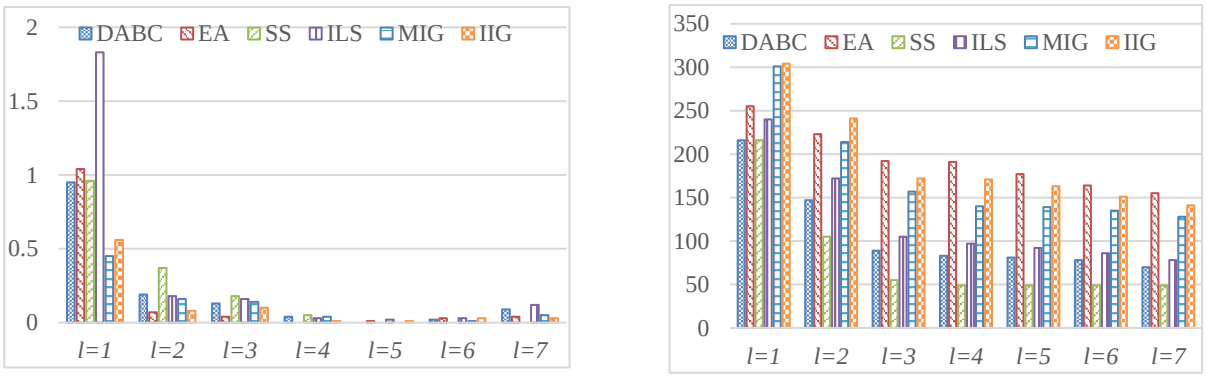
(a) $ARPI_l$ (b) N_l^{alg}

Figure 10. The $ARPI_l$ and N_l^{alg} ($f = 5$) for 30mn milliseconds.



(a) $ARPI_l$ (b) N_l^{alg}

Figure 11. The $ARPI_l$ and N_l^{alg} ($f = 6$) for 30mn milliseconds.



(a) $ARPI_l$ (b) N_l^{alg}

Figure 12. The $ARPI_l$ and N_l^{alg} ($f = 7$) for 30mn milliseconds.

7. Conclusions and future works

This study proposes an improved iterated greedy (IIG) algorithm for solving the distributed permutation flowshop scheduling problem (DPFSP) with minimizing the weighted total completion time of all factories. This problem is a typical multi-factory collaborative scheduling problem, which optimizes the completion time of all factories while considering the task allocation and scheduling sequence of multiple factories simultaneously. It has important theoretical research value and practical application significance. First, the structural characteristics and optimization challenges of the distributed permutation flowshop scheduling problem were thoroughly examined. Building on this analysis, an effective initialization heuristic was designed to generate high-quality initial solutions, thereby establishing a solid foundation for subsequent iterative optimization. To further strengthen the algorithm's local search capabilities, a tailored local search strategy was developed, incorporating problem-specific attributes. By making precise adjustments to the assignment of key processes across factories, this strategy effectively bolsters the algorithm's capacity to escape local optima. To evaluate the effectiveness and superiority of the proposed IIG algorithm, we benchmarked it against several existing representative algorithms. Experimental results on multiple standard test cases demonstrate that IIG achieves superior solution quality and stability in minimizing the weighted total completion time. These findings confirm both the effectiveness of the proposed improvement strategies and the overall advanced performance of the algorithm.

In future work, we will focus on three directions. 1) More objectives for the DPFSP along with utilization of other efficient optimizers will be considered; 2) Multi-objective optimization algorithms for the DPFSP will be designed; 3) Improvement strategies for swarm and evolutionary algorithms, e.g., Q-learning-based local search, will be developed for solving the DPFSP.

Author contributions

Yuan-Zhen Li: Software, investigation, validation, conceptualization, methodology, writing–original draft preparation; Lei-Lei Meng: Writing–review and editing, validation, formal analysis; Biao Zhang: Data curation, investigation, visualization.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This research was partially supported by the National Natural Science Foundation of China 52205529, and the Natural Science Foundation of Shandong Province (ZR2021QE195).

Conflict of interest

The authors declare that there is no conflict of interest.

References

1. K. Gao, Y. Huang, A. Sadollah, L. Wang, A review of energy-efficient scheduling in intelligent production systems, *Complex Intell. Syst.*, **6** (2020), 237–249. https://doi.org/10.1007/s40747-019-00122-6
2. Y. Li, Q. Pan, R. Ruiz, H. Sang, A referenced iterated greedy algorithm for the distributed assembly mixed no-idle permutation flowshop scheduling problem with the total tardiness criterion, *Knowl. Based Syst.*, **239** (2022) 108036. https://doi.org/10.1016/j.knosys.2021.108036
3. L. Meng, W. Cheng, C. Zhang, K. Gao, B. Zhang, Y. Ren, Novel CP models and CP-assisted meta-heuristic algorithm for flexible job shop scheduling benchmark problem with multi-AGV, *IEEE Trans. Syst. Man Cybern. Syst.*, **55** (2025), 8455–8468. https://doi.org/10.1109/TSMC.2025.3604355
4. K. Z. Gao, Z. M. He, Y. Huang, P. Y. Duan, P. N. Suganthan, A survey on meta-heuristics for solving disassembly line balancing, planning and scheduling problems in remanufacturing, *Swarm Evol. Comput.*, **57** (2020), 100719. https://doi.org/10.1016/j.swevo.2020.100719
5. B. Naderi, R. Ruiz, The distributed permutation flowshop scheduling problem, *Comput. Oper. Res.*, **37** (2010), 754–768. https://doi.org/10.1016/j.cor.2009.06.019
6. G. Zhang, K. Xing. Differential evolution metaheuristics for distributed limited-buffer flowshop scheduling with makespan criterion, *Comput. Oper. Res.*, **108** (2019), 33–43. https://doi.org/10.1016/j.cor.2019.04.002
7. Q. Pan, L. Gao, L. Wang, J. Liang, X. Li, Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem, *Expert Syst. Appl.*, **124** (2019), 309–324. https://doi.org/10.1016/j.eswa.2019.01.062
8. V. Fernandez-Viagas, P. Perez-Gonzalez, J. M. Framinan, The distributed permutation flow shop to minimise the total flowtime, *Comput. Ind. Eng.*, **118** (2018), 464–477. https://doi.org/10.1016/j.cie.2018.03.014
9. A. Khare, S. Agrawal, Effective heuristics and metaheuristics to minimise total tardiness for the distributed permutation flowshop scheduling problem, *Int. J. Prod. Res.*, **59** (2021), 7266–7282. https://doi.org/10.1080/00207543.2020.1837982
10. J. Gao, R. Chen, W. Deng, An efficient tabu search algorithm for the distributed permutation flowshop scheduling problem, *Int. J. Prod. Res.*, **51** (2013) 641–651. https://doi.org/10.1080/00207543.2011.644819
11. Y. Xu, L. Wang, S. Wang, M. Liu, An effective hybrid immune algorithm for solving the distributed permutation flow-shop scheduling problem, *Eng. Optimiz.*, **46** (2014), 1269–1283. https://doi.org/10.1080/0305215X.2013.827673
12. S. W. Lin, K. C. Ying, C. Y. Huang, Minimising makespan in distributed permutation flowshops using a modified iterated greedy algorithm, *Int. J. Prod. Res.*, **51** (2013), 5029–5038. https://doi.org/10.1080/00207543.2013.790571
13. V. Fernandez-Viagas, J. M. Framinan, A bounded-search iterated greedy algorithm for the distributed permutation flowshop scheduling problem, *Int. J. Prod. Res.*, **53** (2014), 1111–1123. https://doi.org/10.1080/00207543.2014.948578
14. R. Ruiz, Q. Pan, B. Naderi, Iterated Greedy methods for the distributed permutation flowshop scheduling problem, *Omega*, **83** (2019), 213–222. https://doi.org/10.1016/j.omega.2018.03.004

15. B. Naderi, R. Ruiz, A scatter search algorithm for the distributed permutation flowshop scheduling problem, *Eur. J. Oper. Res.*, **239** (2014), 323–334. https://doi.org/10.1016/j.ejor.2014.05.024
16. H. Bargaoui, O. B. Driss, K. Ghédira, A novel chemical reaction optimization for the distributed permutation flowshop scheduling problem with makespan criterion, *Comput. Ind. Eng.*, **111** (2017) 239–250. https://doi.org/10.1016/j.cie.2017.07.020
17. Y. Fu, Y. Hou, Z. Wang, X. Wu, K. Gao, L. Wang, Distributed scheduling problems in intelligent manufacturing systems, *Tsinghua Sci. Technol.*, **26** (2021), 625–645. https://doi.org/10.26599/TST.2021.9010009
18. N. B. Rad, J. Behnamian, Recent trends in distributed production network scheduling problem, *Artif. Intell. Rev.*, **55** (2022), 2945–2995. https://doi.org/10.1007/s10462-021-10081-5
19. Y. Li, Q. Pan, X. He, H. Sang, K. Gao, X. Jing, The distributed flowshop scheduling problem with delivery dates and cumulative payoffs, *Comput. Ind. Eng.*, **165** (2022), 107961. https://doi.org/10.1016/j.cie.2022.107961
20. J. Pempera, C. Smutnicki, R. Wojcik, Minimizing the cycle time in the distributed flow shop scheduling problem, *IFAC Pap.*, **54** (2021), 1081–1086. https://doi.org/10.1016/j.ifacol.2021.08.203
21. K. Ying, S. Lin, C. Cheng, C. He, Iterated reference greedy algorithm for solving distributed no-idle permutation flowshop scheduling problems, *Comput. Ind. Eng.*, **110** (2017), 413–423. https://doi.org/10.1016/j.cie.2017.06.025
22. K. Ying, S. Lin, Minimizing makespan in distributed blocking flowshops using hybrid iterated greedy algorithms, *IEEE Access*, **5** (2017), 15694–15705. https://doi.org/10.1109/ACCESS.2017.2732738
23. G. Zhang, K. Xing, F. Cao, Discrete differential evolution algorithm for distributed blocking flowshop scheduling with makespan criterion, *Eng. Appl. Artif. Intell.*, **76** (2018), 96–107. https://doi.org/10.1016/j.engappai.2018.09.005
24. K. Karabulut, D. Kizilay, M. F. Tasgetiren, L. Gao, L. Kandiller, An evolution strategy approach for the distributed blocking flowshop scheduling problem, *Comput. Ind. Eng.*, **163** (2022), 107832. https://doi.org/10.1016/j.cie.2021.107832
25. W. Shao, D. Pi, Z. Shao, Optimization of makespan for the distributed no-wait flow shop scheduling problem with iterated greedy algorithms, *Knowl. Based Syst.*, **137** (2017), 163–181. https://doi.org/10.1016/j.knosys.2017.09.026
26. N. N. Zhu, F. Q. Zhao, L. Wang, R. Q. Ding, T. P. Xu, Jonrinaldi, A discrete learning fruit fly algorithm based on knowledge for the distributed no-wait flow shop scheduling with due windows, *Expert Syst. Appl.*, **198** (2022), 116921. https://doi.org/10.1016/j.eswa.2022.116921
27. T. Meng, Q. Pan, L. Wang, A distributed permutation flowshop scheduling problem with the customer order constraint, *Knowl. Based Syst.*, **184** (2019), 104894. https://doi.org/10.1016/j.knosys.2019.104894
28. A. M. Fathollahi-Fard, L. Woodward, O. Akhrif, A distributed permutation flow-shop considering sustainability criteria and real-time scheduling, *J. Ind. Inf. Integr.*, **39** (2024), 100598. https://doi.org/10.1016/j.jii.2024.100598
29. A. M. Fathollahi-Fard, L. Woodward, O. Akhrif, A scenario-based robust optimization model for the sustainable distributed permutation flow-shop scheduling problem, *Ann. Oper. Res.*, 2024. https://doi.org/10.1007/s10479-024-05940-7

30. K. Karabulut, H. Öztop, D. Kizilay, M. F. Tasgetiren, L. Kandiller, An evolution strategy approach for the distributed permutation flowshop scheduling problem with sequence-dependent setup times, *Comput. Oper. Res.*, **142** (2022), 105733. https://doi.org/10.1016/j.cor.2022.105733
31. J. Wang, L. Wang, A knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop, *IEEE Trans. Syst. Man Cybern. Syst.*, **50** (2020), 1805–1819. https://doi.org/10.1109/TSMC.2017.2788879
32. Y. Li, Q. Pan, K. Gao, M. F. Tasgetiren, B. Zhang, J. Li, A green scheduling algorithm for the distributed flowshop problem, *Appl. Soft Comput.*, **10**9 (2021), 107526. https://doi.org/10.1016/j.asoc.2021.107526
33. X. Zhang, X. Liu, A. Cichon, G. Królczyk, Z. Li, Scheduling of energy-efficient distributed blocking flowshop using pareto-based estimation of distribution algorithm, *Expert Syst. Appl.*, **200** (2022), 116910. https://doi.org/10.1016/j.eswa.2022.116910
34. J. Deng, L. Wang, C. Wu, J. Wang, X. Zheng, A competitive memetic algorithm for carbon-efficient scheduling of distributed flow-shop, In: *Intelligent computing theories and application*, Cham: Springer, **977** (2016), 476–488. https://doi.org/10.1007/978-3-319-42291-6_48
35. Y. Fu, G. Tian, A. M. Fathollahi-Fard, A. Ahmadi, C. Zhang, Stochastic multi-objective modelling and optimization of an energy-conscious distributed permutation flow shop scheduling problem with the total tardiness constraint, *J. Clean. Prod.*, **226** (2019), 515–525. https://doi.org/10.1016/j.jclepro.2019.04.046
36. C. Lu, Y. Huang, L. Meng, L. Gao, B. Zhang, J. Zhou, A Pareto-based collaborative multi-objective optimization algorithm for energy-efficient scheduling of distributed permutation flow-shop with limited buffers, *Robot. Comput.-Integr. Manuf.*, **74** (2022), 102277. https://doi.org/10.1016/j.rcim.2021.102277
37. A. M. Fathollahi-Fard, L. Woodward, O. Akhrif, Sustainable distributed permutation flow-shop scheduling model based on a triple bottom line concept, *J. Ind. Inf. Integr.*, **24** (2021) 100233. https://doi.org/10.1016/j.jii.2021.100233
38. J. Deng, L. Wang, A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem, *Swarm Evol. Comput.*, **32** (2017), 121–131. https://doi.org/10.1016/j.swevo.2016.06.002
39. Q. H. Liu, X. Y. Li, L. Gao, G. C. Wang, A multiobjective memetic algorithm for integrated process planning and scheduling problem in distributed heterogeneous manufacturing systems, *Memetic Comp.*, **14** (2022), 193–209. https://doi.org/10.1007/s12293-022-00364-x
40. K. Allali, S. Aqil, J. Belabid, Distributed no-wait flow shop problem with sequence dependent setup time: Optimization of makespan and maximum tardiness, *Simul. Model. Pract. Theory*, **116** (2022), 102455. https://doi.org/10.1016/j.simpat.2021.102455
41. A. P. Rifai, S. T. W. Mara, A. Sudiarso, Multi-objective distributed reentrant permutation flow shop scheduling with sequence-dependent setup time, *Expert Syst. Appl.*, **183** (2021), 115339. https://doi.org/10.1016/j.eswa.2021.115339
42. W. Q. Zhang, C. Li, M. S. Gen, W. D. Yang, Z. W. Zhang, G. H. Zhang, Multiobjective particle swarm optimization with direction search and differential evolution for distributed flow-shop scheduling problem, *Math. Biosci. Eng.*, **19** (2022), 8833–8865. https://doi.org/10.3934/mbe.2022410

43. A. M. Fathollahi-Fard, P. Wu, G. Tian, D. Yu, T. Zhang, J. Yang, et al., An efficient multi-objective adaptive large neighborhood search algorithm for solving a disassembly line balancing model considering idle rate, smoothness, labor cost, and energy consumption, *Expert Syst. Appl.*, **250** (2024), 123908. https://doi.org/10.1016/j.eswa.2024.123908
44. M. E. Baysal, A. Sarucan, K. Buyukozkan, O. Engin, Artificial bee colony algorithm for solving multi-objective distributed fuzzy permutation flow shop problem, *J. Intell. Fuzzy Syst.*, **42** (2022), 439–449. https://doi.org/10.3233/JIFS-219202
45. K. Gao, F. Yang, M. Zhou, Q. Pan, P. N. Suganthan, Flexible job-shop rescheduling for new job insertion by using discrete Jaya algorithm, *IEEE Trans. Cybern.*, **49** (2019), 1944–1955. https://doi.org/10.1109/TCYB.2018.2817240
46. Y. Pan, K. Gao, Z. Li, N. Wu, Solving biobjective distributed flow-shop scheduling problems with lot-streaming using an improved Jaya algorithm, *IEEE Trans. Cybern.*, **53** (2023), 3818–3828. https://doi.org/10.1109/TCYB.2022.3164165
47. R. Ruiz, T. Stützle, A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem, *Eur. J. Oper. Res.*, **177** (2007), 2033–2049. https://doi.org/10.1016/j.ejor.2005.12.009
48. Z. Zhao, M. Zhou, S. Liu, Iterated greedy algorithms for flow-shop scheduling problems: A tutorial, *IEEE Trans. Autom. Sci. Eng.*, **19** (2022), 1941–1959. https://doi.org/10.1109/TASE.2021.3062994
49. Y. X. Pan, K. Z. Gao, Z. W. Li, N. Q. Wu, Improved Meta-Heuristics for solving distributed lot-streaming permutation flow shop scheduling problems, *IEEE Trans. Autom. Sci. Eng.*, **20** (2023), 361–371. https://doi.org/10.1109/TASE.2022.3151648

AIMS Press