A Pareto-Based Estimation of Distribution Algorithm for Solving Multiobjective Distributed No-Wait Flow-Shop Scheduling Problem With Sequence-Dependent Setup Time

Weishi Shao, Dechang Pi, and Zhongshi Shao

***Abstract*— Influenced by the economic globalization, the distributed manufacturing has been a common production mode. This paper considers a multiobjective distributed no-wait flow-shop scheduling problem with sequence-dependent setup time (MDNWFSP-SDST). This scheduling problem exists in many real productions such as baker production, parallel computer system, and surgery scheduling. The performance criteria are the makespan and the total weight tardiness. In the MDNWFSP-SDST, several identical factories are considered with the related flow-shop scheduling problem with no-wait constraints. For solving the MDNWFSP-SDST, a Pareto-based estimation of distribution algorithm (PEDA) is presented. Three probabilistic models including the probability of jobs in empty factory, two jobs in the same factory, and the adjacent jobs are constructed. The PWQ heuristic is extended to the distributed environment to generate initial individuals. A sampling method with the referenced template is presented to generate offspring individuals. Several multiobjective neighborhood search methods are developed to optimize the quality of solutions. The comparison results show that the PEDA obviously outperforms other considered multiobjective optimization algorithms for addressing MDNWFSP-SDST.**

***Note to Practitioners*—This paper is motivated by the process cycles in multiproduction factories (or lines) of baker production, surgery scheduling, and parallel computer systems. In these process cycles, jobs are assigned to multiproduction factories (or lines), and no interruption exists between consecutive operations. This paper models this process as a multiobjective distributed no-wait flow-shop scheduling with SDST. Scheduling becomes more challenging when facing distributed factories. This paper provides an estimation of distributed algorithm with Pareto dominate concept which uses a probabilistic model to generate offspring. Experiment results suggest that the proposed algorithm can find superior solutions of large-scale instances. This scheduling model can be extended to practical problems by considering other constraints, such as assembly process, mixed no-wait, and transporting times. Besides, the proposed algorithm can be applied to solve other distributed scheduling problems and industrial cases, once their constraints are known, i.e., the processing time of operations, the setup time of machines.**

***Index Terms*— Distributed no-wait flow-shop scheduling problem (MDNWFSP), estimation of distribution algorithm (EDA), makespan, multiobjective, sequence-dependent setup time (SDST), total weight tardiness (TWT).**

Manuscript received August 10, 2018; revised December 1, 2018; accepted December 7, 2018. Date of publication January 7, 2019; date of current version July 1, 2019. This paper was recommended for publication by Associate Editor F. Basile and Editor S. Reveliotis upon evaluation of the reviewers' comments. This work was supported in part by the Fundamental Research Funds for the Central Universities under Grant NP2017208, in part by the Funding of Jiangsu Innovation Program for Graduate Education under Grant KYLX16_0382, in part by the Postgraduate Research and Practice Innovation Program of Jiangsu Province under Grant KYCX17_0287, and in part by the National Natural Science Foundation of China under Grant U1433116. *(Corresponding author: Dechang Pi.)*

W. Shao and Z. Shao are with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China (e-mail: shaoweishi@hotmail.com; shaozhongshi@hotmail.com).

D. Pi is with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China, and also with the Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 210093, China (e-mail: nuaacs@126.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2018.2886303

NOMENCLATURE

Parameters

n	Number of jobs.
m	Number of machines.
F	Number of factories.
$p_{i,j}$	Processing time of operation $O_{i,j}$ on machine M_j.
$S_{i,k,j}$	Setup-time of job J_k on machine M_j if job J_i is the immediately preceding job.
d_i	Due date of job J_i.
w_i	Weight of job J_i.

Index

i	Index for jobs where $i = 1, 2, \ldots, n$.
j	Index for machines where $j = 1, 2, \ldots, m$.
f	Index for factories where $f = 1, 2, \ldots, F$.
k	Index for job position in a given sequence $k = 1, 2, \ldots, n$.

Variable

$C_{i,j}$	Continuous variable for the completion time of job J_i on machine M_j.
T_i	Tardiness time of job J_i.
$X_{i,k}$	$\begin{cases} 1, & \text{if } J_k \text{ is processed immediately after job } J_i \\ 0, & \text{otherwise.} \end{cases}$

I. Introduction

THE no-wait constraint in scheduling occurs when two consecutive operations of a job must be processed without any interruption [1]. This scheduling constraint appears in many real production scenarios because it can be employed for all kinds of time-dependent industries such as food, steel manufacturing, and chemical processing industries. As seen in Fig. 1, it shows the common process in a bakery which mainly includes dough production, dough rest, dividing and forming, and proofing and baking. Most of the baking products require proofing and the most commonly used proofing agent is yeast. Then, the proofing is the process of the fermentative decomposition of glucose to CO_2 and other components. This process is strictly time sensitive since the microorganisms get in contact with water and subtrates under preferable conditions of temperature and humidity. Therefore, the basic production in bakeries is commonly subject to a no-wait flow-shop scheduling problem (MDNWFSP) with setup time since there are very small tolerances for waiting time of yeast containing doughs during the production process [2].

However, in today's world, because of the globalization trend and competitive situation, more and more managers realize that traditionally centralized factory cannot be insufficiently flexible to changing production and the requirement from clients and market, so they have changed the single factory to decentralized production centers or distributed factories. These factories or production centers are unnecessary to lie in geographically in the same place or close to each other, and they can lie in different countries or continents. Therefore, the distributed manufacturing has been a common production mode, which can make the factories have effective marketing policies, reduce production cost and management risks, and flexibly adapt to market change. As seen in Fig. 2, we can model the above-mentioned basic bakery production in the distributed environment as a distributed MDNWFSP with setup time. In this distributed scheduling problem, there is a set of F identical factories located at different places, and each factory includes an MDNWFSP with m machines. Jobs need to be assigned to factories and a schedule must be established for each factory. In this paper, we will consider a distributed MDNWFSP with setup time. Certainly, this scheduling model also exists in other real production problems that need to satisfy the no-wait constraint, such as baker production, parallel computer system, and surgery scheduling.

Practical distributed scheduling problems show that the single optimization criterion is inadequate to represent real-world scenarios because scheduling problems in the manufacturing industry typically include multiple conflicting objectives. As a result, a multiobjective optimization problem needs to be addressed. The common objectives used in practical production contain makespan, total flow time, total costs, total tardiness, maximum lateness, and so on. For the distributed scheduling problem with the setup time, a decision-maker generally, first, hopes to minimize the makespan that is equal to the difference between the time when the last job is finished on the final machine and the setup time of the first job on the beginning machine. However, the decision-maker also

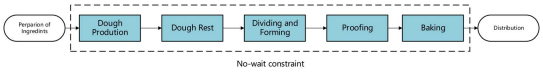


Fig. 1. Basic bakery production model.

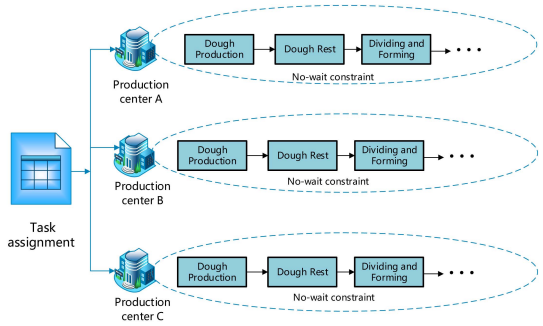


Fig. 2. Basic bakery production model in distributed environment.

concerns with the due date for their customers' satisfaction. The tardiness is the difference between the completion time and the due date which reflects strict requirements on the due date. As we know, it is not always preferable that the completion time is earlier than the due date, since a job completed earlier than its due date is possible to increase the inventory levels and floor space requirements, leading to losses [3]. Therefore, it is necessary to optimize both makespan and total weight tardiness (TWT).

Along with the development of economic globalization, coproduction between companies is more and more common nowadays. As a result, the distributed scheduling and distributed manufacturing systems receive more concern in recent research. So far, the study on distributed scheduling has two directions, one is scheduling models, and the other is optimization methods. For scheduling models, Naderi and Ruiz [4] first proposed distributed permutation flow-shop scheduling (DPFSP), and they proposed two job assignment rules and 12 heuristics to solve this problem. Lin and Ying [5] and Shao *et al.* [6] studied the distributed MDNWFSP and developed several iterated greedy (IG) algorithms to solve this problem. Hatami *et al.* [7], [8] presented the distributed assembly flow-shop scheduling problem, which has two stages: distributed permutation scheduling (production part) and assembly flow-shop scheduling (assembly part). After that, Lin *et al.* [9] proposed a backtracking search hyperheuristic where they designed several heuristic rules to construct the low-level heuristics, and the backtracking search algorithm was used as the high-level strategy. Ribas *et al.* [10] studied the distributed flow-shop scheduling problem with blocking constraint (DBFSP) and employed the iterated local search and the IG algorithm to solve it. Then, Ying and Lin [11] also proposed three version of hybrid IG algorithms to solve the DBFSP. They mixed the IG with variable Tabu list, the constant Tabu list, and the cooling schedule. It should note that, although there is a growing number of research studies in this subject, the studies are still limited to the simple DPFSP concept. Regarding the multiobjective distributed scheduling, the literature is limited. Deng *et al.* [12] presented

a competitive memetic algorithm (CMA) for addressing the multiobjective DPFSP (MODPFSP) with makespan and total tardiness. The CMA uses two populations corresponding to two different objectives, and several objective-specific operators for each population. Later on, Deng *et al.* [13] considered the carbon-efficient constraint in the DPFSP. They used CMA to address the MODPFSP with the objective of minimizing makespan and total carbon emissions. Cai *et al.* [14] studied the multiobjective distributed DPFSP with transportation and eligibility constraints. They considered three objectives that are makespan, maximum lateness, and the total costs of transportation costs and setup costs and proposed an improve NSGAII algorithm to solve this problem. Rifai *et al.* [15] presented a multiobjective distributed reentrant permutation flow-shop scheduling problem where a set of jobs with several reentrant layers are processed in the factories. Three objectives, i.e., the minimization of makespan, total cost, and average tardiness, are simultaneously taken account. Since most of distributed scheduling problems are NP-hard problems, the effectiveness of optimization methods plays a critical role in solving problems. Now, the optimization methods for distributed scheduling are still limited. Some heuristics and metaheuristics have been proposed, such as Nawaz–Enscore–Ham (NEH) with branch and bound [16], variable neighborhood search [17], electron-magnetism metaheuristics [18], genetic algorithm (GA) [19], [20], scatter search [21], Tabu search [22], IG [23], hybrid immune algorithm (HIA) [24], estimation of distribution algorithm (EDA) [25], and chemical reaction optimization [26].

The EDA [27] is regarded as a statistics theory-based metaheuristic. The EDA, first, selects elite individuals from the population and, then, builds a probabilistic model by extracting the statistical information from elite individuals, finally, uses the probabilistic model to generate offspring individuals. Due to the good global search ability of the EDA, it has been used to address the amount of scheduling problems [27]–[31]. However, the EDA for multiobjective scheduling problems is very limited. Tiwari *et al.* [32] proposed a Pareto block-based EDA which employs a bivariate probabilistic model to produce blocks of jobs, and the nondominated sorting method was used to distinguish the solutions. Wang *et al.* [33] used the Pareto elite individuals to estimate the probabilistic information of solution space, and the population was separated into two subpopulations based on a splitting criterion. Several operators based on different optimization objectives were developed for these two subpopulations to produce the offspring. Hao *et al.* [34] proposed a multiobjective EDA to solve the bicriteria stochastic job-shop scheduling problem with the uncertainty of processing time. Wang *et al.* [35] presented a Pareto-archived EDA for addressing the multiobjective resource-constrained project scheduling problem with the objective of minimizing makespan and resource investment criteria.

According to the aforementioned investigation, we think there exist following directions for further researchers.

1) Although many distributed scheduling problems have been investigated, more realistic constraints need to be considered to meet real production. This paper considers a multiobjective distributed MDNWFSP which widely exists in multiproduction factories or lines of bakery production, surgery scheduling, and so on.
2) Several metaheuristics are developed to improve computational time or solution quality of distributed flow-shop scheduling problems. However, still more researchers need to obtain a desirable result for different types of distributed scheduling problems.

Motivated by the application of the above-mentioned constraints in realistic production practice, this paper proposes a multiobjective distributed MDNWFSP with sequence-dependent setup time (MDNWFSP-SDST), whose optimization objectives are makespan and TWT.

Compared with previous studies on distributed flow-shop scheduling problems, the contribution of this paper can be summarized as follows. The MDNWFSP-SDST is proposed according to the real-time scenario of distributing scheduling. To the best of our knowledge, this paper is the first attempt to solve the MDNWFSP-SDST. Previous studies on the distributed scheduling problems do not consider all of no-wait constraint, multiobjective, and setup time. Moreover, we establish the mixed integer linear programming (MILP) model of MDNWFSP-SDST. A Pareto-based EDA (PEDA) is proposed, which provides a new direction for using the EDA to solve distributed scheduling problems. First, most of previous EDAs for distributed scheduling problems only consider jobs position information, but this paper considers the probability of jobs in empty, jobs in common factory, and two adjacent jobs. Second, we propose a sample method with a referenced template which does not appear in previous EDAs that are used to solve distributed scheduling problems. Third, we propose the multiobjective local search. Although these neighborhood searches may have existed in several methods, these neighborhood searches are not used for multiobjective optimization problem, and these neighborhood search greatly enhance the local searching ability. Furthermore, we compare the proposed PEDA with other multiobjective optimization algorithms, the experimental results show the PEDA can solve the MDNWFSP-SDST effectively and efficiently.

The remainder of this paper is organized as follows. Section II shows the detail of MDNWFSP-SDST. Section III presents each component of PEDA. Section IV calibrates the parameters of PEDA and carries out the comparison results. Section V presents the conclusion and future study.

II. Multiobjective Distributed No-Wait Flow-Shop Scheduling Problem With Sequence-Dependent Setup Time

A. Basic Definition of Multiobjective Optimization Problem

To better explain the proposed MDNWFSP-SDST, we give a brief introduction of basic concept of a multiobjective problem. Generally, a multiobjective problem can be defined as follows (for minimization problem):

$$\min\ f(\boldsymbol{X}) = \min[f_1(\boldsymbol{X}), f_2(\boldsymbol{X}), \ldots, f_m(\boldsymbol{X})] \quad (1)$$

$$\boldsymbol{X} = (x_1, x_2, \ldots, x_n) \in R^n \quad (2)$$

$$\text{s.t.} \begin{cases} g_i(\boldsymbol{X}) \geq 0 & i = 1, \ldots, k \\ h_i(\boldsymbol{X}) = 0 & j = 1, \ldots, p \end{cases} \tag{3}$$

where $f_m(\boldsymbol{X})$ denotes the mth subobjective function. $\boldsymbol{X}$ is a solution, which should satisfy the above-mentioned constraints. R^n is the decision variable space. k and p represent the number of equality and inequality constraints, respectively.

1) Pareto Dominance: Let $\boldsymbol{a}$ and $\boldsymbol{b}$ denote two feasible solutions. The solution $\boldsymbol{a}$ dominates the solution $\boldsymbol{b}$ (denoted by $\boldsymbol{a} \prec \boldsymbol{b}$) if and only if $f_i(\boldsymbol{a}) \leq f_i(\boldsymbol{b}), \forall i \in \{1, 2, \ldots, m\}$ and $f_l(\boldsymbol{a}) < f_l(\boldsymbol{b}), \exists l \in \{1, 2, \ldots, m\}$.

2) Pareto Optimal Solution: A solution $\boldsymbol{X}^* \in R^n$ is a Pareto optimal solution if there does not exist any other solution which dominates $\boldsymbol{X}^*$. The corresponding objective function is called the Pareto optimal front vector $f(\boldsymbol{X}^*)$.

The set of all Pareto optimal solutions is known as Pareto set, and the set of all Pareto optimal front vectors is known as the Pareto optimal front ($\boldsymbol{PF}$). The purpose of multiobjective optimization is to find $\boldsymbol{PF}$.

B. Problem Description

The MDNWFSP-SDST is described as follows: a set of jobs $\{J_1, J_2, \ldots, J_n\}$ are available at time zero. They have been allocated to F factories $\{Fa_1, Fa_2, \ldots, Fa_F\}$ and be processed in each factory with the same processing order. Each factory contains an identical flow shop with m machines $[M_1, M_2, \ldots, M_m]$. Each job contains m processing operations $[O_{i,1}, O_{i,2}, \ldots, O_{i,m}]$. When a job has been allocated to a certain factory, all its operations should be processed only in this factory. All operations of each job are no-wait constrained. That is to say, once processing a specific job is started, the job should be processed by the successive machines with no interruption between the consecutive operations. Every operation should not be stopped once it began. No job can be processed on more than one machine, and no machine can process more than one job at a time. The transit time between operations is ignored. The setup process has to be executed between the completion of one job and the start of another job on each machine. Here, we consider the SDSTs which depend on the current job being processed and on the next job in the sequence. The aim of MDNWFSP-SDST is to reasonably allocate jobs to factories and determine the processing order of jobs in each factory to optimize multiple scheduling objectives. The makespan (denoted by $C_{\max}$) and the TWT criteria are to be minimized in this paper. The makespan is production-oriented criterion, but it neglects an important aspect of production which is client satisfaction. The TWT has been used to satisfy different customer demand. These two objectives are not optimized simultaneously since the makespan is the main priority as long as the due date constraint is satisfied. Therefore, the MDNWFSP-SDST can be defined as multiobjective mathematical model with the objective of minimization of makespan and TWT.

A mathematical model is an abstract and good approach that uses mathematical language to describe a problem in detail. Inspired by the fifth mathematical model proposed in [4] for the DPFSP, we propose the MILP model of MDNWFSP-SDST. The following notations are used to describe the MILP of MDNWFSP-SDST.

Since we use sequenced-based variables, dummy jobs 0 need to be defined. The MDNWFSP-SDST can be formulated as follows.

Objective Function

$$\min(f_1, f_2), f_1 = C_{\max}, f_2 = \sum_{i=1}^{n} w_i \times T_i \tag{4}$$

$$\text{s.t.} \sum_{i=0, i\neq k}^{n} X_{i,k} = 1 \quad \forall k \tag{5}$$

$$\sum_{k=0, i\neq k}^{n} X_{i,k} \leq 1 \quad \forall k \tag{6}$$

$$\sum_{k=1}^{n} X_{0,k} = F \tag{7}$$

$$\sum_{i=1}^{n} X_{i,0} = F - 1 \tag{8}$$

$$X_{i,k} + X_{k,i} \leq 1 \quad \forall i \in \{1, 2, \ldots, n-1\}, \; k > i \tag{9}$$

$$C_{k,j} = C_{k,j-1} + p_{k,j} \quad \forall i, j, k \tag{10}$$

$$C_{k,j} \geq C_{i,j} + p_{k,j} + s_{i,k,j} + (X_{i,k} - 1) \cdot M \quad \forall i, k, j, k \neq i \tag{11}$$

$$C_{\max} \geq C_{i,m} \quad \forall j \tag{12}$$

$$T_i \geq C_{i,m} - d_i \quad \forall i \tag{13}$$

$$C_{i,j} \geq 0 \quad \forall i, j \tag{14}$$

$$T_i \geq 0 \quad \forall i \tag{15}$$

$$X_{i,k} \in \{0, 1\} \quad \forall i, k, \; i \neq k. \tag{16}$$

The objective function, as presented in (4), is minimization of makespan (f_1) and TWT (f_2). Constraint sets (5) guarantee that every job must have exactly one processor. Constraint sets (6) indicate that each job has at most one succeeding job. Constraint sets (7) guarantee that dummy job 0| appears F times in the sequence as a predecessor. Constraint sets (8) ensure that dummy job 0 should be a successor $F - 1$ times. Constraint sets (9) ensure that a job cannot be both predecessor and successor of another job simultaneously. Constraint sets (10) state that the starting of processing a job on a machine is exactly to the completion time of processing the job on the preceding machine. This constraint ensures the no-wait constraint. Constraint sets (11) indicate the difference between the completion times of two consecutive jobs on a machine is larger than the summation of the setup time and processing time of the job processed as the later job. Constraint sets (12) define the maximum completion time of the whole process. Constraint sets (13) formulate the tardiness of every job. Constraint sets (14) and (15) guarantee that the completion time of each job on each machine and the tardiness of each job should be nonnegative. Constraint sets (16) define the binary decision variables.

In order to distinguish our model from the DPFSP model of [4], we describe their differences as follows. First, the objectives of MDNWFSP-SDST are $C_{\max}$ and TWT [see constraint sets (4), (12), (13), (14), and (15)], the objective of

DPFSP is $C_{\max}$. Then, the MDNWFSP-SDST should satisfy the no-wait constraint [see constraint sets (10)] that is the starting of processing a job on a machine is exactly equal to the completion time of processing the job on the preceding machine. The DPFSP only ensures that, for every job, the current operation cannot begin before the preceding operation completes. Third, the MDNWFSP-SDST considers the setup time [see constraint sets (11)], while the DPFSP ignores the setup time.

Since the MDNWFSP-SDST can be regarded as an extension of the no-wait permutation flow-shop scheduling for makespan and total tardiness, the latter problem is known to be a strongly NP-hard problem for $m \geq 3$ when the objectives are the makespan and the total tardiness [36], [37]. Moreover, in any case, since the MDNWFSP-SDST can reduce to the classical no-wait permutation flow-shop scheduling problem if $F = 1$, therefore, the MDNWFSP-SDST is an NP-complete problem with the objective of makespan and total tardiness if $n > F$ and $m \geq 3$.

C. Computing Makespan and Total Weight Tardiness

Let $\pi = [\boldsymbol{\pi}^1, \boldsymbol{\pi}^2, \ldots, \boldsymbol{\pi}^F]$ be a given feasible schedule, where $\boldsymbol{\pi}^f = [\pi^f(1), \pi^f(2), \ldots, \pi^f(n^f)]$ is the processing order of jobs in factory f.n^f is the number of jobs belonging to factory f. According to [38], the no-wait constraint has a feature that the completion time of consecutive jobs $J_{i-}1$ and J_i is fixed when the processing time and the setup time is predefined. Note that a dummy job 0 needs to be defined for each factory, whose processing time is zero. Let $D_{\pi^f(i-1),\pi^f(i)}$ denote the difference of completion time between consecutive jobs $\pi^f(i-1)$ and $\pi^f(i)$.$D_{\pi^f(i-1),\pi^f(i)}$ can be computed as follows:

$$D_{\pi^f(i-1),\pi^f(i)} = \max_{j=1,2,\ldots,m} \left\{ \sum_{p=j}^{m} (t_{\pi^f(i),p} - t_{\pi^f(i-1),p}) + t_{\pi^f(i-1),j} + S_{\pi^f(i-1),\pi^f(i),j} \right\}. \tag{17}$$

Next, the makespan $C_{\max}(\boldsymbol{\pi})$ is computed as follows:

$$C_{\max}(\boldsymbol{\pi}^f) = \sum_{i=0}^{n^f-1} D_{\pi^f(i),\pi^f(i+1)}, \quad f = 1, 2, \ldots, F \tag{18}$$

$$C_{\max}(\boldsymbol{\pi}) = \max\{C_{\max}(\boldsymbol{\pi}^f)\}, \quad f = 1, 2, \ldots, F. \tag{19}$$

TWT($\boldsymbol{\pi}$) is computed as (20) and (21)

$$C_{\pi^f(i),m} = \sum_{k=0}^{i-1} D_{\pi^f(k),\pi^f(k+1)} \tag{20}$$

$$T_{\pi^f(i)} = C_{\pi^f(i)} - d_{\pi^f(i)}, \quad f = 1, 2, \ldots, F, \quad i = 1, \ldots, n^f \tag{21}$$

$$\mathrm{TWT}(\boldsymbol{\pi}) = \sum_{i=1}^{n} w_i \times T_i. \tag{22}$$

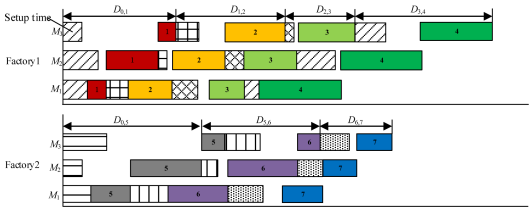


Fig. 3. Example of MDNWFSP-SDST.

The purpose of the MDNWFSP-SDST is to find a scheduling $\boldsymbol{\pi}^*$ with the minimization of makespan and TWT. According to [38], $D_{\pi^f(i-1),\pi^f(i)}$ is only closely related to $\pi^f(i-1)$ and $\pi^f(i)$. In order to decrease the evaluation time in searching process, we can compute $D_{\pi(i-1),\pi(i)}$ for each pair of jobs in the initial phase.

In order to illustrate the MDNWFSP-SDST clearly, we show an example including seven jobs and two factories. The processing sequences of jobs in two factories are $\boldsymbol{\pi}^1 = [1, 2, 3, 4]$ and $\boldsymbol{\pi}^2 = [5, 6, 7]$. The Gantt chart of $\boldsymbol{\pi}^1$ and $\boldsymbol{\pi}^2$ is displayed in Fig. 3, where $D_{0,1}$ represents the difference of completion time between dummy jobs 0 and 1, $D_{1,2}$ denotes the difference of completion time between jobs 1 and 2, the rest D can be done in the same manner. It can clearly be seen from Fig. 3, $C_{\max}(\boldsymbol{\pi}^1) = D_{0,1} + D_{1,2} + D_{2,3} + D_{3,4}$ and $C_{\max}(\boldsymbol{\pi}^2) = D_{0,5} + D_{5,6} + D_{6,7}$.

III. Pareto-Based Estimation of Distribution Algorithm

The EDA is a metaheuristic that collects the distribution information of candidate solutions via sampling a probabilistic model established from promising solutions found so far [39]. With statistical analysis technique, the EDA makes the population track the promising search areas based on the available experience. The key step of EDA is the construction of probabilistic model. Since there exist different types of problems, the suitable probabilistic models and updating mechanism need to be constructed in order to estimate the distribution of solutions. The probabilistic model can reduce the destruction of building blocking and enhance the global exploration, but the local exploitation of EDA is still limited. For an effective optimization algorithm, it can make a tradeoff between the global exploration and local exploitation. In this section, we will propose a PEDA to address the MDNWFSP-SDST.

A. Achieve Set AS

In this paper, we use an achieve set (**AS**) to store all Pareto optimal solutions that construct the Pareto optimal front in the objective space. The achieve set is iteratively updated during the searching process. For a given solution, if it is dominated by any solution in **AS**, we discard this solution. Otherwise, we remove all the solutions which are dominated by this solution and add this solution to **AS**.

B. Solution Representation

The complete representation designed by Naderi and Ruiz [4] is used in this paper, which can represent all possible solutions. There are F lists in

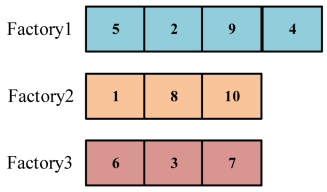


Fig. 4. Example of solution representation.

the solution representation, and every list has a partial permutation that is the processing order of jobs at each factory. In Fig. 4, we give an example with 10 jobs and 3 factories. The operation sequence of jobs for factories 1, 2, and 3 is $5 \rightarrow 2 \rightarrow 9 \rightarrow 4, 1 \rightarrow 8 \rightarrow 10$, and $6 \rightarrow 3 \rightarrow 7$, respectively.

C. Population Initialization

In this paper, we improve a biobjective heuristic PWQ presented by Pan *et al.* [40], it is adopted to address the MDNWFSP in single factory. The PWQ first generates two job sequences $\boldsymbol{\sigma}^P$ and $\boldsymbol{\sigma}^E$ by sorting jobs in descending sums of their total processing time and the EDD heuristic [41] and, then, computes the weighted sum of job position value $\phi_j = w_1 \times \varphi_j(\boldsymbol{\sigma}^P) + w_2 \times \varphi_j(\boldsymbol{\sigma}^E)$, $j = 1, 2, \ldots, n$.$\varphi_j(\boldsymbol{\sigma}^P)$ and $\varphi_j(\boldsymbol{\sigma}^E)$ denote the position values of job j in $\boldsymbol{\sigma}^P$ and $\boldsymbol{\sigma}^E$.w_1 and w_2 denote the weight of f_1 and f_2, which are decided by the population size (PS) and the index l of current individual, i.e., (w_1, w_2) as $(l/(\mathrm{PS}-1), 1-(l/(\mathrm{PS}-1)))$. Next, a sequence $\boldsymbol{\sigma}^0$ is produced by sorting the jobs in ascending weighted sum of job position value. Finally, each job in $\boldsymbol{\sigma}^0$ is inserted into a new sequence by using the insertion procedure of NEH [42]. When evaluating the quality of intermediate sequence $\boldsymbol{\sigma}$, the PWQ heuristic employs a weighted sum to combine f_1 and f_2, as shown in the following equation:

$$f(\boldsymbol{\sigma}) = w_1 \times f_1(\boldsymbol{\sigma}) + w_2 \times f_2(\boldsymbol{\sigma}). \tag{23}$$

However, (23) is not reasonable to all objectives that have different orders of magnitude or units. In order to maintain consistency, we normalize the objectives, as shown in the following equation:

$$f(\boldsymbol{\sigma}) = w_1 \times \frac{f_1(\boldsymbol{\sigma}) - f_1^{\min}}{f_1^{\max} - f_1^{\min}} + w_2 \times \frac{f_2(\boldsymbol{\sigma}) - f_2^{\min}}{f_2^{\max} - f_2^{\min}} \tag{24}$$

where $f_1^{\min}$ and $f_1^{\max}$ denote the minimum and the maximum $C_{\max}$ of all current subsequences that are generated by inserting a job to $\boldsymbol{\sigma}$.$f_2^{\min}$ and $f_2^{\max}$ denote the minimum and the maximum TWT of all current subsequences that are generated by inserting a job to $\boldsymbol{\sigma}$. We extend PWQ heuristic to distributed flow-shop scheduling, shown as *Pseudocode 1*. First, the improved PWQ generates two job sequences $\boldsymbol{\sigma}^P$ and $\boldsymbol{\sigma}^E$ by sorting jobs in descending sums of their processing time and due date. Second, the weighted sum of job position value of each job is computed, and then a permutation $\boldsymbol{\sigma}^0$ by sorting jobs in ascending weighted sum of their position values is generated. Third, the first F jobs of $\boldsymbol{\sigma}^0$ are extracted and inserted into each factory of new sequence $\boldsymbol{\pi}$, i.e., $\pi^f(1) = \sigma^0(f)$, $f = 1, 2, \ldots, F$, and next, the rest jobs of $\boldsymbol{\sigma}^0$ are inserted into all possible positions of $\boldsymbol{\pi}$, the position resulting in the minimum weighted sum of f_1 and f_2 is selected. Different from the original version of PWQ, the improved PWQ uses (24) to normalize the objectives to maintain consistency, and the jobs are assigned to all possible positions of all factories by using the insertion of procedure of NEH.

Pseudocode 1 Improved PWQ

Input: The population size PS
Output: The initial population **POP**

1: Generate a job sequence $\boldsymbol{\sigma}^P$ by sorting jobs in descending sums of their processing times.
2: Generate a job sequence $\boldsymbol{\sigma}^E$ by sorting jobs in descending sums of their due date. Let $l = 1$.
3: **while** $l \leq$ PS **do**
4: Set a weight vector (w_1, w_2) as $(l/(\mathrm{PS}-1), 1-(l/(\mathrm{PS}-1)))$.
5: Calculate the weighted sum of job position values $\phi_j = w_1 \times \varphi_j(\boldsymbol{\sigma}^P) + w_2 \times \varphi_j(\boldsymbol{\sigma}^E)$ for $j = 1, 2, \cdots, n$, where $\varphi_j(\boldsymbol{\sigma}^P)$ and $\varphi_j(\boldsymbol{\sigma}^E)$ denote the position values of job j in permutation $\boldsymbol{\sigma}^P$ and $\boldsymbol{\sigma}^E$.
6: Generate a permutation $\boldsymbol{\sigma}^0$ by sorting jobs in ascending weighted sum of their position values ϕ_j. Let $\boldsymbol{\pi}$ represent the new generated individual.
7: The first F jobs $[\sigma^0(1), \sigma^0(2), \cdots, \sigma^0(F)]$ of $\boldsymbol{\sigma}^0$ are taken and let $\pi^f(1) = \sigma^0(f)$, $f = 1, 2, \cdots, F$.
8: Take job $\sigma^0(k)$, $k = F+1, F+2, \cdots, n$, insert it into all possible positions of $\boldsymbol{\pi}$, and the position resulting in the minimum weighted sum of the two objectives values $f(\pi)$ is selected. Insert $\sigma^0(k)$ to this position.
9: **POP(l)** $= \boldsymbol{\pi}$, $l = l + 1$
10: **end while**

D. Probabilistic Model and Updating Mechanism

Different from the GA which generates offspring individual by crossover and mutation operators, the EDAs generate offspring via sampling a probabilistic model which greatly influences the performance of EDAs. Therefore, it is critical to design an effective probabilistic model for EDA. This section proposes a hybrid probabilistic model to describe the promising search areas of MDNWFSP-SDST. According to the characteristic of different problems, different probabilistic models should be established to estimate the reasonable promising probability distribution. Since the distributed no-wait flow-shop scheduling has two tasks: assigning jobs to factory and determining job sequence in the factory, we consider three kinds of position relationship between jobs and factories, i.e., the jobs in empty factory, the two jobs in same factory, and the adjacent jobs. Since the factories are identical, two solutions have identical job sequences but the sequences exist in different factories. In fact, these two solutions are same and they have the same makespan and TWT. Therefore, the above-mentioned three kinds of relationship could reduce the complexity of establishing model. In the following paragraphs, we establish three probabilistic models to describe the solution space.

1) The probability $\boldsymbol{PE}$ of jobs in empty factory, donated as (25), where pe_i denotes the probability of job i in

empty factory, i.e., the probability of jobs in the first position of any factory. pe_i can be computed as (26), where $IE_i(l)$ is an indicator function which shows whether the job i appears in the first position of any factory in the lth individual in **AS**, and Asize represents the size of **AS**. For all values of i, pe_i is initialized to $1/(Fn)$, which ensures that the whole solution space can be sampled uniformly

$$\boldsymbol{PE} = [pe_1, pe_2, \ldots, pe_n] \tag{25}$$

$$pe_i = \frac{\sum_{l=1}^{\text{Asize}} IE_i(l)}{Asize} \tag{26}$$

$$IE_i(l) = \begin{cases} 1, & \text{job } i \text{ in the first position of any factory} \\ & \text{in the } l\text{-th individual in A}\mathbf{S} \\ 0, & \text{otherwise.} \end{cases} \tag{27}$$

2) The probability $\boldsymbol{PC}$ of two jobs in the same factory, denoted as (28), where $pc_{i,i'}$ denotes the probability of job i and job i' in the same factory. $pc_{i,i'}$ can be computed as (29), where $IC_{i,i'}(l)$ is an indicator function which is equal to 1 when job i and job i' appears in the same factory in the lth individual in **AS**. For all values of i and i', $pc_{i,i'}$ is initialized to $1/(n^2)$

$$\boldsymbol{PC} = \begin{bmatrix} pc_{1,1} & pc_{1,2} & \cdots & pc_{1,n} \\ pc_{2,1} & pc_{2,2} & \cdots & pc_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ pc_{n,1} & pc_{n,2} & \cdots & pc_{n,n} \end{bmatrix} \tag{28}$$

$$pc_{i,i'} = \frac{\sum_{l=1}^{\text{Asize}} IC_{i,i'}(l)}{\sum_{i'=1}^{n} \sum_{l=1}^{\text{Asize}} IC_{i,i'}(l)} \tag{29}$$

$$IC_{i,i'}(l) = \begin{cases} 1, & \text{job } i \text{ and job } i' \text{ appear in the same} \\ & \text{factory in the } l\text{-th individual in } \mathbf{AS} \\ 0, & \text{otherwise.} \end{cases} \tag{30}$$

3) The probability $\boldsymbol{PA}$ of two adjacent jobs, denoted as (31), where $pa_{i,i'}$ represents the probability that job i' appears immediately after the job i. $pa_{i,i'}$ can be computed as (32), where $IA_{i,i'}(l)$ is an indicator function which is equal to 1 when job i' appears immediately after the job j in the lth individual in **AS**. For all values of i' and i, $pa_{i,i'}$ is initialized to $1/(n^2)$

$$\boldsymbol{PA} = \begin{bmatrix} pa_{1,1} & pa_{1,2} & \cdots & pa_{1,n} \\ pa_{2,1} & pa_{2,2} & \cdots & pa_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ pa_{n,1} & pa_{n,2} & \cdots & pa_{n,n} \end{bmatrix} \tag{31}$$

$$pa_{i,i'} = \frac{\sum_{l=1}^{\text{Asize}} IA_{i,i'}(l)}{\sum_{i'=1}^{n} \sum_{l=1}^{\text{Asize}} IA_{i,i'}(l)} \tag{32}$$

$$IA_{i,i'}(l) = \begin{cases} 1, & \text{job } i' \text{ appears immediately after job } i \\ & \text{in } l\text{-th individual in } \mathbf{AS} \\ 0, & \text{otherwise.} \end{cases} \tag{33}$$

The probabilistic model should be adjusted in each generation to track the promising searching region. The Heb rule is used to update probabilistic models, shown as (34)–(36), where t represents the number of generation, $\alpha \in (0, 1)$ is a constant learning rate of probabilistic models

$$\boldsymbol{PE}(t+1) = \alpha \cdot \boldsymbol{PE}(t-1) + (1-\alpha) \cdot \boldsymbol{PE}(t) \tag{34}$$

$$\boldsymbol{PC}(t+1) = \alpha \cdot \boldsymbol{PC}(t-1) + (1-\alpha) \cdot \boldsymbol{PC}(t) \tag{35}$$

$$\boldsymbol{PA}(t+1) = \alpha \cdot \boldsymbol{PA}(t-1) + (1-\alpha) \cdot \boldsymbol{PA}(t). \tag{36}$$

E. Sample With Referenced Template

Sampling the probabilistic model is one of the key steps in the EDA. From [43], it can be known that the EDA has a shortcoming which is that occasionally solutions generated by sampling that are not very representative of the current probabilistic model, but nevertheless are good, are not fully exploited in the future iterations. To deal with this limitation, we present a sampling method with reference template according to the literature [44]–[47]. This sampling method first randomly selects one solution from the population as the referenced template individual. Then, d jobs are randomly selected from the referenced individual regardless of factories, and the rest jobs are copied to a new solution. Finally, the selected d jobs are inserted to the new solution according to the probabilistic model. During the procedure of inserting a job, we first decide which factory the job should be placed in and, then, ensure which position the jobs should be inserted into this factory. The procedure of sample with referenced template is shown as *Pseudocode* 2.

F. Multiobjective Local Search

The purpose of solving multiobjective optimization problems is to find a Pareto front where the set of nondominated solutions is as close as possible to the true Pareto front and at the same time distributed as evenly as possible [48]. Since the local exploitation of EDA is limited, this section proposes five neighborhood search methods to improve the quality of solutions, which employs the operator of insert and swap move between factories or in the factory. Due to the multiobjectives, we keep all the solutions in the searching procedure, and these solutions are added into a temporary nondominated set. For a given solution to be improved, the neighborhood search methods can be described as follows.

1) *Pareto Insert Search:* First, every job is taken out from its original position successively, then reinserted all possible positions of the sequences of all factories. During insertion, all the solutions are stored. This procedure continues until all jobs are checked. After insertion, all the solutions generated by this procedure are added into a nondominated set, and return this nondominated set.
2) *Pareto Swap Search:* Each job is exchanged with other jobs of all factories. During swapping two jobs, all the solutions are kept. This procedure continues until all jobs are checked. All solutions generated by this procedure are added into a nondominated set and return this nondominated set.

Pseudocode 2 Sample With Referenced Template

Input: the number of selected jobs d, a referenced template individual $\boldsymbol{\pi}_r$ from **POP**, $\boldsymbol{PE}$, $\boldsymbol{PC}$,$\boldsymbol{PA}$.
Output: A new individual $\boldsymbol{\pi}$.

randomly select d jobs from $\boldsymbol{\pi}_r$ and add them into $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \cdots, \sigma_d]$.
copy the rest jobs in $\boldsymbol{\pi}_r$ to $\boldsymbol{\pi}$.
for each job σ_i **do**
  for $f = 1$ to F **do**
    $p = 1.0$
    for $j = 1$ to n^f **do**
      $p = p \times pc_{\pi^f(j),\sigma_i}$
    end for
    record the probability of σ_i in factory f, i.e. $RecordP[f] = p$.
  end for
  select the factory with the maximum probability in $\boldsymbol{RecordP}$, and denoted as f_r.
  Let $RecordP'[1] = pe_{\sigma_i}$
  for $j = 2$ to n^{f_r} **do**
    $RecordP'[j] = pa_{\pi^{f_r}(j),\sigma_i}$
  end for
  select the position $recordpos$ with the maximum probability in $\boldsymbol{RecordP'}$
  insert σ_i into the position $recordpos$ in factory f_r in $\boldsymbol{\pi}$.
end for

3) *Pareto Insert Search for Tardiness:* The job with the maximum tardiness is taken out from its original position, and then reinserted all possible positions of the sequence of all factories. During inserting a job, all of the solutions are stored. This procedure continues until all jobs are checked. All solutions generated by this procedure are added into a nondominated set and return this nondominated set.
4) *Pareto Swap Search for Tardiness:* The job with the maximum tardiness is exchanged with other jobs of all factories. During exchanging two jobs, all the solutions are kept. This procedure continues until all jobs are checked. All solutions generated by this procedure are added into a nondominated set and return this nondominated set.
5) *Pareto Insert Search for Critical Factory:* The critical factory refers to the factory with maximum makespan. Each job is removed in the critical factory from its original position successively, and then reinserted all possible positions of the sequences of all factories. During insertion, all of the solutions are stored. This procedure continues until all jobs are checked. All solutions generated by this procedure are added into a nondominated set and return this nondominated set.

In order to apply the above-mentioned neighborhood search methods, we present a multiobjective local search algorithm to optimize the solutions in the archive set and new generated solutions. If a solution $\boldsymbol{s}$ is fully explored by the above-mentioned neighborhood search methods, the solution is marked. The solutions in the returned nondominated set which are not weakly dominated by $\boldsymbol{s}$ or by any solution in the archive set are added to the archive set. The solutions in the **AS** dominated by the newly added solution are removed. We also save all solutions obtained by multiobjective local search in **LPOP**. The procedure of multiobjective local search is shown as *Pseudocode* 3.

Pseudocode 3 Multiobjective Local Search

Input: A solution $\boldsymbol{s}$, the achieve set **AS**, the population generated by multiobjective local search **LPOP**.
Output: AS, **LPOP**

if $\boldsymbol{s}$ is not visited **then**
  $NSet_1$ = *Pareto Insert Search* ($\boldsymbol{s}$)
  $NSet_2$ = *Pareto Swap Search* ($\boldsymbol{s}$)
  $NSet_3$ = *Pareto Insert Search for Tardiness* ($\boldsymbol{s}$)
  $NSet_4$ = *Pareto Swap Search for Tardiness* ($\boldsymbol{s}$)
  $NSet_5$ = *Pareto Insert Search for Critical Factory* ($\boldsymbol{s}$)
  Mark $\boldsymbol{s}$ as being visited.
  for $\boldsymbol{s}' \in \{Nset_1, Nset_2, Nset_3, Nset_4, Nset_5, \boldsymbol{s}\}$ **do**
    if $\boldsymbol{s}'$ is not dominated by any solution in **AS then**
      Add $\boldsymbol{s}'$ into **AS**, and delete the solutions in **AS** that are dominated by $\boldsymbol{s}'$.
      Add $\boldsymbol{s}'$ into **LPOP**.
    end if
  end for
end if

G. Framework of PEDA

To make the EDA applicable to solve MDNWFSP-SDST under consideration, this paper proposes a PEDA. The framework of PEDA is shown as *Pseudocode 4*. The PEDA mainly includes four components, i.e., initialize population, construct the probabilistic model, sample from the probabilistic model, and multiobjective local search. In the initial phase, we use an improved PWQ heuristic to generate the initial population and store the nondominated solutions from the population in the archive set. Then, the probability of jobs in the empty factory, the probability of two jobs in the same factory, the probability of two adjacent jobs is computed by extracting the distribution of solutions in the archive set. Next, the offspring individuals are produced by sampling from the probabilistic model. Finally, the multiobjective local search is applied to the solutions in the offspring population and the archive set. The above-mentioned procedure is repeated until the termination criterion is satisfied. Fig. 5 shows the flowchart of PEDA.

H. Complexity of PEDA

Suppose that there are n jobs, m machines, and F factories in the MDNWFSP-SDST. Because of using the speed-up method to compute makespan and total tardiness, so the computational complexity of computing $C_{\max}$ and TWT are $O(n)$, respectively. In the initial phase, the computational complexity

Pseudocode 4 Framework of PEDA

Input: The parameter PS, α, d

Output: The archive set **AS**

1: Set the archive set **AS** to be empty.
2: Generate the population **POP** by using improved PWQ heuristic, and evaluate each solution in the population.
3: Save the nondominated solutions determined from the initial population in the **AS**.
4: Initialize the probabilistic matrix $\boldsymbol{PE}$, $\boldsymbol{PC}$, $\boldsymbol{PA}$.
5: **while** the termination criterion is not met **do**
6: Construct the probabilistic matrix $\boldsymbol{PE}$, $\boldsymbol{PC}$, $\boldsymbol{PA}$ by extracting the distribution of solutions in **AS**.
7: Update $\boldsymbol{PE}$, $\boldsymbol{PC}$, $\boldsymbol{PA}$
8: Generate the offspring population **POP** with PS solutions by sampling from $\boldsymbol{PE}$, $\boldsymbol{PC}$, $\boldsymbol{PA}$ and evaluate the offspring individuals.
9: Store the nondominated solutions determined from offspring population in **AS**.
10: For each solution in the offspring population, perform multiobjective local search, add the solutions generated by multiobjective local search to **AS**.
11: For each solution that not be performed local search in **AS**, perform multiobjective local search, add the solutions generated by multiobjective local search to **LPOP**.
12: **UPOP** $=$ **POP** $\cup$ **OPOP** $\cup$ **LPOP**
13: Perform *fast nondominated sorting* on **UPOP**.
14: **POP** $=$ **UPOP**[1 : PS], set **OPOP** and **LPOP** to be empty.
15: **end while**

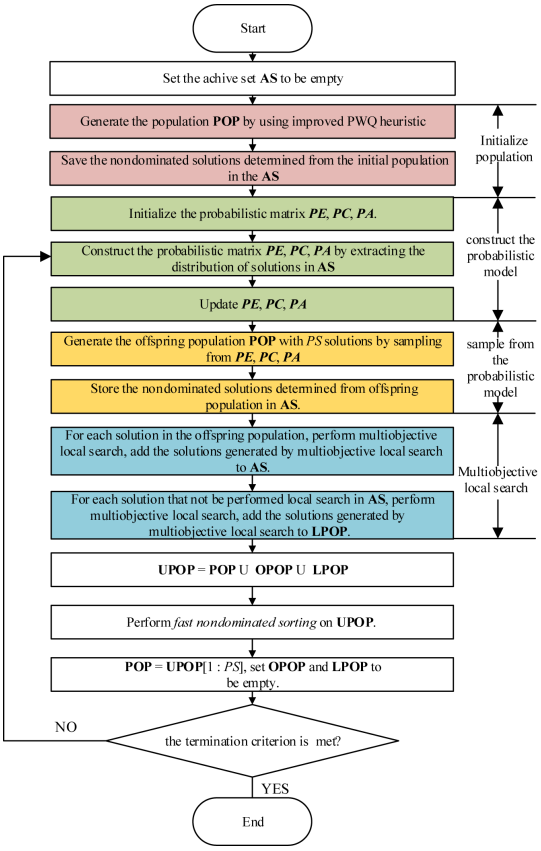


Fig. 5. Flowchart of PEDA.

of computing weight, sorting the jobs, and inserting operation are $O(n)$, $O(nlogn)$, and $O(n^2 \times 2n)$, respectively. Then, the computational complexity of generating p individuals is $O(p(n+nlogn+n^2\times 2n))$. The computational complexity of computing the probability matrix $\boldsymbol{PE}$ of job in empty factory, $\boldsymbol{PC}$ of two jobs in the same factory, and $\boldsymbol{PA}$ of two adjacent jobs are $O(3pn^2)$. The computational complexity of generating p individuals by sampling is $O(2pn)$. The computational complexity of *Pareto Insert Search*, *Pareto Swap Search*, *Pareto Insert Search for Tardiness*, *Pareto Swap Search for Tardiness*, and *Pareto Insert Search for Critical Factory* is $O(n^3)$, $O(n^2)$, $O(n^3)$, and $O(n^3)$. Then, the computational complexity of applying the local search on the archive set and offspring is $O((a+p)O(4n^3+n^2))$, where a is the size of archive set. The computation complexity of fast nondominated sorting is $O(2\beta^2)$, where β is the size of UPOP. In summary, in k iterations, the computation complexity of PEDA is shown as follows:

$$\begin{aligned} &O(k,n,m,F,p) \\ &= O(p(n+n\log n+n^2\times n))+O(k)[O(3pn^2)+O(2pn) \\ &\quad +(a+p)O(n^3+n^3+n^2+n^2+n^3)+O(2\beta^2)] \\ &\approx O(n^3)+O(k)O[pn+pn^2+(a+p)n^3+2\beta^2] \\ &\approx O(k(a+p)n^3+2k\beta^2). \end{aligned} \tag{37}$$

IV. Experimental Comparison

A. Experiment Setup

In order to demonstrate the effectiveness of PEDA, this section carries out an experiment to test the performance of PEDA with testing instances. Since there is no published benchmark instance for the MDNWFSP-SDST, we generate two sets of testing instances (denoted as **SSD50** and **SSD125**) based on the literature [4], [49]. Each set includes several combinations of the number of jobs, machines, and factories. The combinations of $n \times m$ are $\{20, 50, 100\} \times \{5, 10, 20\}$, $200 \times \{10, 20\}$. Each combination consists of 10 different replicates which results in 110 instances in total. We increase these 110 instances with six classes of factories $F = \{2, 3, 4, 5, 6, 7\}$. It leads to 660 instances in total. The processing time, the setup time, the weight, and the due date of each job come from the literature [49]. The setup time of **SSD50** and **SSD125** is uniformly distributed in the range [0–49] and [50–124], respectively. The weight of each job is generated by a uniform $U[1, 10]$ distribution, and a tight due date d_j is adopted for each job j following the expression: $d_j = p_j \times (1 + random \times 3)$ where p_j denotes the total processing time of job j, and $random$ is a random number uniformly in $[0, 1]$. All of the testing instances can be downloaded from the website at http://soa.iti.es. Note that all algorithms are implemented with Java language (JDK 1.6) and performed on a server with two Intel Xeon E5-2650 CPUs and 128G RAM.

TABLE I
PARAMETER SETTING OF COMPARED ALGORITHMS

Algorithms	Parameter setting
CMA	The population size $PS = 20$, the times of performing local search $LS = 10$, the iteration of exchanging individual $LT = 10$.
HIA	The population size $PS = 20$, the ratio of regenerate antibodies $\gamma = 0.1$, the local search method $t_l = 10$, the ratio of cloned antibodies $\alpha = 0.15$.
hMGA	The population size $PS = 50$, the mutation probability $Pm = 0.3$, the crossover probability $Pc = 0.99$,
INSGAII	The population size $PS = 50$, the archive set size $\overline{PS} = 50$, the mutation probability $Pm = 0.5$, the crossover probability $Pc = 0.9$.
PGA_ALS	The population size $PS = 50$, the mutation probability $Pm = 0.1$, the crossover probability $Pc = 1$.
MOEA/D	The population size $PS = 100$, the mutation probability $Pm = 0.1$, the crossover probability $Pc = 1$.
NSGAII	The population size $PS = 100$, the neighborhood size is 20, the mutation probability size $Pm = 0.1$, the crossover probability $Pc = 1$.
PESAII	The population size $PS = 50$, the archive set size $\overline{PS} = 20$, the crossover probability $Pc = 0.9$,the number of divisions in the Grid bisection = 5, the mutation probability$Pm = 0.1$,
SPEAII	The population size $PS = 100$, the archive set size $\overline{PS} = 50$, the mutation probability $Pm = 0.1$, the crossover probability $Pc = 0.9$

TABLE II
ANOVA RESULTS OF I_H

Source	Sum sq.	d. f.	Mean sq.	F-value	p-value
Mean effects					
PS	0.01652	3	0.00551	5.97	0.0005
d	0.26475	3	0.08825	95.74	0
α	0.00407	4	0.00102	1.1	0.3526
Interactions					
$PS * d$	0.00302	9	0.00034	0.36	0.9523
$PS * \alpha$	0.00352	12	0.00067	0.32	0.9864
$d * \alpha$	0.00806	12	0.00092	0.73	0.7247

TABLE III
ANOVA RESULTS OF I_ε^1

Source	Sum sq.	d. f.	Mean sq.	F-value	p-value
Mean effects					
PS	0.0059	3	0.00197	7.17	0.0001
d	0.11824	3	0.03941	143.69	0
α	0.00113	4	0.00028	1.03	0.3895
Interactions					
$PS * d$	0.00145	9	0.00016	0.59	0.8095
$PS * \alpha$	0.00166	12	0.00014	0.5	0.9137
$d * \alpha$	0.00465	12	0.00039	1.41	0.1517

B. Adaption of Existing Compared Algorithms

Because this paper is the first attempt to solve the MDNWFSP-SDST, so there is no specific existing algorithm presented in former studies. We extend nine algorithms to solve the MDNWFSP-SDST, which include NSGAII of Deb *et al.* [48], PESAII of Corne *et al.* [50], SPEAII of Zitzler *et al.* [51], MOEA/D of Zhang *et al.* [52], hMGA of Yandra and Tamura [53], INSGAII of Cai *et al.* [14], PGA_ALS of Pasupathy *et al.* [54], CMA of Deng and Wang [12], and HIA of Xu and Wang [24]. In these algorithms, NSGAII, PESAII, SPEAII, and MOEA/D are proposed for multiobjective function optimization, PGA_ALS and hMGA are presented for addressing the multiobjective permutation flow-shop scheduling problem, INSGAII and CMA are developed for addressing the multiobjective DPFSP, and the HIA is proposed for the DPFSP. The HIA combines the immune search operators and local search. Since the HIA is proposed for the single-objective optimization, we use the fast nondominated sorting and crowding distance to sort the individuals in the parent and offspring population, and the archive set is used to collect the nondominated solutions in the searching process.

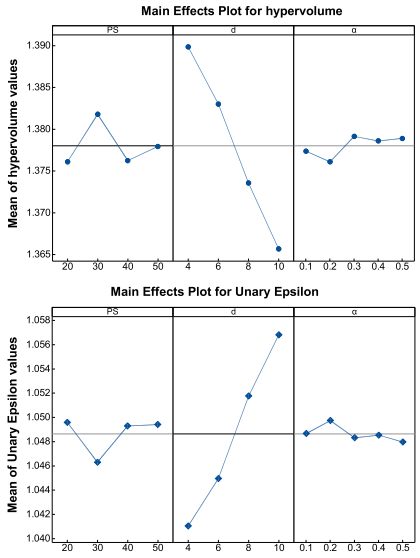


Fig. 6. Main effects plot for hypervolume (I_H) and unary epsilon (I_ε^1) of each parameter.

Note that since hMGA, INSGAII, MOEA, NSGAII, PESAII, PGA_ALS, and SPEAII are not initially used to solve the distributed flow-shop scheduling problem, we adopt the same crossover and mutation operation. The crossover operator proposed by Gao and Chen [19] is used in this section. Suppose there are two parent individuals $\boldsymbol{s1}$ and $\boldsymbol{s2}$. The crossover operator first randomly selects points for each factory of the parent $\boldsymbol{s2}$, which are used to divided job sequence of $\boldsymbol{s1}$. The set of corresponding jobs on right side of $\boldsymbol{s2}$ is removed from $\boldsymbol{s1}$. Then, the remaining jobs in each factory of $\boldsymbol{s1}$ are placed in the order of their appearance and inherited to the child. The right-side jobs of $\boldsymbol{s2}$ are conjoined to the corresponding factories of the child. The mutation operator randomly swaps some pairs of jobs, and the number of pair is $n/4$. It should note that we carefully follow all explanations and details described in the original literature to implement these compared algorithms, their parameters are strictly calibrated by the analysis of variance (ANOVA) method. Due to the limited space, we just list the final parameter setting of each algorithm in Table I. In order to make an effective comparison, the same experimental conditions are employed in testing.

C. Performance Evaluation Indicators

In our experiment, the hypervolume indicator I_H [55] and the unary epsilon indicator I_ε^1 [36] are considered as performance evaluation indicators, which have been widely used to assess the performance of multiobjective algorithms.

The hypervolume indicator I_H was proposed by Zitzler and Thiele [55], which measures the normalized volume of the solution space dominated by the Pareto

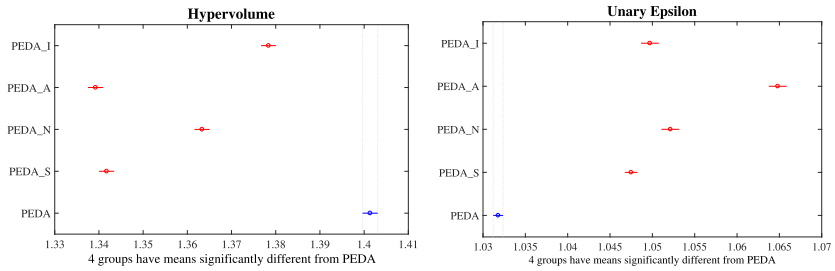


Fig. 7. Mean plot and Tukey's HSD results for hypervolume and unary epsilon of PEDA_I, PEDA_A, PEDA_N, PEDA_S, and PEDA.

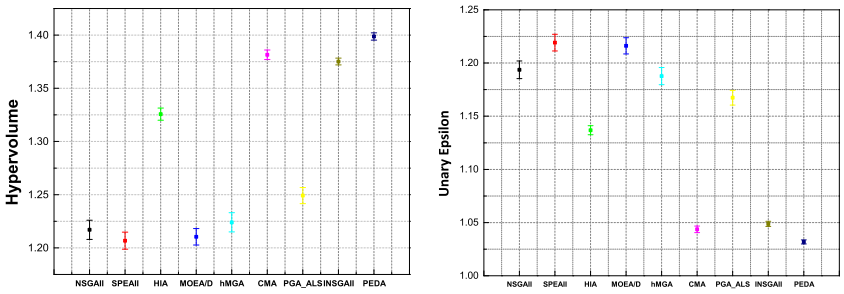


Fig. 8. Mean plot of hypervolume and unary epsilon for compared algorithms.

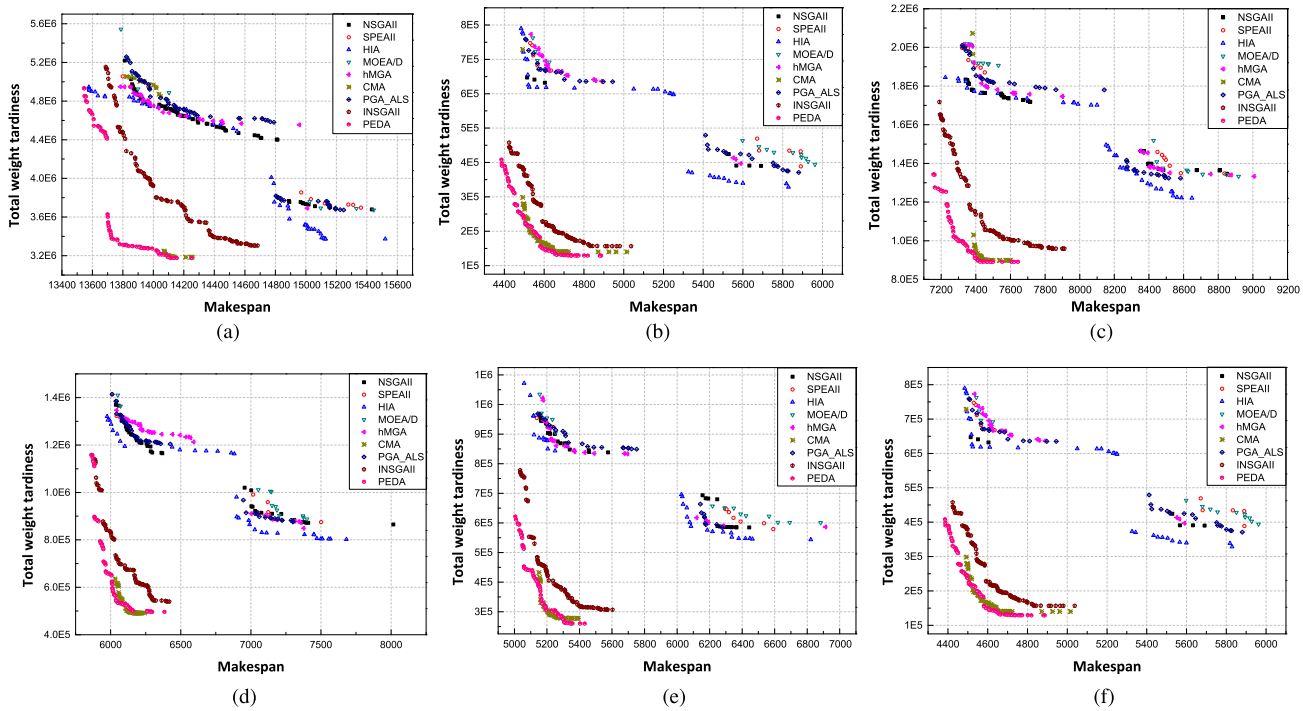


Fig. 9. Pareto fronts obtained by using nine comparative algorithms for the 110th instance from **SSD50**. (a) $f = 2$. (b) $f = 3$. (c) $f = 4$. (d) $f = 5$. (e) $f = 6$. (f) $f = 7$.

approximation generated by an algorithm. As [49], the point (1.2, 1.2) is considered as the reference point. Suppose there is a set of Pareto frontiers $\boldsymbol{PF}$, and $pf \in \boldsymbol{PF}$ is a frontier. $\boldsymbol{s}$ is a solution belonging to $pf \in \boldsymbol{PF}$. The hypervolume indicator for biobjectives is shown as (37), where f_1^i and f_2^i denote the objective values of the solution $\boldsymbol{s}$ belonging to the Pareto frontier pf, and $f_1^{\min}$, $f_1^{\max}$, $f_2^{\min}$, $f_2^{\max}$ denote the best (minimum) and the worst (maximum) value of f_1 and f_2 over all the Pareto frontiers $\boldsymbol{PF}.NSol$ is the number of solutions in the Pareto frontier pf. It should note that when comparing two Pareto frontiers, the higher value of I_H means a better frontier

$$I_H = \sum_{i=1}^{Nsol} \frac{f_1^i - 1.2 \times f_1^{\min}}{1.2 \times \left(f_1^{\max} - f_1^{\min}\right)} + \frac{f_2^i - 1.2 \times f_2^{\min}}{1.2 \times \left(f_2^{\max} - f_2^{\min}\right)}. \quad (38)$$

The unary epsilon indicator I_ε^1 was proposed by Zitzler *et al.* [36], which computes the distance between an

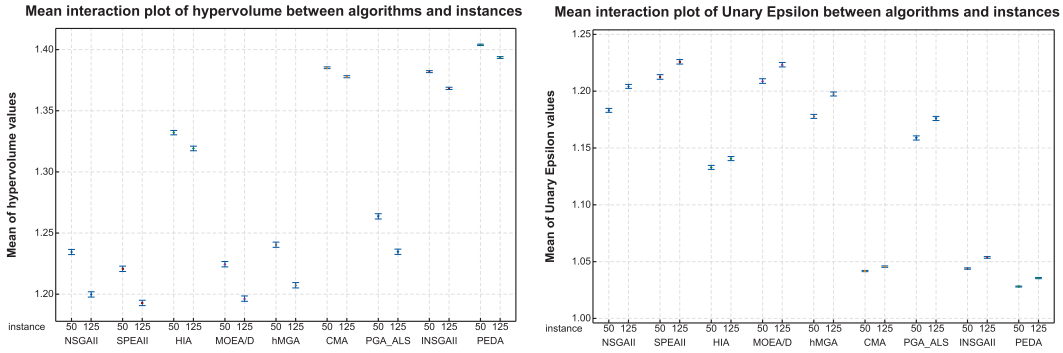


Fig. 10. Mean interaction plot of hypervolume and unary epsilon between algorithms and instances.

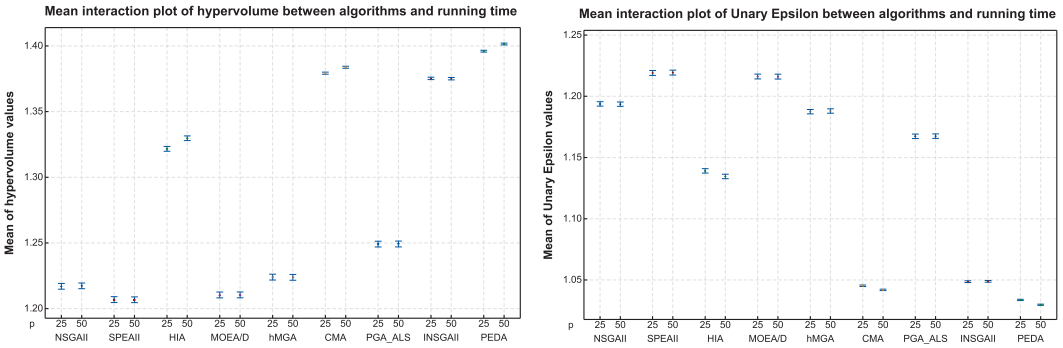


Fig. 11. Mean interaction plot of hypervolume and unary epsilon between algorithms and running time.

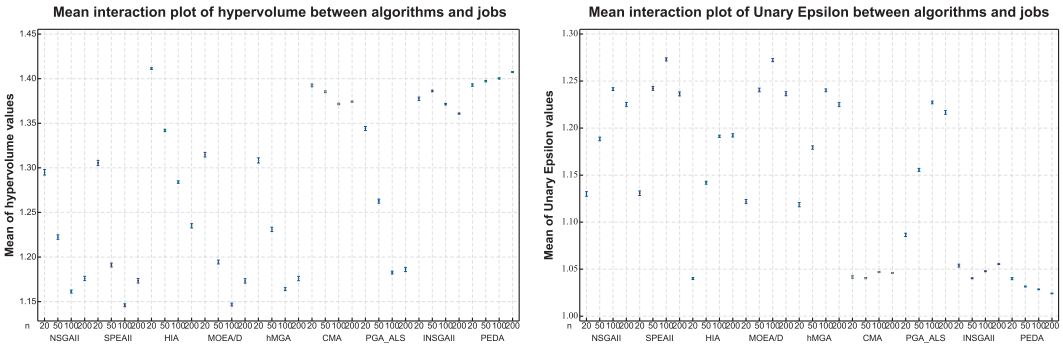


Fig. 12. Mean interaction plot of hypervolume and unary epsilon between algorithms and jobs.

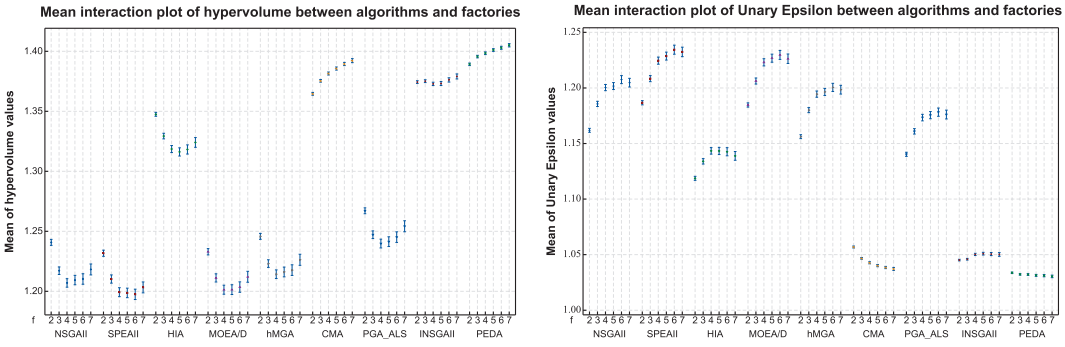


Fig. 13. Mean interaction plot of hypervolume and unary epsilon between algorithms and factories.

obtained Pareto front and the referenced Pareto front. As [49], because the optimal Pareto front for each instance is not known, so we collect the whole of nondominated solutions generated by all considered algorithms as a reference set. Each objective of solutions should be also normalized to one unit by the following equation: $f_j' = (f_j - f_j^{\min})/(f_i^{\max} - f_i^{\min}) + 1$.

TABLE IV
COMPUTATIONAL RESULT OF PEDA_I, PEDA_A, PEDA_N, PEDA_S, AND PEDA

factories	Indicators	Algorithms				
		PEDA_I	PEDA_A	PEDA_N	PEDA_S	PEDA
$f=2$	I_H	1.383	1.351	1.376	1.356	**1.390**
	I_ε^1	1.044	1.050	1.038	1.050	**1.033**
$f=3$	I_H	1.381	1.349	1.370	1.345	**1.398**
	I_ε^1	1.048	1.054	1.044	1.06	**1.031**
$f=4$	I_H	1.375	1.341	1.359	1.336	**1.400**
	I_ε^1	1.053	1.063	1.054	1.068	**1.031**
$f=5$	I_H	1.377	1.34	1.359	1.337	**1.406**
	I_ε^1	1.051	1.066	1.056	1.069	**1.029**
$f=6$	I_H	1.374	1.332	1.356	1.336	**1.406**
	I_ε^1	1.053	1.073	1.060	1.071	**1.028**
$f=7$	I_H	1.379	1.323	1.359	1.340	**1.409**
	I_ε^1	1.050	1.082	1.060	1.071	**1.027**

The values of I_ε^1 are between 1 and 2. The value close to 1 means the considered frontier is close to the reference set, and the value being close to 2 means that the set of solution is distant. Let **B** be the reference set and **A** be an approximation to the Pareto front. I_ε^1 for two objectives is shown as (38), where $\boldsymbol{f}^1 = (f_1^1, f_2^1)$ and $\boldsymbol{f}^2 = (f_1^2, f_2^2)$ represent two objective values belonging to **A** and **B**, respectively, and m denote the number of objectives

$$I_\varepsilon^1(\mathbf{A},\mathbf{B}) = \max_{\boldsymbol{f}^2\in\mathbf{B}} \min_{\boldsymbol{f}^1\in\mathbf{A}} \max_{1\le i\le m} \frac{f_i^1}{f_i^2}. \tag{39}$$

D. Calibration of PEDA

To develop an efficient and effective algorithm, we must find the suitable parameters setting for the PEDA. The PEDA includes three parameters that are the PS the number of selected jobs d, and the learning rate α. This section employs a full factorial design of experiment to calibrate the PEDA. The levels of each parameter are $PS = \{20, 30, 40, 50\}$, $d = \{4, 6, 8, 10\}$, and $\alpha = \{0.1, 0.2, 0.3, 0.4, 0.5\}$. In order to prevent overfitting, we randomly generate 22 other testing instances based on the structure of **SSD50** and **SSD125**, and each combination includes one instance. The number of factories are chosen as 4 and 5. Thus, there are 44 testing instances used to test the parameters of PEDA. Each instance is run with five times and the maximum running time $T_{\max} = n \times m \times f \times 15$ ms. The ANOVA method is used to analyze the computational results, in which the p-value being close to zero demonstrates that there exists a statistical significant difference between each level of factor or interaction. If the p-value is closed to zero, the F-ratio will be used to determine the significance.

Tables II and III show the ANOVA results of I_H and I_ε^1 for the PEDA. From these figures, it can be observed that the PS and the number of selected jobs are significant parameters since the p-values of their I_H and I_ε^1 are less than 0.05. The number of selected jobs d is the largest significant parameter which has the largest F-value. From the main effect plot of d in Fig. 6, the smaller value leads to better results, because the small d may benefit to save local search information. The choice of $d = 4$ results in the best result. The PS is the second significant parameter. It demonstrates that PS has the obvious influence on the performance of the PEDA. From Fig. 6, PS = 30 leads to better solutions than other candidate levels. The small-scaled population may reduce the diversity of population after performing an amount of local search on it, but if there are overmuch individuals in the population, performing many local searches on them may cost more time. The learning rate α has no significant impact on the performance of the PEDA, so the value of α is set as 0.5 according to the main effect plot. According to the above-mentioned investigation, the parameters setting of PEDA is PS = 30, $d = 4$, and $\alpha = 0.5$.

E. Performance Analysis of Each Part of PEDA

This section tests and analyzes the contribution of critical components of PEDA: the improved PWQ heuristic, the combination of local search and PEDA, and the sample method with referenced template. Four variants of PEDA are generated: PEDA_I, PEDA_A, PEDA_N, and PEDA_S. PEDA_I randomly generates the initial population. PEDA_A and PEDA_N are obtained by removing local searching method applied to the achieve set and the offspring individuals, respectively. In PEDA_S, each offspring individual is produced through sampling the probabilistic model directly. First, we determine the first job of each factory by sampling the probability matrix $\boldsymbol{PE}$ of job in empty factory. Then, the jobs in the same factory are decided through sampling the probability matrix $\boldsymbol{PC}$ of job in the same factory. Finally, we get the processing order via sampling the probability matrix $\boldsymbol{PA}$ of adjacent jobs. It should note that the roulette wheel selection is used as the sampling method. All considered algorithms adopt the same termination criterion, i.e., the maximum running time $T_{\max} = n \times m \times f \times 25$ ms. The **SSD50** is employed to verify the performance of each considered algorithm. Table IV lists the computation result of PEDA_I, PEDA_A, PEDA_N, PEDA_S, and PEDA. Fig. 7 shows the mean plot and Tukey's HSD results of I_H and I_ε^1 for all compared algorithms.

As seen in Table IV, the PEDA achieves larger I_H and smaller I_ε^1 values than PEDA_I, PEDA_A, PEDA_N, and PEDA_S. From Fig. 7, the mean values of I_H and I_ε^1 are close to 1.44 and 1, respectively, and there is no overlapping interval between PEDA and PEDA_I, PEDA_A, PEDA_N, and PEDA_S. It shows that the contribution of each component to the PEDA. The PEDA_A obtains worse results among considered variants of PEDA, it implies that the local search for achieve set plays a key contribution. The results of PEDA_S are worse than PEDA, it demonstrates that the high effectiveness and efficiency of our proposed sample method. Based on the above-mentioned results, we can conclude that the PEDA can make a good balance between each component.

F. Computational Result

To demonstrate the effective performance of PEDA for addressing the MDNWFSP-SDST, this section carries out the computational results in detail. The PEDA is compared to other nine algorithms in Section IV-B, i.e., NSGAII, PESAII, SPEAII, MOEA/D, hMGA, INSGAII, PGA_ALS, CMA, and

TABLE V
STATISTICAL RESULTS OF HYPERVOLUME AND UNARY EPSILON FOR COMPARED ALGORITHMS

Algorithms	f=2								f=3							
	SSD50				**SSD125**				**SSD50**				**SSD125**			
	ρ = 25		ρ = 50		ρ = 25		ρ = 50		ρ = 25		ρ = 50		ρ = 25		ρ = 50	
	I_ε^1	I_H	I_ε^1	I_H	I_ε^1	I_H	I_ε^1	I_H	I_ε^1	I_H	I_ε^1	I_H	I_ε^1	I_H	I_ε^1	I_H
NSGAII	1.153	1.253	1.153	1.253	1.171	1.228	1.171	1.228	1.175	1.233	1.177	1.234	1.195	1.200	1.195	1.202
SPEAII	1.180	1.242	1.179	1.243	1.194	1.221	1.194	1.220	1.201	1.225	1.200	1.226	1.216	1.195	1.216	1.195
PESAII	1.530	0.498	1.530	0.501	1.548	0.468	1.546	0.472	1.498	0.554	1.500	0.563	1.525	0.518	1.523	0.523
HIA	1.117	1.348	1.111	1.357	1.126	1.338	1.121	1.347	1.132	1.332	1.126	1.341	1.141	1.318	1.136	1.327
MOEA/D	1.177	1.245	1.177	1.245	1.193	1.221	1.193	1.221	1.197	1.226	1.199	1.226	1.215	1.196	1.213	1.197
hMGA	1.148	1.258	1.148	1.259	1.165	1.233	1.164	1.234	1.170	1.239	1.171	1.239	1.190	1.207	1.190	1.207
CMA	1.056	1.368	1.053	1.371	1.061	1.357	1.057	1.362	1.047	1.377	1.043	1.381	1.050	1.369	1.046	1.375
PGA_ALS	1.133	1.276	1.132	1.279	1.148	1.256	1.147	1.257	1.152	1.261	1.153	1.261	1.170	1.233	1.169	1.234
INSGAII	1.041	1.380	1.041	1.379	1.049	1.369	1.048	1.370	1.037	1.385	1.044	1.379	1.052	1.367	1.050	1.369
PEDA	**1.033**	**1.390**	**1.028**	**1.398**	**1.040**	**1.380**	**1.034**	**1.389**	**1.031**	**1.398**	**1.026**	**1.405**	**1.039**	**1.386**	**1.033**	**1.394**
	f=4								f=5							
NSGAII	1.194	1.219	1.193	1.218	1.209	1.194	1.205	1.196	1.191	1.230	1.192	1.229	1.213	1.188	1.211	1.190
SPEAII	1.223	1.208	1.222	1.209	1.227	1.190	1.226	1.190	1.223	1.215	1.222	1.215	1.234	1.182	1.235	1.182
PESAII	1.493	0.585	1.494	0.573	1.501	0.561	1.503	0.556	1.465	0.619	1.468	0.616	1.494	0.581	1.496	0.588
HIA	1.143	1.320	1.139	1.328	1.148	1.309	1.143	1.317	1.142	1.320	1.138	1.328	1.149	1.304	1.145	1.312
MOEA/D	1.219	1.211	1.221	1.210	1.227	1.192	1.225	1.192	1.221	1.218	1.221	1.219	1.233	1.184	1.233	1.184
hMGA	1.188	1.227	1.190	1.224	1.200	1.203	1.200	1.203	1.187	1.236	1.188	1.235	1.205	1.197	1.206	1.197
CMA	1.044	1.380	1.040	1.386	1.045	1.377	1.041	1.383	1.038	1.389	1.037	1.392	1.045	1.378	1.040	1.384
PGA_ALS	1.169	1.249	1.168	1.251	1.177	1.230	1.178	1.229	1.168	1.259	1.166	1.259	1.184	1.224	1.184	1.224
INSGAII	1.046	1.378	1.047	1.378	1.053	1.368	1.055	1.367	1.046	1.382	1.045	1.383	1.056	1.366	1.057	1.362
PEDA	**1.031**	**1.400**	**1.027**	**1.406**	**1.037**	**1.392**	**1.033**	**1.396**	**1.029**	**1.406**	**1.026**	**1.409**	**1.037**	**1.392**	**1.033**	**1.398**
	f=6								f=7							
NSGAII	1.193	1.233	1.193	1.232	1.222	1.188	1.222	1.189	1.190	1.240	1.192	1.239	1.219	1.197	1.218	1.197
SPEAII	1.225	1.214	1.226	1.213	1.244	1.181	1.244	1.182	1.224	1.219	1.226	1.218	1.239	1.188	1.241	1.188
PESAII	1.455	0.642	1.455	0.643	1.482	0.611	1.486	0.606	1.444	0.666	1.448	0.667	1.464	0.636	1.461	0.645
HIA	1.139	1.323	1.137	1.328	1.150	1.306	1.144	1.316	1.137	1.328	1.133	1.334	1.145	1.313	1.140	1.322
MOEA/D	1.219	1.220	1.220	1.220	1.240	1.186	1.239	1.188	1.217	1.227	1.219	1.226	1.235	1.197	1.234	1.197
hMGA	1.187	1.237	1.189	1.236	1.215	1.197	1.212	1.200	1.183	1.248	1.187	1.247	1.212	1.205	1.212	1.205
CMA	1.037	1.391	1.035	1.394	1.042	1.385	1.040	1.388	1.035	1.396	1.034	1.397	1.043	1.384	1.037	1.392
PGA_ALS	1.166	1.263	1.168	1.262	1.189	1.228	1.190	1.227	1.165	1.272	1.165	1.271	1.187	1.238	1.188	1.236
INSGAII	1.045	1.384	1.046	1.383	1.057	1.367	1.054	1.371	1.043	1.388	1.046	1.386	1.058	1.369	1.055	1.373
PEDA	**1.028**	**1.406**	**1.026**	**1.410**	**1.037**	**1.396**	**1.034**	**1.400**	**1.027**	**1.409**	**1.025**	**1.412**	**1.037**	**1.398**	**1.033**	**1.402**

HIA. We run these 10 algorithms based on the 660 instances of **SSD50** and **SSD125**, respectively. For making a fair comparison, each comparative algorithm has the same termination criterion, i.e., the maximum stopping time. Two kinds of maximum stopping time are considered, which are $T_{\max} = n \times m \times f \times \rho$ ms, $\rho = 25, 50$. Table V summarizes the statistical results for I_H and I_ε^1 of each comparative algorithm grouped by different factories. As seen in Table V, in the average values of the measure indicators, i.e., hypervolume and unary epsilon, the PEDA turns out to be the best performance, the second in ranking is CMA and INSGAII whose results are close to each other, and the worst is PESAII. Fig. 8 shows the mean plot with 95% confidence interval of results of I_H and I_ε^1 for all compared algorithms. Since the results of PESAII are much worse than other algorithms, in order to show the comparison results clearly we do not display its results in statistical figures. As seen in Fig. 8, the union I_H value of PEDA is the nearest to 1.44 as well as its I_ε^1 is the nearest to 1. Moreover, there is no overlapping interval between PEDA and other algorithms, it demonstrates that PEDA is better than other compared algorithms significantly from statistical perspective. In addition, the algorithms with Pareto-based local search methods, i.e., PEDA, CMA, INSGAII, HIA, and PGA_ALS are obviously superior to that without any local search, which verifies the effectiveness of Pareto-based local search. Fig. 9 shows the Pareto fronts obtained by using nine comparative algorithms for the 110th instance from **SSD50**. It can be seen from these figures that the Pareto fronts yielded by PEDA and CMA are much closer to the bottom-left corner of coordinate axis than other algorithms, it demonstrates their strong exploration and exploitation ability. However, although the CMA obtains some same or better solutions compared to PEDA, the distribution of solutions of CMA is poor because its solutions only exist around the original point, and it has lesser solution that lies around the coordinate axes. Therefore, the Pareto front of CMA is worse than that of PEDA.

To evaluate whether the differences of computational results are significant from the statistical perspective, the ANOVA method is used to analyze the interaction between compared algorithms and the scale of jobs, the numbers of factories, the maximum stopping running time, and the testing instances. Figs. 10–13 display mean interaction plots of hypervolume and unary epsilon between algorithms and the above-mentioned factors. From these figures, it can be concluded as follows.

1) All algorithms obtain better solutions for **SSD50** than **SSD125**, it demonstrates the influence of the setup time, i.e., larger setup time may increase the difficulty of MDNWFSP-SDST.
2) PEDA, CMA, and HIA obtain better solutions with additional running time, while I_H and I_ε^1 of other algorithms are not changed significantly.
3) The I_H values increase and the I_ε^1 values of PEDA decrease with the higher number of jobs, whereas other algorithms obtain the opposite results. Moreover, the results of PEDA are better than other algorithms in most of the scales of jobs.
4) The I_H values increase and the I_ε^1 values of PEDA, CMA, and INSGAII reduce with higher number of factories, whereas other algorithms obtain the better results in the small-scale factories, i.e., $f = 2$.

It demonstrates the strong searching ability of PEDA, CMA, and INSGAII for large-scale factories. Furthermore, for all factories, the I_H values of PEDA locate above-mentioned other algorithms and the I_ε^1 values lie under other algorithms, it demonstrates that the results of PEDA are better than other algorithms. According to above-mentioned experimental results and analysis, it can draw a conclusion that the PEDA can solve the MDNWFSP-SDST effectively and efficiently.

V. Conclusion

This paper studies a multiobjective distributed MDNWFSP-SDST with two performance criteria, i.e., makespan and TWT. There exist several same factories in the MDNWFSP-SDST and each factory includes a flow-shop scheduling problem with no-wait constraint. The setup process is considered between the completion of one job and the start of next job on each machine. The task of MDNWFSP-SDST is to reasonably assign jobs to factories and find the processing sequence of jobs in each factory to optimize multiple scheduling objectives. For the MDNWFSP-SDST, this paper develops a PEDA. The PWQ heuristic is extended to the distributed environment to generate the initial individuals. Three probabilistic models, i.e., the probability of the jobs in empty factory, the two jobs in same factory, and the adjacent jobs are established. A sampling method with referenced template is presented to generate offspring individuals. To optimize the local exploitation ability, the multiobjective local search is performed on the solutions in the offspring population and the archive set.

Comprehensive statistical analysis has been used to investigate the performance of proposed method. From the results, PEDA performs better than other compared algorithms. It demonstrates the high effectiveness and efficiency of PEDA. Based on the applicability of studied problem and the proposed methods, our work displays a promising prospect in addressing more complex distributed scheduling problems.

For future study, it would be interesting to consider distributed distinct factories environment that is much closer to practical production. The development of effective multiobjective optimization algorithms is another future research direction. Furthermore, the problem properties of MDNWFSP-SDST should be studied further.

References

[1] A. Allahverdi, "A survey of scheduling problems with no-wait in process," *Eur. J. Oper. Res.*, vol. 255, no. 3, pp. 665–686, 2016.
[2] F. T. Hecker, W. B. Hussein, O. Paquet-Durand, M. A. Hussein, and T. Becker, "A case study on using evolutionary algorithms to optimize bakery production planning," *Expert Syst. Appl.*, vol. 40, no. 17, pp. 6837–6847, 2013.
[3] D. Gong, Y. Han, and J. Sun, "A novel hybrid multi-objective artificial bee colony algorithm for blocking lot-streaming flow shop scheduling problems," *Knowl.-Based Syst.*, vol. 148, pp. 115–130, May 2018.
[4] B. Naderi and R. Ruiz, "The distributed permutation flowshop scheduling problem," *Comput. Oper. Res.*, vol. 37, no. 4, pp. 754–768, 2010.
[5] S.-W. Lin and K.-C. Ying, "Minimizing makespan for solving the distributed no-wait flowshop scheduling problem," *Comput. Ind. Eng.*, vol. 99, pp. 202–209, Sep. 2016.
[6] W. Shao, D. Pi, and Z. Shao, "Optimization of makespan for the distributed no-wait flow shop scheduling problem with iterated greedy algorithms," *Knowl.-Based Syst.*, vol. 137, pp. 163–181, Dec. 2017.
[7] S. Hatami, R. Ruiz, and C. Andrés-Romano, "The distributed assembly permutation flowshop scheduling problem," *Int. J. Prod. Res.*, vol. 51, no. 17, pp. 5292–5308, 2013.
[8] S. Hatami, R. Ruiz, and C. Andrés-Romano, "Heuristics and metaheuristics for the distributed assembly permutation flowshop scheduling problem with sequence dependent setup times," *Int. J. Prod. Econ.*, vol. 169, no. 11, pp. 76–88, Nov. 2015.
[9] J. Lin, Z.-J. Wang, and X. Li, "A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem," *Swarm Evol. Comput.*, vol. 36, pp. 124–135, Oct. 2017.
[10] I. Ribas, R. Companys, and X. Tort-Martorell, "Efficient heuristics for the parallel blocking flow shop scheduling problem," *Expert Syst. Appl.*, vol. 74, pp. 41–54, May 2017.
[11] K.-C. Ying and S.-W. Lin, "Minimizing makespan in distributed blocking flowshops using hybrid iterated greedy algorithms," *IEEE Access*, vol. 5, no. 99, pp. 15694–15705, Aug. 2017.
[12] J. Deng and L. Wang, "A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem," *Swarm Evol. Comput.*, vol. 32, pp. 121–131, 2017, doi: 10.1016/j.swevo.2016.06.002.
[13] J. Deng, L. Wang, C. Wu, J. Wang, and X. Zheng, "A competitive memetic algorithm for carbon-efficient scheduling of distributed flow-shop," in *Proc. Int. Conf. Intell. Comput.*, 2016, pp. 476–488.
[14] S. Cai, K. Yang, and K. Liu, "Multi-objective optimization of the distributed permutation flow shop scheduling problem with transportation and eligibility constraints," *J. Oper. Res. Soc. China*, vol. 6, no. 3, pp. 391–416, Sep. 2018.
[15] A. P. Rifai, H.-T. Nguyen, and S. Z. M. Dawal, "Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling," *Appl. Soft Comput.*, vol. 40, pp. 42–57, Mar. 2016.
[16] J. Gao and R. Chen, "An NEH-based heuristic algorithm for distributed permutation flowshop scheduling problems," *Sci. Res. Essays*, vol. 6, no. 14, pp. 3094–3100, 2011.
[17] J. Gao, R. Chen, W. Deng, and Y. Liu, "Solving multi-factory flowshop problems with a novel variable neighbourhood descent algorithm," *J. Comput. Inf. Syst.*, vol. 8, no. 5, pp. 2025–2032, 2012.
[18] H. Liu and L. Gao, "A discrete electromagnetism-like mechanism algorithm for solving distributed permutation flowshop scheduling problem," in *Proc. IEEE Int. Conf. Manuf. Automat.*, Dec. 2010, pp. 156–163.
[19] J. Gao and R. Chen, "A hybrid genetic algorithm for the distributed permutation flowshop scheduling problem," *Int. J. Comput. Intell. Syst.*, vol. 4, no. 4, pp. 497–508, 2011.

[20] J. Gao, R. Chen, and Y. Liu, "A knowledge-based genetic algorithm for permutation flowshop scheduling problems with multiple factories," *Int. J. Adv. Comput. Technol.*, vol. 4, no. 7, pp. 121–129, 2012.

[21] B. Naderi and R. Ruiz, "A scatter search algorithm for the distributed permutation flowshop scheduling problem," *Eur. J. Oper. Res.*, vol. 239, no. 2, pp. 323–334, 2014.

[22] J. Gao, R. Chen, and W. Deng, "An efficient tabu search algorithm for the distributed permutation flowshop scheduling problem," *Int. J. Prod. Res.*, vol. 51, no. 3, pp. 641–651, 2013.

[23] S.-W. Lin, K.-C. Ying, and C.-Y. Huang, "Minimising makespan in distributed permutation flowshops using a modified iterated greedy algorithm," *Int. J. Prod. Res.*, vol. 51, no. 16, pp. 5029–5038, 2013.

[24] Y. Xu, L. Wang, S. Wang, and M. Liu, "An effective hybrid immune algorithm for solving the distributed permutation flow-shop scheduling problem," *Eng. Optim.*, vol. 46, no. 9, pp. 1269–1283, 2014.

[25] S.-Y. Wang, L. Wang, M. Liu, and Y. Xu, "An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem," *Int. J. Prod. Econ.*, vol. 145, no. 1, pp. 387–396, 2013.

[26] H. Bargaoui, O. B. Driss, and K. Ghédira, "A novel chemical reaction optimization for the distributed permutation flowshop scheduling problem with makespan criterion," *Comput. Ind. Eng.*, vol. 111, pp. 239–250, Sep. 2017.

[27] P. Larraanaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, vol. 2. Boston, MA, USA: Springer, 2001.

[28] B. Jarboui, M. Eddaly, and P. Siarry, "An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems," *Comput. Oper. Res.*, vol. 36, no. 9, pp. 2638–2646, 2009.

[29] Q.-K. Pan and R. Ruiz, "An estimation of distribution algorithm for lot-streaming flow shop problems with setup times," *Omega*, vol. 40, no. 2, pp. 166–180, 2012.

[30] K. Wang, Y. Huang, and H. Qin, "A fuzzy logic-based hybrid estimation of distribution algorithm for distributed permutation flowshop scheduling problems under machine breakdown," *J. Oper. Res. Soc.*, vol. 67, no. 1, pp. 68–82, 2016.

[31] L. Wang, S. Wang, Y. Xu, G. Zhou, and M. Liu, "A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem," *Comput. Ind. Eng.*, vol. 62, no. 4, pp. 917–926, 2012.

[32] A. Tiwari, P.-C. Chang, M. K. Tiwari, and N. J. Kollanoor, "A Pareto block-based estimation and distribution algorithm for multi-objective permutation flow shop scheduling problem," *Int. J. Prod. Res.*, vol. 53, no. 3, pp. 793–834, 2015.

[33] L. Wang, S. Wang, and M. Liu, "A Pareto-based estimation of distribution algorithm for the multi-objective flexible job-shop scheduling problem," *Int. J. Prod. Res.*, vol. 51, no. 12, pp. 3574–3592, 2013.

[34] X. Hao, M. Gen, L. Lin, and G. A. Suer, "Effective multiobjective EDA for bi-criteria stochastic job-shop scheduling problem," *J. Intell. Manuf.*, vol. 28, no. 3, pp. 833–845, 2017.

[35] L. Wang, C. Fang, C.-D. Mu, and M. Liu, "A Pareto-archived estimation-of-distribution algorithm for multiobjective resource-constrained project scheduling problem," *IEEE Trans. Eng. Manage.*, vol. 60, no. 3, pp. 617–626, Aug. 2013.

[36] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 117–132, Apr. 2003.

[37] J. N. D. Gupta, "Flowshop schedules with sequence dependent setup times," *J. Oper. Res. Soc. Jpn.*, vol. 29, no. 3, pp. 206–219, 1986.

[38] B. Qian, L. Wang, R. Hu, D. X. Huang, and X. Wang, "A DE-based approach to no-wait flow-shop scheduling," *Comput. Ind. Eng.*, vol. 57, no. 3, pp. 787–805, 2009.

[39] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," *Swarm Evol. Comput.*, vol. 1, no. 3, pp. 111–128, 2011.

[40] Q.-K. Pan, L. Wang, and B. Qian, "A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems," *Comput. Oper. Res.*, vol. 36, no. 8, pp. 2498–2511, 2009.

[41] E. Vallada, R. Ruiz, and G. Minella, "Minimising total tardiness in the m-machine flowshop problem: A review and evaluation of heuristics and metaheuristics," *Comput. Oper. Res.*, vol. 35, no. 4, pp. 1350–1373, 2008.

[42] M. Nawaz, E. E. Enscore, Jr., and I. Ham, "A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91–95, 1983.

[43] A. Salhi, J. A. V. Rodríguez, and Q. Zhang, "An estimation of distribution algorithm with guided mutation for a complex flow shop scheduling problem," in *Proc. 9th Annu. Conf. Genetic Evol. Comput.*, 2007, pp. 570–576.

[44] S.-H. Chen, P.-C. Chang, T. C. E. Cheng, and Q. Zhang, "A self-guided genetic algorithm for permutation flowshop scheduling problems," *Comput. Oper. Res.*, vol. 39, no. 7, pp. 1450–1457, 2012.

[45] S.-H. Chen and M.-C. Chen, "Addressing the advantages of using ensemble probabilistic models in Estimation of Distribution Algorithms for scheduling problems," *Int. J. Prod. Econ.*, vol. 141, no. 1, pp. 24–33, 2013.

[46] W. Shao, D. Pi, and Z. Shao, "A hybrid discrete optimization algorithm based on teaching–probabilistic learning mechanism for no-wait flow shop scheduling," *Knowl.-Based Syst.*, vol. 107, pp. 219–234, Sep. 2016.

[47] Q. Zhang, J. Sun, and E. Tsang, "An evolutionary algorithm with guided mutation for the maximum clique problem," *IEEE Trans. Evol. Comput.*, vol. 9, no. 2, pp. 192–200, Apr. 2005.

[48] X. Wang and L. Tang, "A machine-learning based memetic algorithm for the multi-objective permutation flowshop scheduling problem," *Comput. Oper. Res.*, vol. 79, pp. 60–77, Mar. 2017.

[49] M. Ciavotta, G. Minella, and R. Ruiz, "Multi-objective sequence dependent setup times permutation flowshop: A new algorithm and a comprehensive study," *Eur. J. Oper. Res.*, vol. 227, no. 2, pp. 301–312, 2013.

[50] D. W. Corne, N. R. Jerram, J. D. Knowles, M. J. Oates, and J. Martin, "PESA-II: Region-based selection in evolutionary multiobjective optimization," in *Proc. 3rd Annu. Conf. Genet. Evol. Comput. (GECCO)*. San Francisco, CA, USA: Morgan Kaufmann, Jul. 2001, pp. 283–290.

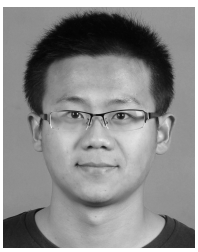
[51] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the performance of the strength Pareto evolutionary algorithm," Comput. Eng. Commun. Netw. Lab, Swiss Federal Inst. Technol., Zurich, Switzerland, Tech. Rep. 103, May 2001.

[52] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[53] Y. Arkeman and H. Tamura, "A new multiobjective genetic algorithm with heterogeneous population for solving flowshop scheduling problems," *Int. J. Comput. Integr. Manuf.*, vol. 20, no. 5, pp. 465–477, 2007.

[54] T. Pasupathy, C. Rajendran, and R. K. Suresh, "A multi-objective genetic algorithm for scheduling in flow shops to minimize the makespan and total flow time of jobs," *Int. J. Adv. Manuf. Technol.*, vol. 27, nos. 7–8, pp. 804–815, 2006.

[55] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.

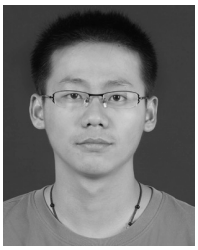


Weishi Shao received the B.Mgt. degree in information management and information system from Anhui Polytechnic University, Wuhu, China, in 2012 and the M.Eng. degree in computer software and theory from Lanzhou University, Lanzhou, China, in 2015. He is currently pursuing the Ph.D. degree with the Nanjing University of Aeronautics and Astronautics, Nanjing, China.

His current research interests include manufacturing scheduling, intelligent optimization methods, and data mining.

Dechang Pi was born in 1971. He received the B.Eng. and M.Eng. degrees from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 1994 and 1997, respectively, and the Ph.D. degree in mechatronic engineering from the Nanjing University of Aeronautics and Astronautics in 2002.

He is currently a Professor and Ph.D. Supervisor with the Nanjing University of Aeronautics and Astronautics. He has authored or co-authored some academic papers which have been published in international journal, such as the IEEE SENSORS JOURNAL, the IEEE SYSTEMS JOURNAL, *Knowledge-Based Systems*, *Expert Systems with Applications*, the *Journal of Systems Engineering and Electronics*, *Applied Soft Computing*, *Engineering Optimization*, *Computers and Industrial Engineering*, and the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS. His research interests are data mining, big data analysis in manufacturing, and intelligent optimization methods.



Zhongshi Shao received the B.Eng. degree in computer science and technology and the M.Eng. degree in software engineering from the Lanzhou University of Technology University, Lanzhou, China, in 2012 and 2015, respectively. He is currently pursuing the Ph.D. degree with the Nanjing University of Aeronautics and Astronautics, Nanjing, China.

His current research interests include manufacturing scheduling, intelligent optimization methods, and data mining.