Expert Systems with Applications 160 (2020) 113678

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

An ensemble discrete differential evolution for the distributed blocking flowshop scheduling with minimizing makespan criterion

Fuqing Zhao [a,*], Lexi Zhao [a], Ling Wang [b], Houbin Song [a]

[a] School of Computer and Communication Technology, Lanzhou University of Technology, Lanzhou 730050, China
[b] Department of Automation, Tsinghua University, Beijing 10084, China

ARTICLE INFO

Article history:
Received 15 November 2019
Revised 17 June 2020
Accepted 17 June 2020
Available online 03 July 2020

Keywords:
Distributed blocking flowshop
Discrete differential evolution
Heuristics method
Front delay
Elitist retain strategy

ABSTRACT

The distributed blocking flowshop scheduling problem (DBFSP) plays an essential role in the manufacturing industry and has been proven to be as a strongly NP-hard problem. In this paper, an ensemble discrete differential evolution (EDE) algorithm is proposed to solve the blocking flowshop scheduling problem with the minimization of the makespan in the distributed manufacturing environment. In the EDE algorithm, the candidates are represented as discrete job permutations. Two heuristics method and one random strategy are integrated to provide a set of desirable initial solution for the distributed environment. The front delay, blocking time and idle time are considered in these heuristics methods. The mutation, crossover and selection operators are redesigned to assist the EDE algorithm to execute in the discrete domain. Meanwhile, an elitist retain strategy is introduced into the framework of EDE algorithm to balance the exploitation and exploration ability of the EDE algorithm. The parameters of the EDE algorithm are calibrated by the design of experiments (DOE) method. The computational results and comparisons demonstrated the efficiency and effectiveness of the EDE algorithm for the distributed blocking flowshop scheduling problem.

1. Introduction

Production scheduling solves the problem of allocating resources (usually machines) to a task or job at a given time to optimize the goal (Zhang, Xing, & Cao, 2018). With the rapid development of the manufacturing technology and the increasing popularity of globalized production, the traditional centralized production planning and scheduling are no longer able to respond the growing market demand (Zhang et al., 2018). Since the market disperses throughout the world, the production model of various companies has changed from single factory to a multi-factory (Li, Yang, Ruiz, Chen, & Sui, 2018). Therefore, the scheduling problem in a distributed production environment has become one of the a hot research issues (Ruiz, Pan, & Naderi, 2019). In the distributed production environment, each factory is considered a separate entity, and all factories work in parallel. Each factory has the same or different configurations and efficiencies and is subject to the same or different constraints. The goal is to assign jobs to the appropriate factory for processing. The layout of the distributed production environment is shown in Fig. 1.

The scheduling in the distributed production environment is classified as distributed scheduling problem (DFSP) (Behnamian & Ghomi, 2016; Naderi & Ruiz, 2010). The parallel flowshop scheduling problem is a special instance of DFSP (He, Kusiak, & Artiba, 1996; Ribas, Companys, & Tort-Martorell, 2019). The scheduling of jobs in a factory with several identical parallel production lines is considered in parallel flowshop scheduling problem. If one production line is existed in the production, the DFSP is the parallel flowshop scheduling problem (Ribas et al., 2019). Compared to the single production center, the distributed production environment has lower production costs and risks, and the quality of product was improved (Naderi & Ruiz, 2014).

In the different types of scheduling problems, the DFSP has attracted attention from researchers and practitioners in the scheduling domain (Maccarthy & Liu, 1993). The distributed permutation flow-shop scheduling problem (DPFSP) was first presented by Naderi and Ruiz (2010), which extended the tradition permutation flowshop scheduling problem (PFSP) to the distributed production environment. Following the work of DPFSP, several researchers proposed various high-performing algorithms for solving the DPFSP. A bounded-search iterated greedy (BIG) algorithm was proposed by Fernandez and Framinan (2015). A new heuristic, iterated greedy (IG) algorithm, was introduced to improve the performance of algorithm. Three different constructive heuristics (DLR,

* Corresponding author.
E-mail address: wangling@tsinghua.edu.cn (L. Wang).

https://doi.org/10.1016/j.eswa.2020.113678
0957-4174/

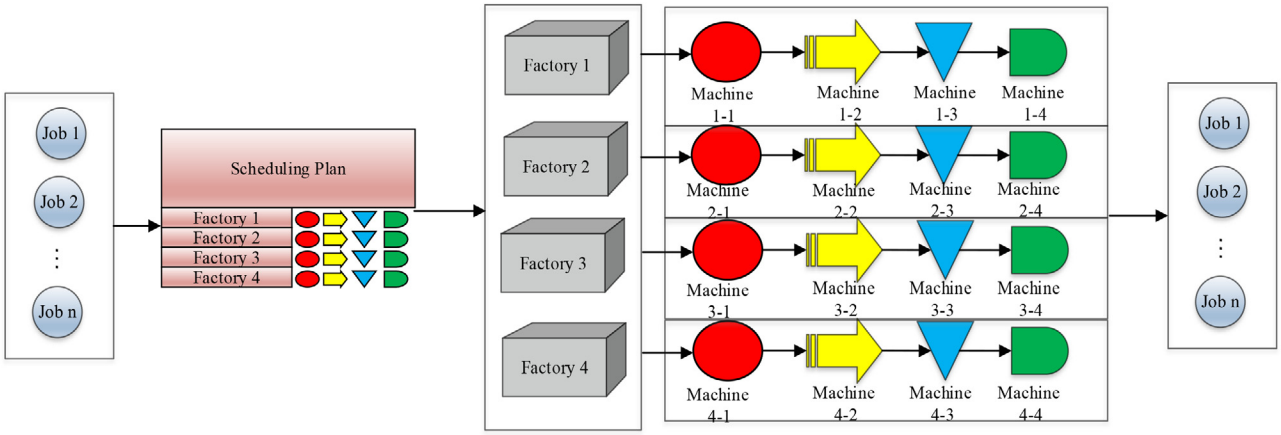


Fig. 1. The environment of the distributed production.

DNEH and DLR-DNEH(x)) and four metaheuristics (DABC, SS, ILS and IG) were proposed by Pan, Gao, Wang, et al. (2019) for solving the DPFSP. Ruiz et al. (2019) proposed an enhanced IG algorithm called IG2S. In IG2S, the initialization, construction and destruction procedures are the intensification mechanisms to improve the local search ability of algorithm.

In addition, other types of DFSP including the distributed no-wait flowshop scheduling problem (DNWFSP) (Lin & Ying, 2016), the distributed no-idle permutation flow-shop scheduling problem (DNIPFSP) (Ying, Lin, Cheng, & He, 2017), the distributed two-stage assembly flow-shop scheduling problem (DTSAFSP) (Zhang & Xing, 2018), and the energy-efficient scheduling of the distributed permutation flow-shop problem (EEDPFSP) (Wang & Wang, 2018), have been investigated in recent years.

If no buffers between two adjacent machines, the FSP are evolved into blocking flowshop scheduling problems (BFSP). In the BFSP, jobs completed on the previous machine are blocked only on the current machine until the next machine is idle. The BFSP is an common scheduling scenarios in modern manufacturing and production systems, e.g., the iron and steel industry (Gong, Tang, & Duin, 2010), manufacturing systems (Zhao, Qin, Yang, et al., 2019), chemical industry (Merchan & Maravelias, 2016), manufacture of metallic parts (Shao, Pi, & Shao, 2019), assembly lines (Zhao, Xue, et al., 2018), and robotic cell (Ribas, Companys, & Tort-Martorell, 2015). BFSP with more than two machines has been proved as a NP-hard problem (Hall & Sriskandarajah, 1996).

Due to the complex nature of the traditional BFSP, exact methods are used only to solve small scale instances. For large-scale and moderate size problems, the heuristic algorithm and *meta*-heuristic algorithm have a satisfactory performance to find a desirable solution than exact methods. In the past few decades, the heuristic algorithm and *meta*-heuristic algorithm have been proven as an effective way to solve the BFSP, such as the profile fitting (PF) heuristic (McCormick, Pinedo, Shenker, & Wolf, 1989), the minimax (MM) heuristic (Ronconi, 2004), the Nawaz-Enscore-Ham (NEH) heuristic (Nawaz, Enscore, Emory, & Ham, 1983) and the branch-and-bound (b&b) algorithm (Ronconi & Armentano, 2001).

With the development of technology and science, the production model of various enterprise has changed from the single factory to the multi-factory (Li et al., 2018). The multi-factory production model improves the efficiency of production, increases the dependability of system and the flexibility of process simultaneous. Therefore, the distributed scheduling problem has become as a popular research topic (Pan, Gao, Li, et al., 2019). Recently, the distributed blocking flowshop scheduling problems (DBFSP) have attracted comprehensive attention from researchers and practitioner in various fields. The solution methods including heuristics and *meta*-heuristics (Shao, Pi, & Shao, 2020), have been proposed to solve the DBFSP. The heuristic has been testified as an effective way to solve the flowshop scheduling problem (Fernandez-Viagas et al., 2018). An iterated greedy algorithm, which is named IGA, is applied to solve the parallel flow shop scheduling problem (Ribas et al., 2019). A novel hybrid discrete differential evolution (HDDE) algorithm, which is based on the characteristics of discreteness and distribution features, is proposed by L. Wang, Pan, Suganthan, Wang, and Wang (2010) to solve the BFSP. A revised artificial immune system (RAIS) algorithm is presented by Lin and Ying (2013) for the BFSP. A populated local search strategy is introduced in differential evolution (DE_PLS) proposed by Tasgetiren, Pan, Kizilay, and Suer (2015) to enhance the exploitation and exploration of the DE_PLS for solving the DBFSP. In the latest study, Zhang et al. (2018) proposed a discrete DE (DDE) algorithm for the DBFSP with makespan minimization. Furthermore, Zhang and Xing (2019) proposed an enhanced DDE (CDE) algorithm for the distributed limited-buffer flowshop scheduling problem.

Although various algorithms have been proposed for solving the DBFSP, the optimal solution of the DBFSP is still a difficult problem to obtain the satisfactory solution sets in desirable time, especially for the large-scale instances (Pan & Wang, 2011). As a population-based metaheuristic algorithm, DE has the advantages of strong versatility, simple and effective mechanism to solve discrete combinatorial optimization problems. The application of DE algorithm for the scheduling problems is mainly through two schemes including encoding the scheduling solutions as numerical vectors and discretizing the evolutionary operators as the permutation representation.

In this paper, an ensemble discrete differential evolution algorithm (EDE) is proposed to solve the DBFSP with minimizing the makespan criterion. The main contributions in this paper are listed as follows.

- The coding and decoding mechanism of a classic single-shop scheduling solution is extended as a novel representation rule for multi-shop scheduling solutions. Furthermore, unlike traditional greedy selection strategy of DE, bias selection and elitism strategies are adopted to improve the global search capability of EDE.
- The problem-related implicit properties are extracted out as a prior knowledge to construct a framework to provide a relatively reliable initial solution for the EDE.
- The two competitive discrete mutation strategies, called $DE/rand/1$ and $DE/current-to-pbest/1$, are discretized and integrated into the EDE to balance the exploration and exploitation of the algorithm.

The remainder of this paper is organized as follows. The definition and details about the DBFSP are introduced in Section 2. The basic DE algorithm and the proposed EDE algorithm are described in Section 3. The numerical experiments and comparisons are presented in Section 4. The conclusion and some future works are suggested in Section 5.

2. Formulation of DBFSP

2.1. Problem description

The blocking flowshop scheduling problem (BFSP) is first described and analyzed with Graham, Lawler, Lenstra, Kan, and Rinnooy. (1979). n jobs from set $N = \{N_j | j = 1, 2, \cdots, n\}$ have to be sequenced through m machines from set $M = \{M_i | i = 1, 2, \cdots, n\}$ without intermediated buffers. The processing of job J_i requires a set of m operations $O_j = \{O_{j,i} | i = 1, 2, \cdots, m\}$, where $O_{j,i}$ is executed on machine M_i with processing time $P_{j,i}$, job J_i having completed operation $O_{j,i}$, has to remain on M_i until M_{i+1} becomes idle. All machines and jobs are available at time 0. In addition, the processing of jobs follows the same order on all machines. Each machine can dispose at only one job, and each job is processed on at most one machine at a time, the preemption of the machine is not allowed.

The distribute blocking flowshop scheduling problem is conceptually similar to the parallel blocking flowshop scheduling problem (Ribas, Companys, & Tort-Martorell, 2017). The problem is defined as follows: n jobs have to be scheduled in one of the F identical flowshop with m machines. Each factory is able to processes all jobs and each job is only processed by one machine at a time. Each machine processes only one job at a time. Preemption is not allowed. With the blocking constraint, a job does not deviate from the machine until the next machine is idle. A complete timetable of the DBFSP is divided into two interrelated parts: First, assign each job to factories. Second, sort the jobs in each factory. The final objective is to schedule the jobs to the different flowshop such that the maximum makespan (C_{max}) among them is minimized.

2.2. Mathematical model of the DBFSP

In DBFSP, a solution π is formed by the sequence of jobs in each flowshop ($\pi = (\sigma_1, \sigma_2, \cdots, \sigma_f)$). Let $c_{j,k,f}$ be the departure time of a job that occupies position k in machine j at factory f. Based on the MIP model of the DBFSP presented by Naderi and Ruiz (2010), the mathematical program is formalized as follows.

$$MinC_{max}(\pi) \tag{1}$$

$$\sum_{f=1}^{F}\sum_{k=1}^{n} x_{k,i,f} = 1 \quad i = 1, 2, \cdots, n \tag{2}$$

$$\sum_{i=1}^{n} x_{k,i,f} \le 1 \quad f = 1, 2, \cdots, F \quad k = 1, 2, \cdots, n \tag{3}$$

$$c_{1,k,f} \ge c_{1,k-1,f} + \sum_{i=1}^{n} x_{k,i,f} \cdot p_{1,i} \quad k = 1, 2, \cdots, n \quad f = 1, 2, \cdots, F \tag{4}$$

$$c_{j,k,f} \ge c_{j-1,k,f} + \sum_{i=1}^{n} x_{k,i,f} \cdot p_{j,i} \quad j = 2, 3, \cdots, m$$
$$k = 2, 3, \cdots, n \quad f = 1, 2, \cdots, F \tag{5}$$

$$c_{j,k,f} \ge c_{j+1,k-1,f} \quad j = 2, 3, \cdots, m-1 \quad k = 2, 3, \cdots, n$$
$$f = 1, 2, \cdots, F \tag{6}$$

$$c_{j,0,f} = 0 \quad j = 1, 2, \cdots, m \quad f = 1, 2, \cdots, F \tag{7}$$

$$c_{0,k,f} = 0 \quad k = 1, 2, \cdots, n \quad f = 1, 2, \cdots, F \tag{8}$$

$$C_{max} \ge C_{m,n,f} \quad f = 1, 2, \cdots, F \tag{9}$$

$$x_{k,i,f} \epsilon \{0, 1\} \quad k = 1, 2, \cdots, n \quad i = 1, 2, \cdots, n \quad f = 1, 2, \cdots, F \tag{10}$$

$$c_{j,k,f} \ge 0 \quad k = 1, 2, \cdots, n \quad i = 1, 2, \cdots, n \quad f = 1, 2, \cdots, F \tag{11}$$

where $x_{k,i,f}$ is a decision variable, if job i occupies position k in the sequence of flowshop f, and $x_{k,i,f} = 1$. The objective function is set at Eq. (1). Eq. (2) ensures that each job n must be exactly at one machine m and only at one factory f. Eq. (3) ensures that only one job at most is allocated to each position at a factory. The starting time of the job n which occupies position k in the first machine at factory f is defined in Eq. (4). The relationship of the starting time of each job n in two machines at the assigned factory is defined in Eq. (5). Eq. (6) calculates the starting time of a job n under the blocking situation, when the next machine has to be available. Eqs. (7) and (8) are the initial conditions. Eq. (9) defines the makespan. Finally, Eqs. (10) and (11) define the domain of the decision variable.

3. EDE for DBFSP with minimizing makespan

To simplify formulation exposition, the following notations are used:

m	Number of machines
n	Number of jobs
f	Number of factories
C	Time level, $T = 5, 10, 15$
i	Index of machines
j	Index of jobs
T	Time level
M	Maximum elapsed CPU time
C_{max}	Maximal completion time of the last job to be processed in any factory
g	Number of current generations
$x_{ri,g}$	$ri$is an exclusive and randomly selected number from $[1, Np]$, for different i
$x_{b,g}$	Best individual in the population in a generation g
$x_{pb,g}$	Randomly selected number from the top $N * p$in a generation g
F	Mutation parameter
Cr	Crossover rate

3.1. Traditional DE algorithm

Traditional DE is a stochastic *meta*-heuristic method introduced by Price (1996). It is one of the most effective and popular algorithms for solving complex numerical optimization problems and high-dimension problems. In addition, the developed DE and its variants have been applied in various real-world problems, for example, for solving the flowshop scheduling problems (Zhao, Liu, Zhang, Ma, & Zhang, 2017) or no-wait flowshop scheduling problems (Zhao, Liu, et al., 2018).

At each generation in DE, the three operators, named the mutation operator, the crossover operator and the selection operator, are executed on the target vector (Zamuda & Sosa, 2019). The procedure of the traditional DE algorithm is as follows.

Step. 1: Initialization phase.

Set mutation parameter F and crossover parameter Cr and randomly yield the initial target population with P_s. Let generation $g = 0$.

Step. 2: Mutation phase.

The mutation individual $v_{i,g}$ is generated by Eq. (12). Set $g = g + 1$.

$$DE/rand/1\, v_{i,g} = x_{r1,g} + F \cdot (x_{r2,g} - x_{r3,g}) \tag{12}$$

Step. 3: Crossover phase.

After the mutation operation, a trial vector $u_{i,g}$ is generated by the crossover operation between $x_{i,g}$ and $v_{i,g}$. $u_{i,g}$ and is defined as follows.

$$u_{i,g}^{j} = \begin{cases} v_{i,g}^{j} & if\ rand_j(0,1) \le Cr\ or\ j = j_{rand} \\ x_{i,g}^{j} & otherwise \end{cases} \tag{13}$$

Step. 4: Selection phase

The desirable vector between $x_{i,g}$ (the target vector) and $u_{i,g}$ (the trial vector) is guaranteed by the selection operation to access the next generation after the crossover operation. The selection operation is described as follows.

$$x_{i,g+1}^{j} = \begin{cases} x_{i,g}^{j} & if\ f(x_{i,g}^{j}) \le f(u_{i,g}^{j}) \\ u_{i,g}^{j} & otherwise \end{cases} \tag{14}$$

Step. 5: If the required termination condition has not been found, go back to Step. 2. Until the end termination condition is met, then output $x_{i,g+1}^{j}$.

3.2. EDE algorithm

Because of the continuous nature of the original DE algorithm, the traditional DE algorithm cannot be directly used to solve the combinational optimization problem with discrete characteristics such as the DBFSP (Baioletti, Milani, & Santucci, 2020). Therefore, a new method called the EDE algorithm is proposed to solve the distributed blocking flowshop scheduling problem. Furthermore, the solution representation, mutation strategy, crossover operation, selection phase and initialization are illustrated in this section.

3.2.1. Solution representation

For the DBFSP with n jobs and f factories, a candidate solution is expressed as $\pi = \{\pi_k | k = 1, 2, \cdots, f\}$, in which $\pi_k = \{J_{\pi_k(1)}, J_{\pi_k(2)}, \cdots, J_{\pi_k(n)}\}$ and $\sum_{k=1}^{f} n_k = n$. Thus, the processing sequence of jobs at factory is F_k.

Example 1. Consider a DBFSP with $f = 3$ and $n = 7$. A feasible solution is $\pi = \{\pi_1, \pi_2, \pi_3\}$, where $\pi_1 = [J_5, J_7, J_2]$, $\pi_2 = [J_1, J_3]$ and $\pi_3 = [J_6, J_4]$. When decoding, jobs J_2, J_5 and J_7 are assigned at factory F_1 and processed in the order $[J_5 \rightarrow J_7 \rightarrow J_2]$. Similarly, jobs J_1, J_3 are assigned at factory F_2 and jobs J_4, J_6 are assigned at factory F_3 and processed in the order $[J_1 \rightarrow J_3]$ and $[J_6 \rightarrow J_4]$.

3.2.2. Ensembled population initialization strategy

In the EDE algorithm, the population is initialized with three different types of strategies, and each method is assigned the same population size. The quality and diversity of the initial population are enhanced by the ensembled method. Three different strategies are described as follows.

Distributed NEH (NEH1) strategy: this method, proposed by Naderi and Ruiz (2010), is an adaptation of the NEH heuristic (Nawaz et al., 1983) to the DPFSP. There are four steps in the NEH1 method.

Step. 1: π_k is taken as the processing sequence of jobs at factory F_k and $\pi_k \neq \varnothing$. A job sequence $J = [J_1, J_2, \cdots, J_n]$ is generated according to the non-increasing order of the processing time (Naderi & Ruiz, 2010)of each job on each machine.

Step. 2: For each factory $F_k (k = 1, 2, \cdots, f)$, job J_k is assigned and π_k is updated.

Step. 3: Set $k = k + 1$. If $k \le n, J_k$ is assigned to all possible site of existing sequence at each factory, and then the job is placed to the position with the lowest local completion time.

Step. 4: If $k \le n$, repeat step 3. Then output π_k.

Trapeziums (TR) strategy: The TR strategy was proposed by Ribas et al. (2017) to solve the DBFSP. The procedure of TR strategy identical to that of the NEH1, except for the selection of the first job in each factory. In TR strategy, two indexes are calculated for each job according to Esq. (15) and (16). Afterwards, Eq. (17) is applied to calculate $R(i)$ for each job, and the jobs are sorted in increasing order. The job is selected with minimum $R(i)$ and put in the first position in sequence π_k.

$$S1_i = \sum_{j=1}^{m} (m - j) \cdot p_{j,i} \tag{15}$$

$$S2_i = \sum_{j=1}^{m} p_{j,i} \tag{16}$$

$$R(i) = 2 \cdot \lambda \cdot \frac{S1_i}{(m-1)} + (1 - \lambda) \cdot S2_i \tag{17}$$

The index $S1_i$ represents the front delay of each job and index $S2_i$ is the makespan of each job. When the first job is chosen in the sequence for each factory, it is necessary to consider the front delay, because only considering the job sort with the minimum sum of the processing time is not effective for minimizing the makespan.

Example 2. Considering the following machine $m = 3$ and job $n = 3$, the processing times of J_1, J_2 and J_3 on each machine are $J_1 = (3, 2, 1), J_2 = (1, 3, 2)$ and $J_3 = (2, 1, 3)$, respectively. Obviously, the sum of the processing time for J_1, J_2 and J_3 *is* 6. However, the makespan of schedule sets $\pi_1 = \{J_1, J_2, J_3\}$, $\pi_2 = \{J_2, J_1, J_3\}$ and $\pi_3 = \{J_3, J_1, J_2\}$ in the BFSP is 13, 10 and 12, respectively. As shown in Fig. 2, the difference between π_1, π_2 and π_3 is the front delay induced by the first job scheduled.

HPF2 strategy: The HPF2 strategy is an enhanced version of the TR strategy also proposed by Ribas et al. (2017). As shown in Fig. 2, when the blocking constraint is considered, the makespan is not only affect by the front delay but also decided by the block time, idle time or the sum of both. Therefore, it is necessary to consider the relationship between the block time, idle time and front delay.

In HPF2, the influence of each job on the makespan is considered from the block time, idle time and front delay. The main steps in HPF2 strategy are described as follows.

Step 1: Select the job with minimum $R(i)$, and put it in the first position of sequence π_k. Set $k = 1$.

Step 2: While $k < n$, the job with minimum $ind(i, k)$ was selected and assign it to all possible sites of the existing sequence at each factory, and then place the job to the position with minimum makespan. $k = k + 1$.

The $ind(i, k)$ is calculated by Eq. (18), where i denotes the job, k is the location and σ is the partial sequence of $k - 1$ jobs.

$$ind(i,k) = \rho \cdot \sum_{j=1}^{m} \left(c_{j,k}(\sigma * i) - c_{j,k-1}(\sigma) - p_{j,i}\right) + (1 - \rho) \cdot \left(c_{m,k}(\sigma * i) - c_{m,k-1}(\sigma)\right) \tag{18}$$

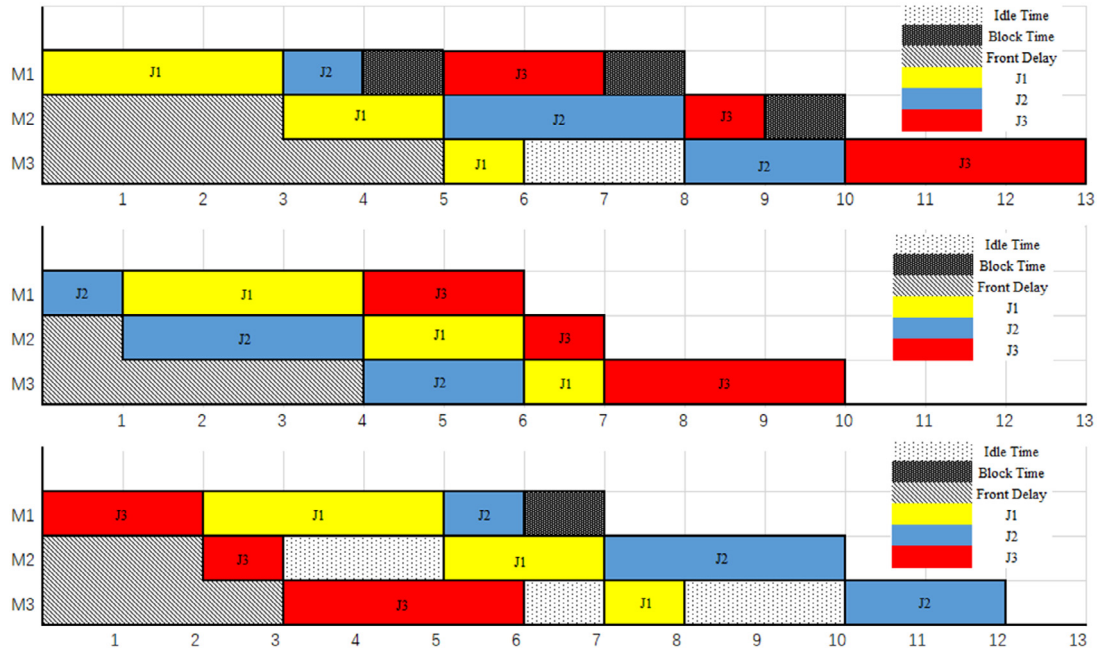


Fig. 2. Gantt chart of π_1, π_2 and π_3.

3.2.3. Competitive evolutionary mutation mechanism

The mutation strategies are randomly chosen from Eqs. (19) and (20) in the early $0.1 \cdot M$ seconds. Let $X = \{x_k | k = 1, 2, \cdots, f\}$ denote the target vector, and the mutant is denoted by $V = \{v_k | k = 1, 2, \cdots, f\}$. The mutant vector V is produced by Eqs. (19) and (20).

$$v_{i,g} = x_{i,g} \oplus F \otimes (x_{r1,g} \otimes x_{r2,g}) \tag{19}$$

$$v_{i,g} = x_{i,g} \oplus F \otimes (x_{pb,g} \ominus x_{i,g}) \oplus F \otimes (x_{r1,g} \ominus x_{r2,g}) \tag{20}$$

where F is the mutation factor, $x_{pb,g}$ is a randomly selected number from the top $N \cdot c$ in generation g, and $x_{ri,g}(ri = 1, 2)$ is an exclusive and randomly selected number from $[1, Np]$ and is different from i. N is a set of local optima in generation g and $c = 0.1$. Then let $x_{bl} = \{x_{bl}(1), x_{bl}(2), \cdots, x_{bl}(n)\}$ and $bl = r1, r2, i$.

Additionally, the competitive evolutionary mechanism is introduced in mutation operation, and the strategy that produced excellent individuals are allocated to larger population sizes. After $0.1 \cdot M$ seconds, if $s_{1,g}$ is greater than $s_{2,g}$, Eq. (19) is applied to generate $v_{i,g}$, else, Eq. (20) is applied. The success rate s_1, s_2 is set to 0.5 at the beginning and calculated as follows.

$$s_i = \frac{\sum_{i=1}^{g} c_{i,g}}{\sum_{i=1}^{g} k_{i,g} + \sum_{i=1}^{g} p_{i,g}} \tag{21}$$

where $c_{i,g}$ is set to zero at the beginning and $i = 1, 2$. Then, if a desirable factor is generated by Eq. (21), $c_{1,g} = c_{1,g} + 1$; otherwise, $c_{2,g} = c_{2,g} + 1$. $k_{i,g}$ and $p_{i,g}$ denote the number of times to use the first or second configuration in every generation. In addition, the population size assigned to each strategy is $s_i \cdot Np$ and is less than $0.9 \cdot Np$ in the same time.

For clarity, Eq. (19) is taken as an example, and the definitions of $\oplus$, $\ominus$ and $\otimes$ are separately described as follows.

$$\Delta = x_{r1} \ominus x_{r2} \Longleftrightarrow \delta_j = \begin{cases} x_{r1}(j), & if\ x_{r1}(j) \neq x_{r2}(j) \\ 0, & otherwise \end{cases} \tag{22}$$

$$\theta = F \otimes \Delta \Longleftrightarrow \theta_j = \begin{cases} \Delta(j), & if\ rand < F \\ 0, & otherwise \end{cases} \tag{23}$$

$$v_i = x_{i,g} \oplus \theta \Longleftrightarrow v_i = \begin{cases} swap, & if\ rand < 0.5 \\ insert, & otherwise \end{cases} \tag{24}$$

where $j = 1, 2, .., n$.

Example 3. To explain Eqs. (22)–(24), consider $x_{i,g} = \{5, 2, 4, 1, 3\}$, $x_{r1} = \{3, 1, 2, 5, 4\}$ and $x_{r2} = \{2, 1, 3, 4, 5\}$. The random number $rand(j_i)$ in Eqs. (23)–(25) is set in the order of 0.6, 0.1, 0.2, 0.9, 0.3. The mutation factor $F = 0.5$.

Step. 1:$\Delta = x_{r1,g} \theta x_{r2,g}$

In the θ operation, $x_{r1} = \{3, 1, 2, 5, 4\}$ and $x_{r2} = \{2, 1, 3, 4, 5\}$. According to Eq. (22), $x_{r1}(j_1) \neq x_{r2}(j_1)$, $\Delta(j_1) = 3$; $x_{r1}(j_2) = x_{r2}(j_2)$, and $\Delta(j_2) = 0$. By that analogy, $\Delta = \{3, 0, 2, 5, 4\}$.

Step.2: θ =F $\otimes$ Δ

In the $\otimes$ operation, $rand = \{0.6, 0.1, 0.2, 0.9, 0.3\}$ and $\Delta = \{3, 0, 2, 5, 4\}$. The mutation factor $F = 0.5$. According to *Eq.*(24), $rand(j_1) > F$ and $\theta(j_1) = 0$; $rand(j_2) < F$ and $\theta(j_2) = 0$. By that analogy, have $\theta = \{0, 0, 2, 0, 4\}$.

Step. 3: $v = x_{i,g} \oplus \theta$

In the $\oplus$ operation,$rand = \{0.6, 0.1, 0.2, 0.9, 0.3\}$, $x_{i,g} = \{5, 2, 4, 1, 3\}$ and $\theta = \{0, 0, 2, 0, 4\}$. It is found that $\theta(j_i) \neq 0$. According to Eq. (25), $rand(j_3) = 0.2 < 0.5$ and $\theta(j_3) = x_{i,g}(j_2) = 2$; swapping $x_{i,g}(j_2)$ and $x_{i,g}(j_3)$, have $v' = \{5, 4, 2, 1, 3\}$; $rand(j_5) = 0.9 > 0.5$, $\theta(j_5) = v'(j_2) = 4$ and $v'(j_5) = 3$; inserting $v'(j_5)$ after $v'(j_2)$, have $v = \{5, 4, 3, 2, 1\}$. Finally, v is the mutant vector.

3.2.4. Crossover operation

Let $X = \{x_k | k = 1, 2, .., f\}$ denote the target vector, and the corresponding mutant is denoted by $V = \{v_k | k = 1, 2, .., f\}$. In the crossover operation, the trial vector U is produced by combining X and V as follows.

Step. 1: Get μ_k from Eq. (25)

$$\mu_k = \begin{cases} \mu_k(i) = k, i = i + 1, & if\ rand > Cr \\ 0, & otherwise \end{cases} \tag{25}$$

where μ_k is a set of locations for performing crossover operations and $k = 1, 2, \cdots, n$. The crossover factor $Cr = 0.5$.

Step. 2: Let $U_1 = X$. The jobs $X_k(j)$ and $j \in \mu_k$, in U_1 are sequentially replaced starting from the first factory and following the order of their appearance in V. Then, in the same way, let $U_2 = V$. The jobs $V_k(j)$ and $j \in \mu_k$, in U_2 are sequentially replaced by the following of the order of their appearance in X.

Step. 3: Calculate the makespan of U_1 and U_2; make the better one as trial vector U.

Example 4. To explain the crossover operation, $x_{i,g}$ and rand(j_i) are set as described in Example 3, and the mutant vector is set as $V = \{3, 5, 1, 4, 2\}$. According to Eq. (25), $\mu_k = \{1, 3, 5\}$. Then, $x_{i,g}(\mu_1) = 5$, $x_{i,g}(\mu_2) = 4$ and $x_{i,g}(\mu_3) = 3$, following the order of their appearance in V; replaced sequentially, $3 \rightarrow 5 \rightarrow 4$, and have $U_1 = \{5, 2, 4, 3, 1\}$. In the same way, $V(1) = 3$, $V(3) = 1$ and $V(5) = 2$, replaced sequentially, $2 \rightarrow 1 \rightarrow 3$, have $U_2 = \{2, 5, 1, 4, 3\}$. Finally, the superior one in U_1 and U_2 was selected as the vector U.

3.2.5. Biased selection operation

In the EDE algorithm, the greedy selection operator is replaced by a biased selection operator. The diversity of the population in EDE algorithm was enhanced by the biased selection operation and avoids population being trapped in the local optimum. Under the minimizing makespan criterion, the new target individual $X_{i,g+1}$ is produced by Eq. (26).

$$X_{i,g+1} = \begin{cases} U, & if\ \omega < 0\ or\ rand < max\{\eta - \omega, 0\} \\ X, & otherwise \end{cases} \tag{26}$$

$$\omega = \frac{f(U) - f(X)}{f(X)} \tag{27}$$

where ω is the relative error of U with respect to X, η is the selection scale factor and $\eta \in (0, 1)$. In a biased selection strategy, if U is just slightly worse than X, U still has a small probability of entering into the next generation.

3.2.6. Elitism strategy

Although the biased selection operator improves the diversity of the population, the excellent individual is not entered in the next generation under certain probability, which delays the convergence speed of the algorithm (Nayak, Rout, Jagadev, & Swarnkar, 2017). Considering the convergence speed of EDE algorithm, the optimal individual is recorded in every generation. When the current optimal individual is worse than that of the last generation, the recorded optimal individual in the last generation is inserted in a random place in the current population.

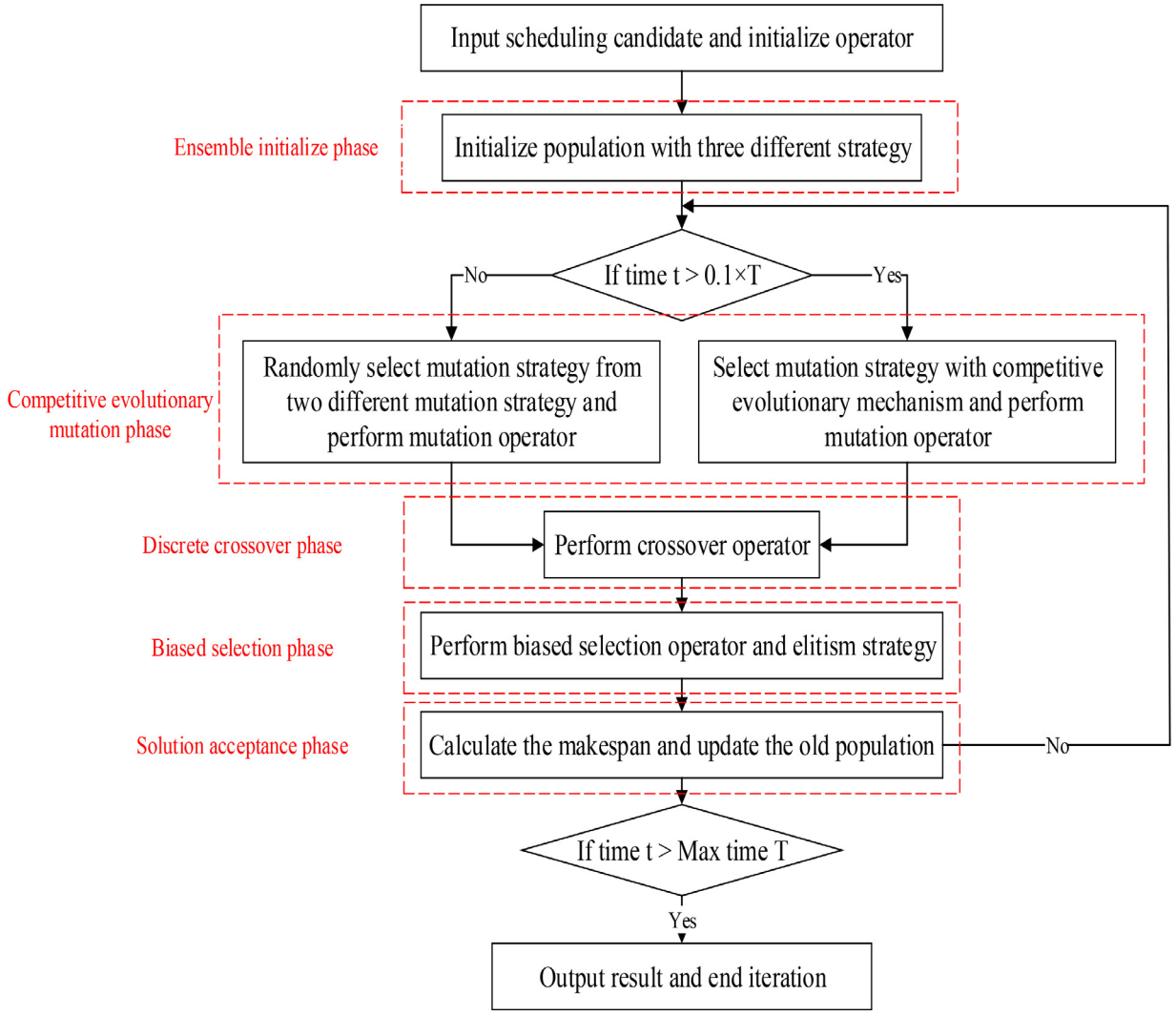


Fig. 3. Flowchart of EDE for DBFSP.

3.2.7. Procedure of the EDE algorithm

From the description of EDE, EDE holds the same parameter and algorithm framework as that of the DDE, DE_PLS and HDDE algorithms. Nevertheless, the optimization mechanisms of EDE, DDE and HDDE are completely different in the initialization, mutation, crossover and selection operation. In EDE algorithm, two heuristics method and one random strategy are used to produce a desirable initial solution with a certain quality and diversity. The competitive evolutionary mutation mechanism was introduced in mutation operation, which include two different mutation strategies. Moreover, the biased selection operation and elitism strategy was introduced in selection operation. In DDE algorithm, two methods were used to produce the initial population, and only one mutation strategy in mutation operation. In HDDE algorithm, a job-permutation-based mutation operator was introduced in mutation operation and an improved NEH method was used to initial population. In summary, from the description of EDE algorithm, EDE is a completely different algorithm from HDDE and DDE. Based on the above designs, the pseudo-code of EDE is shown in Algorithm 1 and the flowchart of EDE is shown in Fig. 3.

Algorithm 1: Main procedure of the EDE algorithm
Input: Test scheduling function
Output: Minimum makespan C_{max} and the schedule X_{best}
Initialize population with three different strategy and time $t = 0$
Initialize operator F, Cr and success rate s_1, s_2
While $t \leq T$
for $i = 1 to Np$ **do**
  if $t \leq 0.1 \cdot T$
    if $rand(0,1) < s_{1,j}$
  Randomly select mutation strategy from Eqs. (20) and (21)
    end if
    else
      if $s_{1,j} > s_{2,j}$
        using Eq. (20)
    else
        using Eq. (21)
    end if
  end if
  Generate trial vector U with Eq. (26)
  Update $X_{i,g+1}$ with Eq. (28) and $c_{1,g}$ and$c_{2,g}$
  Calculate s_1, s_2 with Eq. (22)
 end for
end while

4. Experiments and comparisons

In this section, to test the performance of the proposed EDE algorithm, a numerical experiment and comparison with other four state-of-the-art algorithms are presented based on the adaptation of the well-known flowshop benchmark set of Taillard (1993), as it was proposed in Naderi and Ruiz (2010). Afterwards, the parameters are calibrated with the DOE method. Finally, the comprehensive details of all the experiments are presented out in the following sections.

4.1. Experimental settings and tested methods

Naderi and Ruiz (2010) presented 420 small-scale instances of jobs $n = 2, 4, \cdots, 16$, machines $m = 2, 3, \cdots, 5$ and factories $F = 2, 3, 4$. They also proposed 720 large-scale instances after adding the number of factories $F = 2, 3, \cdots, 7$ to 120 well-known instances of Taillard (1993). More details are found at http://soa.iti.es/instancias-problemas.

All compared algorithms were coded in Matlab2016b and implemented on a PC with a 3.4 GHz Intel(R) Core™ i7-7700 CPU, 16 GB of RAM and 64-bit OS to guarantee that the comparison is fair. The results were measured by the index of average relative percent deviation (ARPD) which is calculated as follows.

$$ARPD = \frac{1}{R} \cdot \sum_{i=1}^{R} \frac{C_i - C_{min}}{C_{min}} \cdot 100 \tag{28}$$

where R is the number of runs. C_i is the solution generated by a specific algorithm in the ith experiment for a given instance. C_{min} is the minimum makespan found by all algorithm. Obviously, the smaller the value of ARPD is, the better the performance of the algorithm is.

To investigate the performance of the EDE algorithm with different parameters, the proposed algorithm is tested on the part of the benchmarks of the large-scale instances. In order to avoid parameter overfitting, the new instances are generated as follows. Based on the problem size of Taillard (TA) instances, the new test instance randomly intercepts instances of corresponding size on the VRF (Vallada, Ruiz, & Framinan, 2015) benchmark. Each scale generates five instances, and a total of $5 \times 12 = 60$ new test instances are generated.

The stopping criterion in this paper is set as the maximum elapsed CPU time $M = n \cdot m \cdot F \cdot T$ milliseconds, where T is a parameter that was tested at several levels. For the comparisons among the algorithms, set T in three levels for $T = 5, 15, 30$. In addition, $T = 30$ is used in the calibration experiments. All compared algorithms are independently run 10 times on the benchmark function, and each run is started from scratch. As a result, there are $(140 \times 3 + 120 \times 6) \times 10 \times 3 = 34200$ results for each algorithm and $34200 \times 4 = 136800$ results in total.

4.2. Parameter analysis of EDE

The selection of control operator and parameters setting play an important role in DE and its variants, which affects the performance and efficiency of DE algorithm (Brest, Maučec, & Bošković, 2017). Usually, the parameters setting is determined in two different methods, the previous literature and experimentally (Shao, Pi, Shao, Pi, Shao, & Yuan, 2019). For the previous literature, the parameters are determined by summarizing previous relevant literature. In particular, the setting and combination of parameters in the previous literature have a strong instructive significance, which further helps to narrow down the scope of the selection of parameters.

However, the first method often does not provide accurate values. For the second method, many potential values of parameters are tested to obtain the optimal calibration. Furthermore, in the second method, the sensitivity of parameters is analyzed by using several statistical methods. However, the second method often consumes considerable computational time due to testing all combinations of parameters, and its accuracy is influenced by the potential values of parameters. Therefore, the merits of these two methods are combined in this paper.

The performance of the EDE algorithm is dependent on the parameters that are set in the mutation operation, crossover operation and selection operation. The EDE has five critical parameters. NP (population size); F (mutation factor); Cr (crossover rate); λ (scale operator) and ρ (scale operator). Furthermore, the DOE approach (Montgomery, 2017) is employed to find the best levels for these factors. The One-way Analysis of Variance (ANOVA) pro-

cedure is used to analyze the obtained experimental results to determine the main effects and interaction of parameters, and then establish the best combination of parameters.

The levels for each factor are set as follows: $F = (0.2, 0.3, 0.4, 0.5)$, $Cr = (0.3, 0.5, 0.7)$, $NP = (20, 40, 60)$, $\lambda = (0.5, 0.6, 0.7)$ and $\rho = (0.4, 0.5, 0.6)$. The Time level T was set to 30. These parameters yield a total of $4 \times 3 \times 3 \times 3 \times 3 = 324$ different configurations for the EDE algorithm. Each configuration is executed 5 times. As a result, a total of $120 \times 324 \times 5 = 19440$ results are generated in this experiment.

The experimental results are analyzed by means of ANOVA function. In the DOE experiment, three main assumptions, i.e., normality, homogeneity of variance and independent of the residuals, are checked and accepted. In addition, the $F - ratio$ is a clear indicator of significance when the p values are less than the confidence level. The larger the $F - ratio$ is, the more effect the factor has on the response variable. The ANOVA results are reported in Table 1.

In Table 1, the p value of parameters Np, Cr, F and λ are below confidence level of $\alpha = 0.05$, which means that the EDE algorithm are sensitive to these parameters. In addition, Cr has the largest $F - ratio$, which indicates that Cr is the significant parameter that affects the performance of the proposed algorithm. Fig. 4 gives the main effects plot of all parameters. It is clearly observed that the preference choices are the combination of $F = 0.3$, $Cr = 0.7$, $NP = 20$, $\rho = 0.5$ and $\lambda = 0.5$.

Table 1
ANOVA results over calibrating the parameters of EDE.

Source	Sum of squares	Degrees of freedom	Mean Square	F-ratio	p-value
ρ	0.1991	3	0.06636	0.79	0.5029
λ	0.1057	2	0.05286	0.63	0.5357
Np	3.842	2	1.9921	2.27	**0.0949**
F	4.1181	2	2.05905	24.37	**0.0135**
Cr	12.6016	2	6.30079	74.58	**0.0001**
$\rho \cdot \lambda$	0.5073	6	0.08455	1.10	0.4252
$\rho \cdot Np$	0.2746	6	0.04577	0.54	0.7762
$\rho \cdot F$	0.0866	6	0.01443	0.17	0.9844
$\rho \cdot Cr$	0.3376	6	0.05627	0.67	0.6772
$\lambda \cdot NP$	1.0228	4	0.2557	3.03	**0.0183**
$\lambda \cdot F$	0.9245	4	0.23112	2.74	0.0294
$\lambda \cdot Cr$	0.4669	4	0.11673	1.38	0.2406
$Np \cdot F$	2.128	4	0.53201	6.3	0.1593
$Np \cdot Cr$	4.3024	4	1.07561	12.73	**0.0001**
$F \cdot Cr$	3.6979	4	0.92447	10.94	**0.0001**
Residual	22.305	264	0.08449		
Total	53.4624	323			

The 2-level interactions of all parameters are shown in Table 1, the $p - value$ of $F \cdot \lambda$, $Cr \cdot NP$ in Table 1 are less than 0.05. Thus, it is necessary to consider the interactions between F and λ, Cr and NP. The interaction plot of parameters is showed in Fig. 5.

However, although the optimal values and interaction plot for each parameter are shown in Figs. 4 and 5, but it is still not confirmed whether the lower Np, Cr and λ obtain better performance. Therefore, an additional experiment was performed to find the optimal value of Np, Cr and λ. In the additional experiment, set $Np \leq 20$, since 20 is the current optimal value for Np in the previous experiment. The potential values of Np are set {5 10 15 20}. Similarly, the potential values of Cr are set {0.1 0.2 0.3} and λ are set {0.2 0.3 0.4 0.5}. These parameters yield a total of $4 \times 3 \times 4 = 48$ different configurations for the EDE algorithm. Set the other two parameters F and ρ according to the main effect plot. The new 60 instances were generated in the same method in Section 4.1. As a result, a total of $48 \times 60 \times 5 = 14,400$ results are generated in this experiment. All of the results mentioned above are also analyzed by the ANOVA technique. The analysis results are reported in the ANOVA plots with 95% confidence intervals in Fig. 6.

In Fig. 6, the experiment result show that the best combination of parameters is $Np = 20$, $Cr = 0.3$ and $\lambda = 0.5$, which is in consistent with Figs. 4 and 5. Based on the analysis above, the optimum combination of parameters in EDE are suggested as follows: $F = 0.3$, $Cr = 0.3$, $NP = 20$, $\lambda = 0.5$ and $\rho = 0.5$.

4.3. Operational analysis

To test the validity of ensembled strategies in the EDE algorithm, compared the ensembled population initialization (EI) strategy with three different single population initialization methods, the NEH1 method, TR strategy method and HPF2 method. The ARPD results of NEH1, TR, HPF2 and EI in different function types of the large-scale instances of Ta001 to Ta120 are shown as Table 2, and the best result for each function is shown in boldface. The factory $F = 2, 3, \cdots, 7$ and $T = 30$ level and all compared algorithms are independently run 5 times on benchmark functions.

As the results shown in Table 2, the ensembled initialization strategy was a promising way to product a desirable solution. In order to further analyze the component performance of EDE algorithm, the EDE algorithm is compared with two different discrete

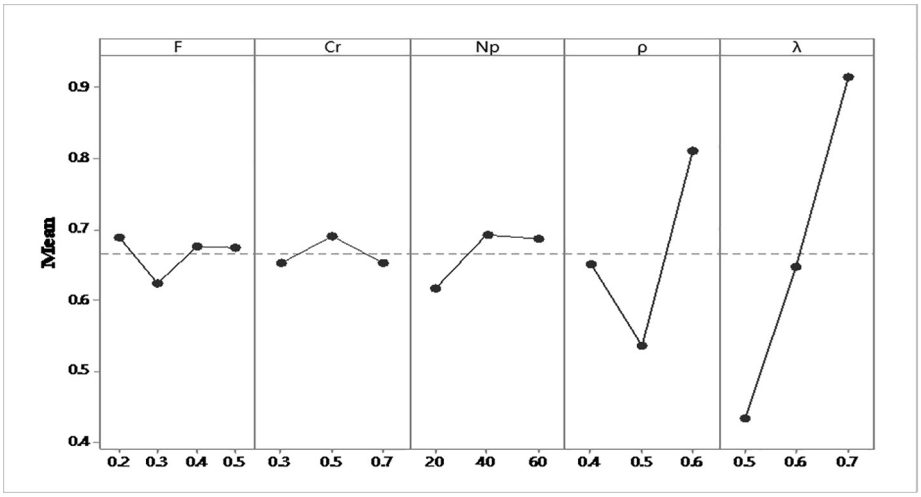
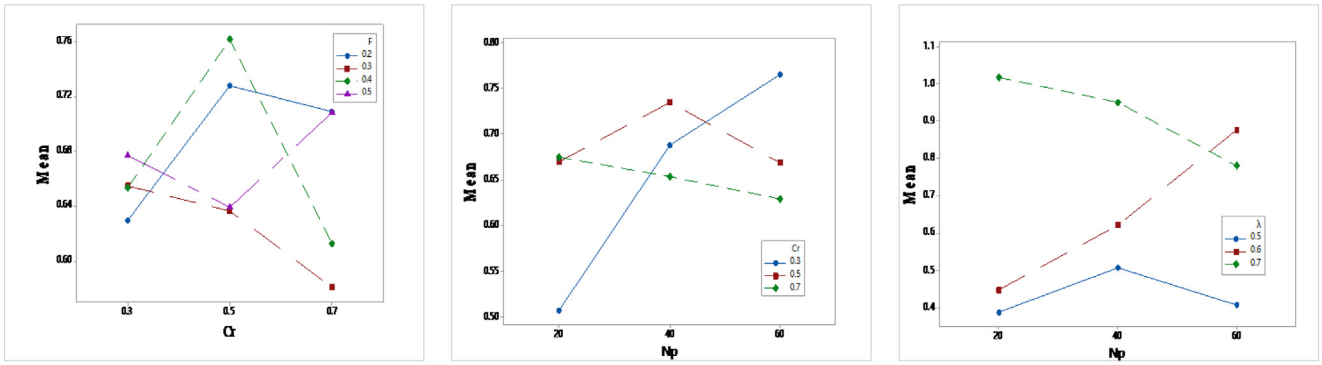


Fig. 4. Main effects plot of parameters.



(a) F and Cr (b) Cr and NP (c) λ and NP

Fig. 5. Interaction plot of parameters.

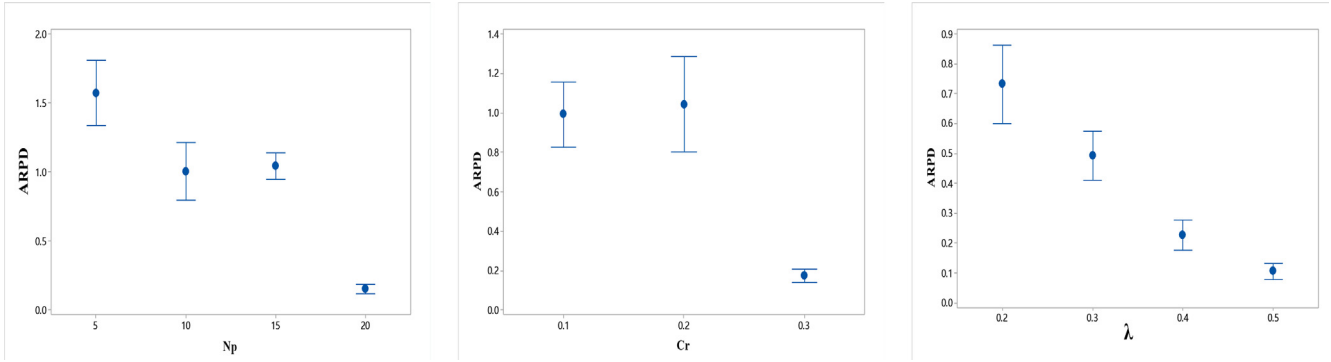


Fig. 6. Means plot and 95% confidence level intervals of Np,. Cr and λ.

Table 2
ARPD results for the test algorithms HPF2, NEH1, TR and EI.

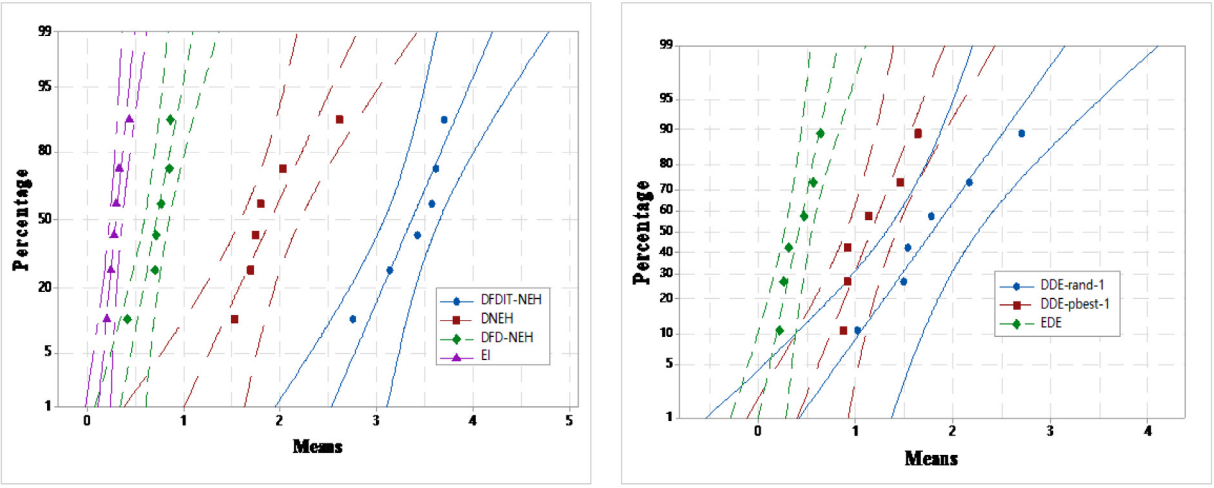
T	Factory	HPF2	NEH1	TR	EI
30	2	2.754	1.532	0.843	**0.431**
	3	3.131	1.796	0.692	**0.275**
	4	3.426	1.683	0.713	**0.328**
	5	3.570	2.031	0.862	**0.239**
	6	3.611	1.736	0.413	**0.196**
	7	3.703	2.614	0.765	**0.293**
Average		3.366	1.898	0.714	**0.293**

Table 3
ARPD results for the test algorithms DDE-rand-1, DDE-pbest-1 and EDE.

T	Factory	DDE-rand-1	DDE-pbest-1	EDE
30	2	1.017	0.871	**0.213**
	3	1.490	0.907	**0.254**
	4	1.533	0.913	**0.307**
	5	1.772	1.134	**0.472**
	6	2.165	1.456	**0.563**
	7	2.704	1.639	**0.641**
Average		1.780	1.153	**0.408**

single mutation strategy methods, the discrete $DE/rand/1$ (DDE-rand-1) strategy and the $DE/current-to-pbest/1$ (DDE-pbest-1).

From Table 2, Table 3 and Fig. 7, the effectiveness of the proposed ensembled method is demonstrated by the experimental results. The proposed EDE algorithm is superior to the compared algorithm in all factory levels. The reason is that the ensembled strategies enhance the exploration ability of EDE to avoid falling into local optima. In summary, the performance of EDE is effectively improved by introducing the ensembled strategies in population initialization and mutation operation.



(a) (b)

Fig. 7. Probability plot of the compared algorithms.

Table 4
Parameter settings.

Algorithms	Authors (Year)	Parameter settings
HDDE	L. Wang et al. (2010)	$F = 0.2, Cr = 0.2, Np = 20, P_l = 0.2$
IG2S	Ruiz et al. (2019)	$T = 0.2, d = 5, \rho = 0.95, d2 = 6$
DE_PLS	Tasgetiren et al. (2015)	$F = 0.1, Cr = 0.1, \beta = 0.0005, Np = 10, \delta = 20$
DDE	Zhang et al. (2018)	Small-scale instance $P_s = 50, \kappa = 0.2, C_p = 0.1, \lambda = 0.02$ Large-scale instance$P_s = 100, \kappa = 0.2, C_p = 0.3, \lambda = 0.02$

Table 5
ARPD results for the test algorithms by CPU time limit T and number of factories of F in small-scale instances.

T	Factory	HDDE	DE_PLS	DDE	IG2S	EDE
5	2	0.761	0.432	**0**	**0**	**0**
	3	0.654	0.371	**0**	**0**	**0**
	4	0.860	0.533	0.030	0.024	**0.020**
Average		0.759	0.445	0.010	**0**.008	**0.007**
15	2	0.413	0.203	**0**	**0**	**0**
	3	0.537	0.187	0.013	**0**	**0**
	4	0.576	0.131	0.023	**0.018**	**0.013**
Average		0.509	0.174	0.012	0.006	**0.003**
30	2	0.307	0.217	**0**	**0**	**0**
	3	0.457	0.178	0.009	**0**	**0**
	4	0.409	0.111	0.015	0.012	**0.010**
Average		0.391	0.169	0.008	0.004	**0.003**
Tot. average		0.553	0.263	0.010	0.006	**0.004**

4.4. Comparison of the proposed algorithms for small-scale instances

The EDE algorithm is compared with four state-of-the-art algorithms, HDDE, IG2S, DE_PLS and DDE, using 1260 small-scale test functions. The competitive algorithms are listed in Table 4, and the parameters of the compared algorithms are shown in Table 4.

The ARPD for each method, grouped per T and F, are listed in Table 5. Each cell within the table is the average RPD over 1400 results.

According to Table 5, the results show that the proposed EDE algorithm is superior to the compared algorithm in all levels of the small-scale instance.

4.5. Comparison of the proposed algorithms for large-scale instances

In this section, the EDE algorithm is further extended to 2160 large-scale instances. According to the number of F, n and m, the summarized results are exhibited in Table 6. Each cell within the table is the average RPD over 1200 results. More details are illustrated in Tables 7–9. Meanwhile, the Fisher's least-significant difference (LSD) interval for the EDE and the compared algorithms are presented in Fig. 8–10.

According to Tables 5 and 6, the EDE algorithm shows excellent performance for solving the DBFSP in large-scale and small-scale instances. In addition, the performance of the algorithms is propor-

Table 6
ARPD results for the test algorithms by CPU time limit T and number of factories of F in large-scale instances.

T	Factory	HDDE		DE_PLS		DDE		IG2S		EDE	
		ARPD	SD.	ARPD	SD.	ARPD	SD.	ARPD	SD.	ARPD	SD.
5	2	0.573	0.574	0.366	0.545	0.373	0.282	0.297	0.231	**0.228**	0.261
	3	0.948	0.829	0.438	0.781	0.532	0.420	0.346	0.402	**0.265**	0.378
	4	0.853	1.004	0.802	0.969	0.676	0.407	0.186	0.163	**0.127**	0.358
	5	1.731	0.962	1.022	0.909	1.082	0.409	0.32	0.215	**0.217**	0.353
	6	1.407	0.997	0.600	0.851	0.309	0.435	0.23	0.331	**0.133**	0.328
	7	0.83	0.980	0.591	0.904	0.314	0.431	0.23	0.421	**0.142**	0.904
Average		1.057	0.891	0.636	0.827	0.547	0.398	0.268	0.294	**0.185**	0.430
		ARPD	SD.	ARPD	SD.	ARPD	SD.	ARPD	SD.	ARPD	SD.
15	2	0.835	0.691	0.49	0.573	0.52	0.544	0.248	0.307	**0.183**	0.290
	3	0.796	0.830	0.425	0.826	0.516	0.752	0.381	0.416	**0.282**	0.386
	4	1.185	0.932	0.924	0.999	0.919	0.731	0.249	0.432	**0.153**	0.389
	5	1.871	0.937	1.101	0.957	1.069	0.796	0.286	0.411	**0.226**	0.354
	6	0.743	1.028	0.533	0.996	0.307	0.772	0.225	0.398	**0.132**	0.357
	7	0.859	1.001	0.596	0.983	0.313	0.843	0.286	0.501	**0.173**	0.345
Average		1.048	0.903	0.678	0.889	0.607	0.740	0.279	0.411	**0.191**	0.350
		ARPD	SD.	ARPD	SD.	ARPD	SD.	ARPD	SD.	ARPD	SD.
30	2	0.904	0.618	0.486	2.629	0.509	0.754	0.266	0.631	**0.206**	0.594
	3	1.325	0.822	1.305	0.728	0.54	0.826	0.527	0.841	**0.431**	0.748
	4	1.53	0.993	1.013	0.791	1.121	0.927	0.297	0.532	**0.19**	0.727
	5	1.909	0.948	1.071	0.766	1.085	0.928	0.293	0.673	**0.216**	0.788
	6	0.831	0.986	0.602	0.863	0.335	1.017	0.215	0.615	**0.155**	0.763
	7	0.908	0.972	0.622	0.970	0.361	0.990	0.235	0.672	**0.146**	0.834
Average		1.235	0.890	0.850	1.125	0.658	0.907	0.305	0.660	**0.224**	0.742
Total average		1.113	0.895	0.721	0.947	0.604	0.681	0.284	0.455	**0.200**	0.507

Table 7
ARPD results for the test algorithms by CPU time limit $T = 5$ and number of factories of F.

$n * m$	$F = 2$					$F = 3$					$F = 4$				
	HDDE	DE_PLS	DDE	IG2S	EDE	HDDE	DE_PLS	DDE	IG2S	EDE	HDDE	DE_PLS	DDE	IG2S	EDE
20×5	1.243	0.315	0.378	0.066	**0.03**	1.258	0.632	0.699	0.189	**0.157**	1.352	0.544	0.614	0.273	**0.193**
20×10	1.057	0.349	0.5	0.067	**0.061**	0.989	0.347	0.651	0.352	**0.286**	1.198	0.53	0.304	0.356	**0.288**
20×20	0.457	0.258	0.657	0.241	**0.126**	0.511	0.514	0.314	0.23	**0.156**	0.853	0.439	0.325	0.295	**0.181**
50×5	0.802	0.564	0.512	0.342	**0.269**	0.743	0.483	0.716	0.187	**0.182**	1.153	0.592	0.666	0.255	**0.214**
50×10	0.474	0.307	0.329	0.337	**0.293**	0.949	0.709	0.386	0.142	**0.097**	0.966	0.358	0.454	0.346	**0.262**
50×20	0.344	**0**.361	0.391	**0.275**	0.333	0.785	0.305	0.373	0.16	**0.139**	0.661	0.427	0.513	0.078	**0.070**
100×5	0.483	0.544	0.654	**0.294**	0.574	0.836	0.268	0.549	0.28	**0.159**	0.561	0.605	0.262	0.333	**0.17**
100×10	0.43	0.396	0.34	0.268	**0.225**	0.783	0.751	0.534	0.216	**0.083**	1.078	0.422	0.885	0.183	**0.177**
100×20	0.305	0.341	**0.145**	0.197	0.178	1.19	0.27	0.369	0.159	**0.074**	0.697	0.483	0.456	0.206	**0.117**
200×10	0.418	0.288	0.306	0.226	**0.205**	0.672	0.552	0.651	0.337	**0.266**	0.859	0.632	0.683	0.247	**0.203**
200×20	0.473	0.243	0.263	0.251	**0.151**	0.604	0.347	0.348	0.317	**0.251**	0.689	0.516	0.363	0.205	**0.200**
500×20	0.386	0.512	0.365	0.398	**0.291**	0.703	0.706	0.649	0.411	**0.346**	0.783	0.411	0.585	**0.287**	0.296
Avg	0.572	0.366	0.373	0.296	**0.228**	0.835	0.490	0.519	0.248	**0.183**	0.904	0.486	0.509	0.265	0.205
$n * m$	$F = 5$					$F = 6$					$F = 7$				
	HDDE	DE_PLS	DDE	IG2S	EDE	HDDE	DE_PLS	DDE	IG2S	EDE	HDDE	DE_PLS	DDE	IG2S	EDE
20×5	2.022	0.506	0.675	0.173	**0.088**	0.662	**0.301**	0.79	0.408	0.304	1.89	1.083	0.753	0.147	**0.102**
20×10	1.076	0.486	0.879	0.227	**0.179**	0.732	0.451	0.406	0.454	**0.304**	1.747	0.616	**0.159**	0.406	0.306
20×20	0.573	0.487	0.383	0.233	**0.118**	0.465	0.315	0.31	0.24	**0.148**	2.407	**0.3**	0.469	0.364	0.345
50×5	0.995	0.352	0.696	0.252	**0.133**	0.804	0.451	0.751	0.564	**0.41**	1.415	2.566	**0.262**	0.935	0.801
50×10	1.513	0.297	0.487	0.411	**0.276**	0.607	0.473	0.51	0.278	**0.246**	1.304	1.998	0.748	0.585	**0.462**
50×20	0.961	0.384	0.403	0.197	**0.114**	1.04	0.439	0.532	0.367	**0.299**	1.298	1.683	0.731	0.457	**0.343**
100×5	0.584	0.529	0.371	**0.271**	0.311	1.31	0.587	0.891	0.609	**0.491**	1.279	2.158	**0.515**	0.989	0.856
100×10	1.189	0.48	0.606	0.288	**0.242**	0.77	0.516	**0.283**	0.36	0.328	1.189	1.35	0.749	0.736	**0.614**
100×20	0.809	0.361	0.319	0.261	**0.243**	0.987	0.39	0.384	0.065	**0.024**	1.007	1.461	0.924	0.469	**0.322**
200×10	0.561	0.526	0.568	**0.464**	0.548	0.912	0.625	0.484	0.478	**0.326**	0.825	0.889	0.373	0.35	**0.242**
200×20	0.76	0.572	0.768	**0.356**	0.367	0.601	0.535	0.452	0.298	**0.149**	0.893	0.87	**0.276**	0.441	0.405
500×20	0.830	0.487	0.434	**0.337**	0.566	0.661	0.244	**0.164**	0.454	0.36	0.648	0.681	0.515	0.446	**0.378**
Avg	0.948	0.437	0.532	0.345	**0.265**	0.795	0.424	0.515	0.381	**0.282**	1.325	1.304	0.539	0.527	0.431

tional to time T. The reason is that DE is a population-based algorithm and takes time to evolve iteratively. However, in the same amount of time T, the total average of ARPD value is 0.507 by EDE, which remarkably outperforms the other four compared algorithms.

The Bonferroni-Dunn's test method was introduced to calculate the critical difference for comparing the differences of the algorithms with $\alpha = 0.05$ and $\alpha = 0.1$. The ranking of the algorithms is obtained through the Friedman test and the Bonferroni-Dunn's test. To detect the significant difference between the EDE algo-

Table 8
ARPD results for the test algorithms by CPU time limit $T = 15$ and number of factories of F.

$n*m$	$F=2$					$F=3$					$F=4$				
	HDDE	DE_PLS	DDE	IG2S	EDE	HDDE	DE_PLS	DDE	IG2S	EDE	HDDE	DE_PLS	DDE	IG2S	EDE
20×5	1.458	1.125	1.301	0.283	**0.235**	1.897	1.568	1.029	0.32	**0.203**	1.936	1.569	1.321	0.42	**0.307**
20×10	1.696	1.348	0.907	0.218	**0.21**	1.795	1.074	1.276	0.309	**0.251**	2.145	0.848	1.281	0.41	**0.267**
20×20	1.236	0.64	0.606	0.228	**0.209**	1.563	0.739	0.791	0.351	**0.236**	2.017	0.665	1.079	0.35	**0.224**
50×5	0.924	1.028	0.628	0.242	**0.186**	1.174	0.85	0.986	0.302	**0.187**	1.458	1.534	1.254	0.198	**0.129**
50×10	0.747	0.924	0.638	0.107	**0.106**	1.524	0.978	0.989	0.251	**0.111**	1.367	1.095	1.172	0.363	**0.266**
50×20	0.736	0.688	0.557	0.149	**0.075**	1.028	0.75	0.769	0.213	**0.198**	1.665	0.903	1.066	0.225	**0.185**
100×5	0.888	0.723	0.685	0.153	**0.047**	1.109	0.845	1.118	0.188	**0.148**	1.689	1.385	1.278	0.29	**0.184**
100×10	0.697	0.743	0.616	0.082	**0.042**	1.139	0.791	0.89	0.281	**0.123**	1.26	1.152	0.973	0.299	**0.135**
100×20	0.499	0.418	0.494	0.215	**0.136**	0.797	0.994	0.749	0.214	**0.068**	1.683	0.727	1.146	0.303	**0.148**
200×10	0.465	0.778	0.457	0.237	**0.109**	0.855	1.036	0.927	0.116	**0.069**	1.238	0.98	1.193	0.209	**0.188**
200×20	0.502	0.574	0.606	0.144	**0.071**	0.822	0.754	0.837	0.192	**0.129**	1.37	0.629	0.828	0.276	**0.139**
500×20	0.384	0.637	0.613	0.173	**0.097**	0.521	0.713	0.666	0.245	**0.112**	0.53	0.666	0.859	0.223	**0.102**
Avg	0.852	0.802	0.675	0.185	**0.126**	1.185	0.924	0.918	0.248	**0.152**	1.529	1.012	1.1208	0.297	**0.189**
$n*m$	$F=5$					$F=6$					$F=7$				
	HDDE	DE_PLS	DDE	IG2S	EDE	HDDE	DE_PLS	DDE	IG2S	EDE	HDDE	DE_PLS	DDE	IG2S	EDE
20×5	2.126	1.013	1.633	0.478	**0.348**	2.776	1.849	1.273	0.314	**0.292**	2.641	1.189	1.48	0.402	**0.313**
20×10	2.686	1.019	1.111	0.418	**0.27**	2.102	1.075	1.133	0.332	**0.318**	2.18	1.081	1.272	0.3	**0.211**
20×20	2.242	0.881	0.89	0.227	**0.201**	2.26	0.734	0.875	0.412	**0.258**	1.576	0.878	0.995	0.345	**0.223**
50×5	2.233	1.405	1.431	0.326	**0.239**	1.785	1.158	1.543	0.361	**0.246**	2.907	1.521	1.146	0.264	**0.247**
50×10	1.794	1.307	1.27	0.394	**0.238**	2.382	1.126	1.304	0.267	**0.245**	2.701	1.468	0.938	0.399	**0.273**
50×20	1.628	0.909	0.909	0.243	**0.184**	2.036	0.815	0.891	0.294	**0.28**	2.022	0.836	1.115	0.295	**0.247**
100×5	1.546	1.206	1.014	0.384	**0.247**	1.743	1.023	1.372	0.226	**0.208**	2.20	1.638	1.356	0.252	**0.237**
100×10	1.507	1.153	1.33	0.335	**0.254**	1.832	1.274	1.359	0.27	**0.179**	1.992	1.048	1.176	0.331	**0.234**
100×20	1.214	1.009	0.834	0.258	**0.149**	1.829	0.772	0.678	0.302	**0.241**	1.452	1.035	0.721	0.284	**0.146**
200×10	0.982	0.808	1.025	0.272	**0.212**	1.438	1.335	0.843	0.258	**0.142**	1.284	0.781	0.913	0.192	**0.170**
200×20	1.575	0.99	0.901	0.275	**0.12**	1.241	1.158	0.902	0.307	**0.177**	1.313	0.757	1.162	0.332	**0.166**
500×20	1.239	0.569	0.631	0.23	**0.141**	1.03	0.892	0.653	0.207	**0.128**	0.644	0.622	0.75	0.124	**0.119**
Avg	1.731	1.022	1.081	0.32	**0.216**	1.871	1.102	1.068	0.286	**0.226**	1.909	1.071	1.085	0.293	**0.215**

Table 9
ARPD results for the test algorithms by CPU time limit $T = 30$ and number of factories of F.

$n*m$	$F=2$					$F=3$					$F=4$				
	HDDE	DE_PLS	DDE	IG2S	EDE	HDDE	DE_PLS	DDE	IG2S	EDE	HDDE	DE_PLS	DDE	IG2S	EDE
20×5	2.919	1.15	0.601	0.339	**0.204**	1.038	0.727	0.415	0.223	**0.075**	1.247	0.868	0.48	0.113	**0.101**
20×10	2.326	0.964	0.442	0.061	**0.049**	0.945	0.75	0.42	0.368	**0.207**	0.85	0.636	**0.221**	0.256	0.256
20×20	2.1	0.694	0.339	0.332	**0.182**	0.676	0.422	0.264	0.2	**0.131**	0.675	0.487	0.223	0.125	**0.087**
50×5	2.13	0.723	0.353	0.424	**0.263**	0.663	0.574	0.251	0.156	**0.134**	1.18	0.829	0.445	0.366	**0.297**
50×10	1.795	0.719	0.41	0.194	**0.102**	0.835	0.507	0.307	0.202	**0.151**	0.967	0.58	0.254	0.182	**0.089**
50×20	1.429	0.536	0.254	0.31	**0.204**	0.613	0.413	0.181	0.156	**0.129**	0.711	0.472	0.325	0.256	**0.125**
100×5	1.115	0.52	0.229	0.164	**0.101**	0.798	0.57	0.22	0.267	**0.111**	0.875	0.714	0.55	0.255	**0.135**
100×10	0.752	0.394	0.159	0.298	**0.141**	0.804	0.563	0.313	0.298	**0.14**	0.8	0.546	**0.206**	0.269	0.238
100×20	0.598	0.357	0.307	0.21	**0.125**	0.66	0.544	0.342	0.291	**0.145**	0.654	0.619	0.397	0.158	**0.079**
200×10	0.529	0.379	0.172	0.08	**0.053**	0.63	0.472	0.405	0.148	**0.129**	0.863	0.564	0.348	0.292	**0.201**
200×20	0.508	0.379	0.173	0.176	**0.073**	0.749	0.402	0.276	0.303	**0.145**	0.669	0.572	0.391	0.168	**0.106**
500×20	0.678	0.386	0.267	0.170	**0.095**	0.507	0.447	0.293	0.086	**0.082**	0.475	0.336	0.184	0.145	**0.14**
Avg	1.406	0.601	0.308	0.229	**0.132**	0.743	0.532	0.307	0.224	**0.131**	0.830	0.601	0.335	0.215	**0.154**
$n*m$	$F=5$					$F=6$					$F=7$				
	HDDE	DE_PLS	DDE	IG2S	EDE	HDDE	DE_PLS	DDE	IG2S	EDE	HDDE	DE_PLS	DDE	IG2S	EDE
20×5	1.111	0.689	0.34	0.287	**0.181**	1.438	0.955	0.582	0.141	**0.091**	0.987	0.692	0.403	0.266	**0.144**
20×10	0.89	0.563	0.298	0.215	**0.165**	0.902	0.733	0.393	0.271	**0.213**	1.043	0.633	0.403	0.24	**0.114**
20×20	0.544	0.512	0.188	0.131	**0.089**	0.574	0.49	0.304	0.214	**0.089**	0.714	0.563	0.326	0.179	**0.092**
50×5	1.206	0.778	0.532	0.201	**0.124**	1.004	0.649	0.28	0.25	**0.165**	1.149	0.915	0.519	0.273	**0.141**
50×10	1.051	0.756	0.409	0.298	**0.209**	1.018	0.653	0.333	0.3	**0.173**	1.008	0.633	0.459	0.092	**0.09**
50×20	0.788	0.561	0.330	0.287	**0.132**	0.708	0.495	0.176	0.292	**0.142**	0.819	0.547	**0.26**	0.293	0.261
100×5	0.832	0.554	**0.159**	0.233	0.173	1.006	0.827	0.375	0.314	**0.182**	1.212	0.751	0.396	0.271	**0.200**
100×10	0.744	0.67	0.318	0.22	**0.143**	0.897	0.593	0.277	0.347	**0.213**	0.901	0.581	0.374	0.188	**0.159**
100×20	0.902	0.444	0.329	0.219	**0.097**	0.593	0.333	**0.165**	0.33	0.168	0.755	0.643	0.223	0.173	**0.109**
200×10	0.664	0.591	0.318	0.245	**0.175**	0.851	0.496	**0.295**	0.386	0.312	0.912	0.57	0.358	0.227	**0.093**
200×20	0.732	0.498	0.313	0.322	**0.163**	0.769	0.510	0.295	0.298	**0.173**	0.675	0.469	0.27	0.32	**0.203**
500×20	0.498	0.473	0.239	0.099	**0.051**	0.548	0.422	0.280	0.283	**0.153**	0.721	0.472	0.337	0.301	**0.151**
Avg	0.830	0.590	0.314	0.229	**0.141**	0.858	0.596	0.312	0.285	**0.172**	0.907	0.622	0.360	0.235	**0.146**

rithm and other compared algorithms, Wilcoxon's test was introduced, and the algorithms were compared in pairs. The results are shown in Table 10. In Table 10, $R+$ indicates that the EDE algorithm was superior to the comparison algorithm; in contrast, $R-$ means that the comparison algorithm was better than the EDE algorithm, and $R\approx$ means that the comparison algorithm has a similar value to the EDE algorithm. The results of Friedman test are showed in Figs. 11–16.

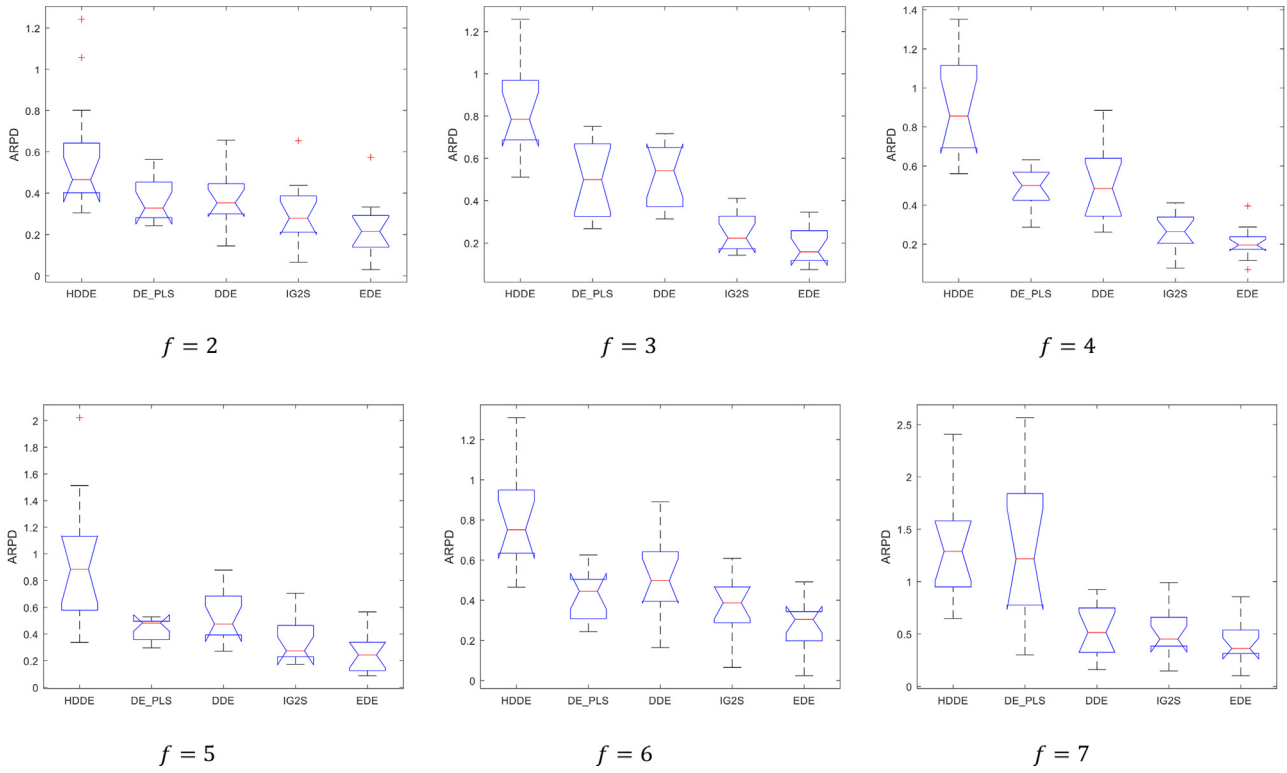


Fig. 8. Mean and 95% LSD interval for the test algorithms in $T = 5$.

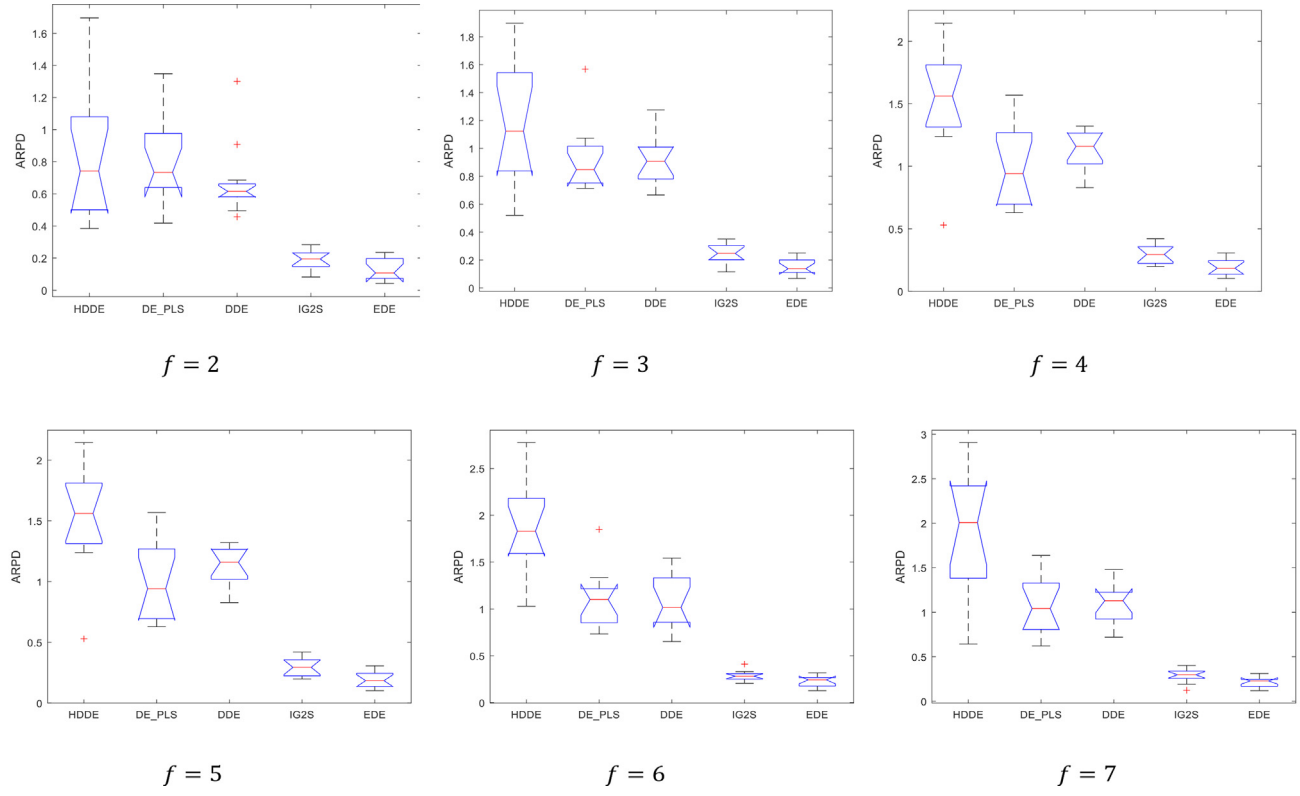


Fig. 9. Mean and 95% LSD interval for the test algorithms in $T = 15$.

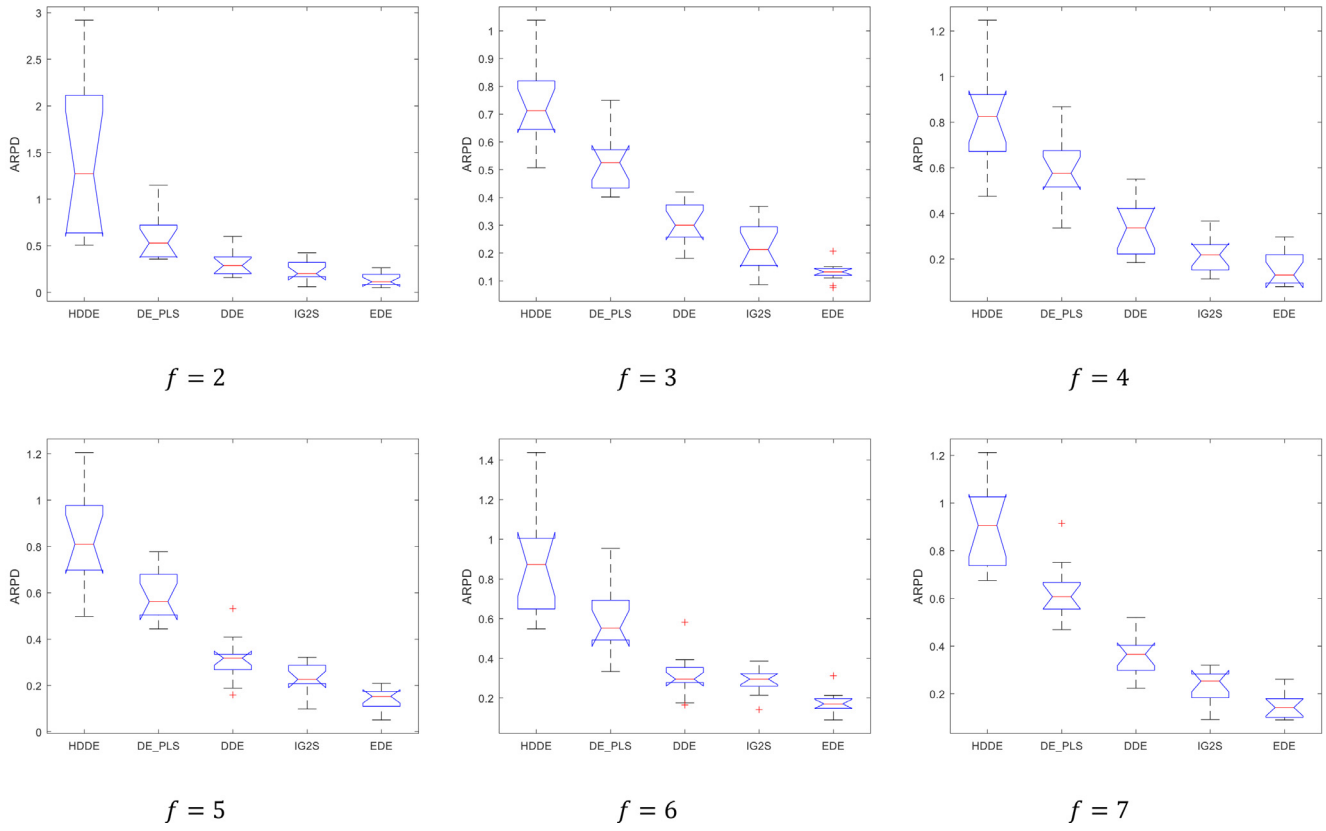


Fig. 10. Mean and 95% LSD interval for the test algorithms in $T = 30$.

The statistical results under different statistical indicators are different, different statistical methods were used to further verify the previous conclusions. The ANOVA test of the EDE algorithm and the IG2S algorithm in time-levels $T = 5, 15, 30$ are described in Figs. 17–19.

There are five main columns in Tables 11–13

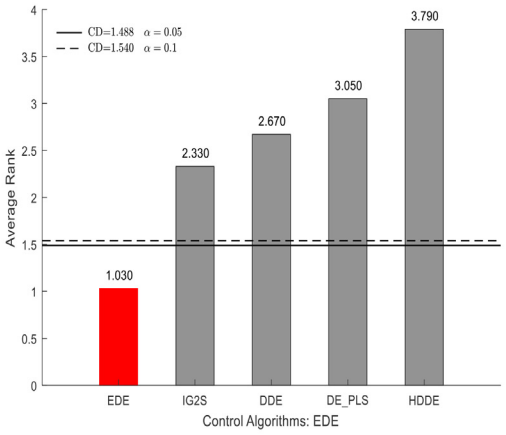
- The first shows the sum of squares (SS).
- The second shows the degrees of freedom (df).
- The third shows the mean squares, which is the ratio SS/df.
- The fourth shows the F statistic, which is the ratio of the MS's.

Table 10
Rankings obtained through Wilcoxon's test for the test algorithms by CPU time limit $T = 30$.

f	EDE VS	R^+	R^-	$R \approx$	$p - value$	$\alpha = 0.05$
2	HDDE	6527	48	625	0.000	**Yes**
	DE_PLS	6362	96	742	0.000	**Yes**
	DDE	5322	147	1731	0.037	**Yes**
	IG2S	5103	263	1834	0.041	**Yes**
3	HDDE	6899	40	261	0.000	**Yes**
	DE_PLS	6431	126	643	0.000	**Yes**
	DDE	5564	155	1481	0.030	**Yes**
	IG2S	5803	131	1266	0.028	**Yes**
4	HDDE	6863	37	300	0.000	**Yes**
	DE_PLS	6287	96	817	0.000	**Yes**
	DDE	5893	143	1164	0.035	**Yes**
	IG2S	6017	116	1067	0.017	**Yes**
5	HDDE	6931	30	239	0.000	**Yes**
	DE_PLS	6425	115	660	0.000	**Yes**
	DDE	5963	163	1074	0.028	**Yes**
	IG2S	6052	127	1021	0.019	**Yes**
6	HDDE	7003	32	165	0.000	**Yes**
	DE_PLS	6785	185	230	0.000	**Yes**
	DDE	6132	236	832	0.029	**Yes**
	IG2S	6253	89	858	0.020	**Yes**
7	HDDE	7049	41	110	0.000	**Yes**
	DE_PLS	6983	191	126	0.000	**Yes**
	DDE	6329	315	556	0.037	**Yes**
	IG2S	6719	276	205	0.023	**Yes**

Algorithm	Mean Rank
EDE	1.030
IG2S	2.330
DDE	2.670
DE_PLS	3.050
HDDE	3.790
Crit.Diff.α=0.05	1.488
Crit.Diff.α=0.10	1.540

(a)

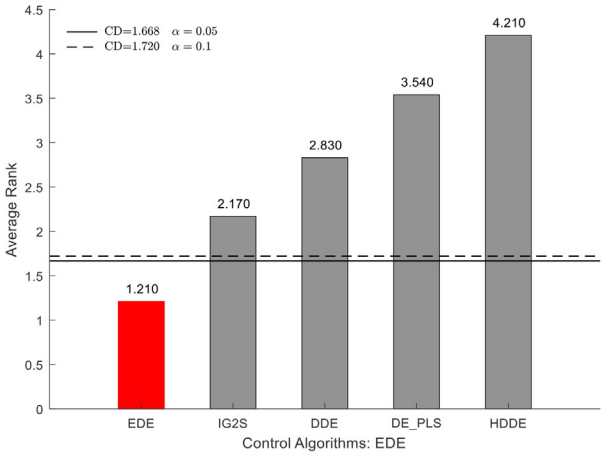


(b)

Fig. 11. Friedman test at time-level $T = 30$ and factory-level $f = 2$.

Algorithm	Mean Rank
EDE	1.210
IG2S	2.170
DDE	2.830
DE_PLS	3.540
HDDE	4.210
Crit.Diff.α=0.05	1.668
Crit.Diff.α=0.10	1.720

(a)

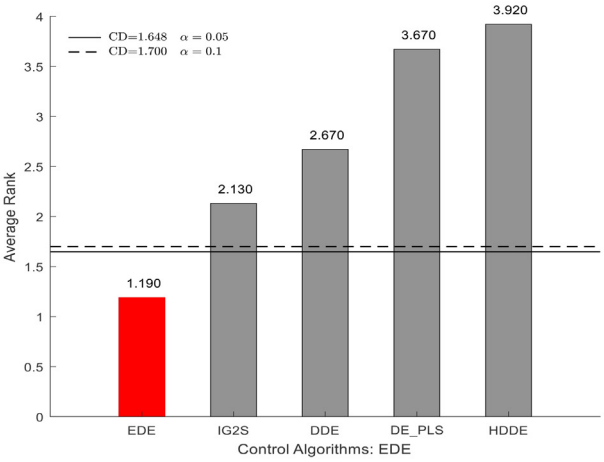


(b)

Fig. 12. Friedman test at time-level $T = 30$ and factory-level $f = 3$.

Algorithm	Mean Rank
EDE	1.190
IG2S	2.130
DDE	2.670
DE_PLS	3.670
HDDE	3.920
Crit.Diff.α=0.05	1.648
Crit.Diff.α=0.10	1.700

(a)

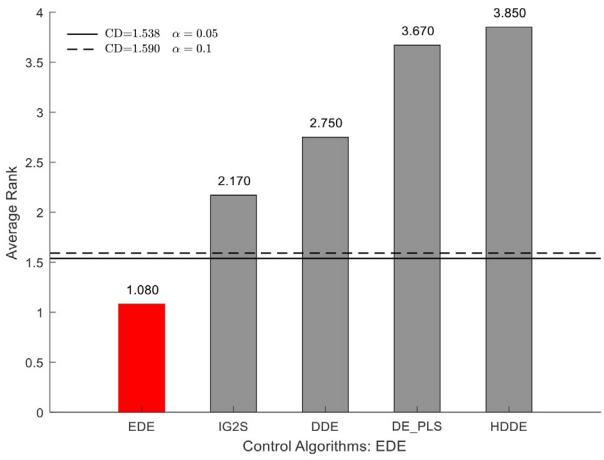


(b)

Fig. 13. Friedman test at time-level $T = 30$ and factory-level $f = 4$.

Algorithm	Mean Rank
EDE	1.080
IG2S	2.170
DDE	2.750
DE_PLS	3.670
HDDE	3.850
Crit.Diff.α=0.05	1.538
Crit.Diff.α=0.10	1.590

(a)

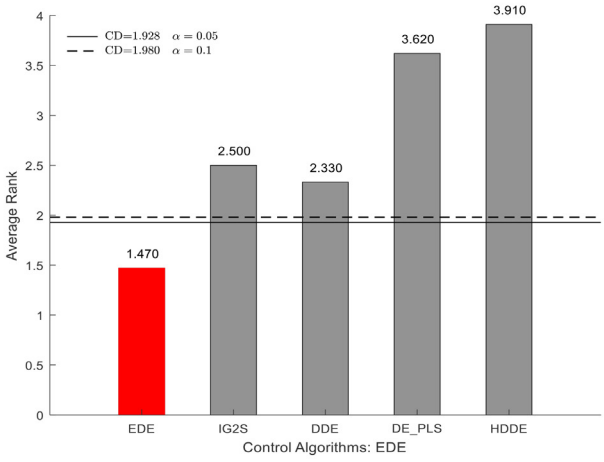


(b)

Fig. 14. Friedman test at time-level $T = 30$ and factory-level $f = 5$.

Algorithm	Mean Rank
EDE	1.470
IG2S	2.500
DDE	2.330
DE_PLS	3.620
HDDE	3.910
Crit.Diff.α=0.05	1.928
Crit.Diff.α=0.10	1.980

(a)

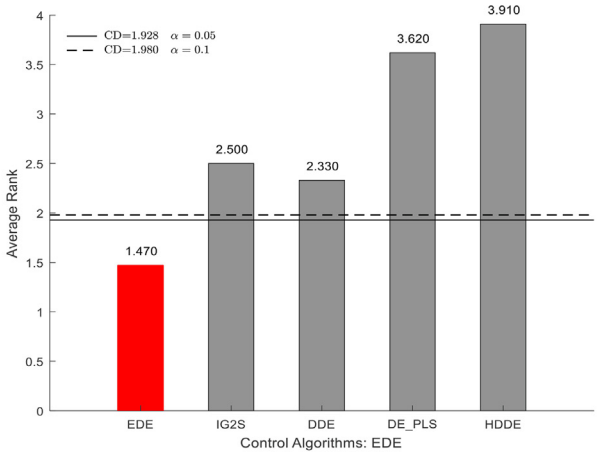


(b)

Fig. 15. Friedman test at time-level $T = 30$ and factory-level $f = 6$.

Algorithm	Mean Rank
EDE	1.470
IG2S	2.500
DDE	2.330
DE_PLS	3.620
HDDE	3.910
Crit.Diff.α=0.05	1.928
Crit.Diff.α=0.10	1.980

(a)



(b)

Fig. 16. Friedman test at time-level $T = 30$ and factory-level $f = 7$.

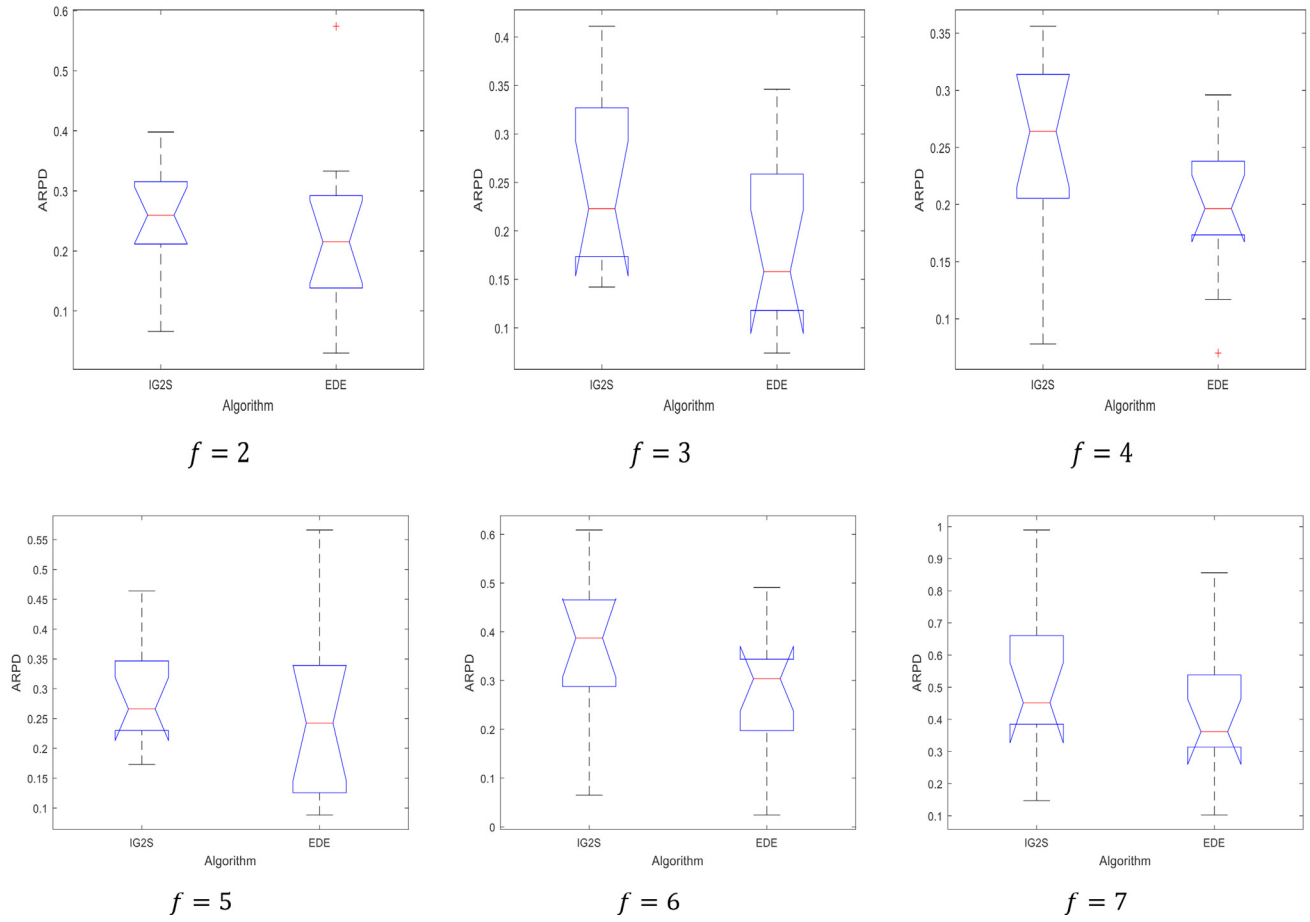


Fig. 17. The box plots of EDE and IG2S in $T = 5$.

- The fifth shows the p-value, which is derived from the cumulative distribution function of F. As F increase, the p-value decrease.

If the p-value is near zero, this casts doubt on the null hypothesis and suggests that at least one sample mean is significantly different than the other sample means. It is declaring a result significant if the p-value is less than 0.05 or 0.01.

From the experimental series and analysis, the results of the datum and figures showed that the EDE algorithm was the desirable algorithm on large-scale instances.

4.6. The effect of problem variables on the algorithm

In this section, the impact of problem variables on algorithm performance is investigated. Fig. 17 shows the interaction between the algorithmic performance and the number of machines and jobs, and illustrates the influences of problem scale parameters on the algorithmic performance. Table 14 shows the ARPD results for the compared algorithms in different machine-level.

As shown in Fig. 20, with the increasing of machines m and jobs n, the EDE algorithm is still superior to the compared algorithm. The reason is that HDDE, DE_PLS, CED and EDE are population-based algorithms that require enough CPU time for mutation and crossover operation to produce a satisfactory solution. Meanwhile, the complexity of production process is significantly expanded as the increasing of the m, n and f.

However, the HDDE and DE_PLS algorithms starts from a random initial solution and hard to converge to a satisfactory solution in a limited CPU time. However, EDE and CED algorithms have different heuristics methods to produce a desirable initial solution and converge to a satisfactory solution in less CPU time.

As the above experimental series and analysis show, the proposed EDE algorithm is an effective algorithm for solving the DBFSP through minimizing the makespan criterion. The reasons are concluded as follows. First, two different heuristics methods and one random strategy are used to initialize the population, and an ensembled mutation strategy is used in the mutation operation, which enhanced the diversity of the population. At the same time, the characteristics of the distributed blocking flowshop problem are considered in two different heuristics methods. Second, the EDE algorithm is a population-based algorithm. Although the population-based algorithms are sensitive to the algorithm parameters, such as the mutation factor, crossover factor or population size, they generally provide excellent results on large-scale problems (Zhao, Qin, Zhang, et al., 2019). In this paper, the parameter combinations of the EDE algorithm were analyzed by DOE method to select the best combination of key parameters. Third, different from traditional DE, the greedy selection operator is replaced by the biased selection factor, and the elitism strategy is introduced at the same time, to ensure the select pressure in excellent individuals. Finally, according to no-free-lunch (NFL) theorems, there is no single algorithm suitable for all potential optimization problems (Wolpert & Macready, 1997). Therefore, the ensembled method, which is supposed to take advantage of previous knowledge of DE and its variants to create a new variant, is an effective method to design high-quality DE algorithms.

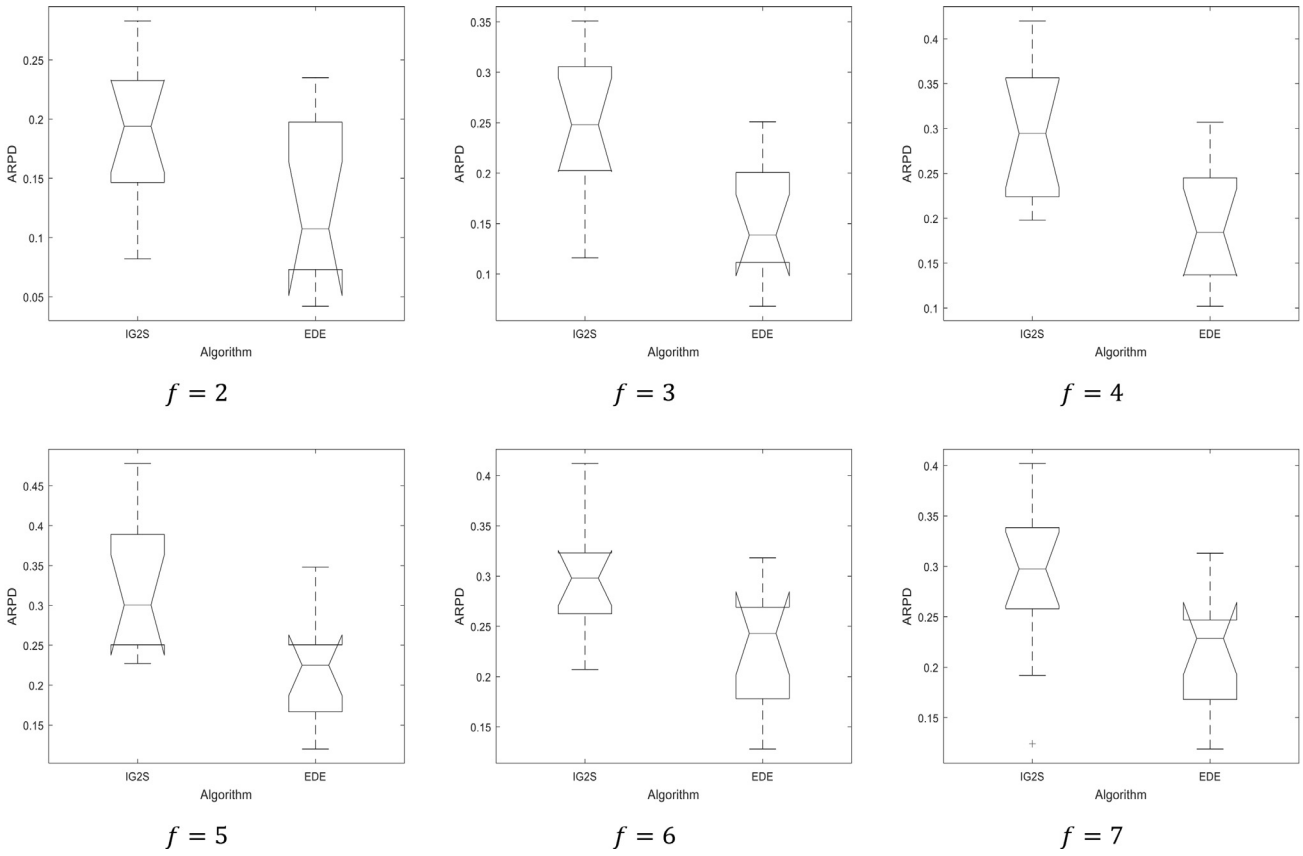


Fig. 18. The box plots of EDE and IG2S in $T = 15$.

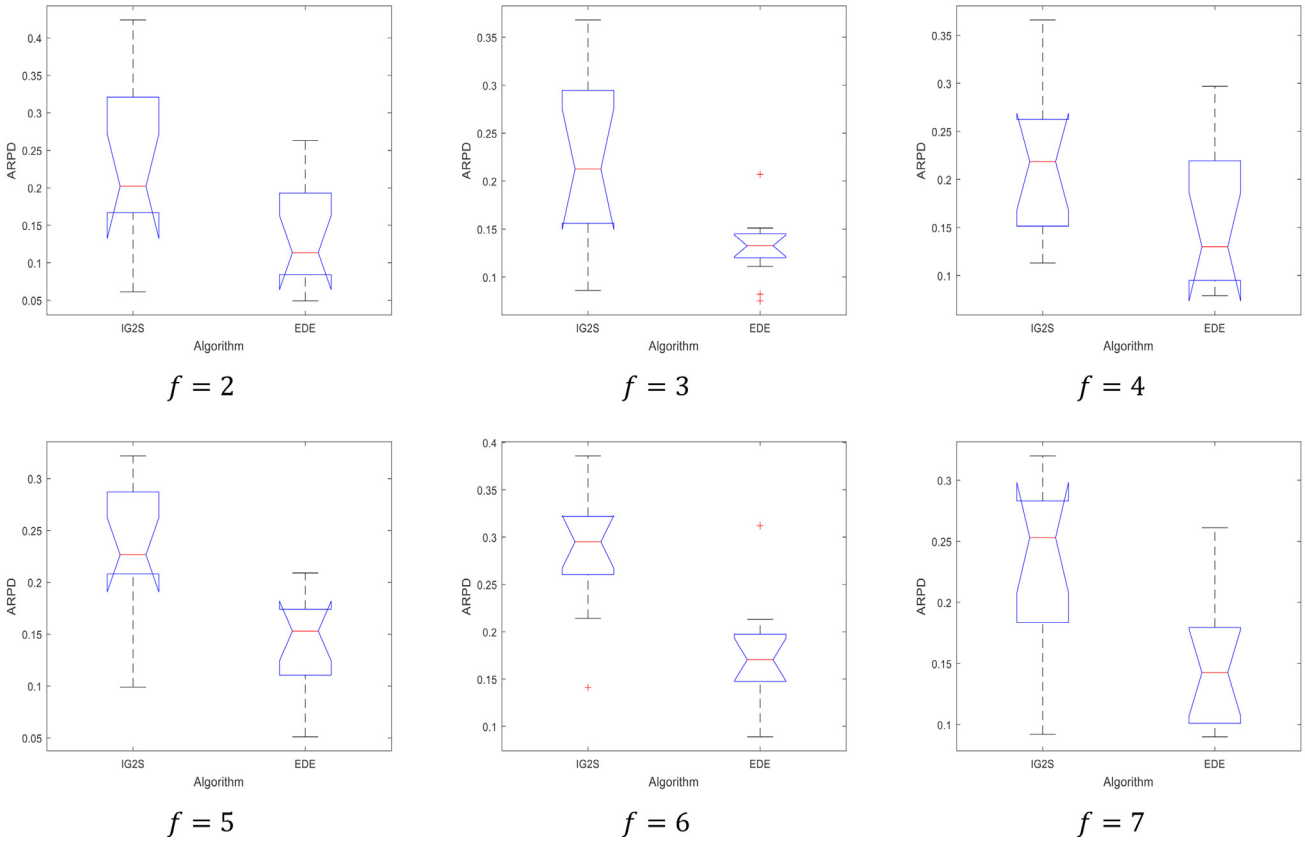


Fig. 19. The box plots of EDE and IG2S in $T = 30$.

Table 11
The One-way Analysis of Variance (ANOVA) table of EDE and IG2S in time-level $T = 5$.

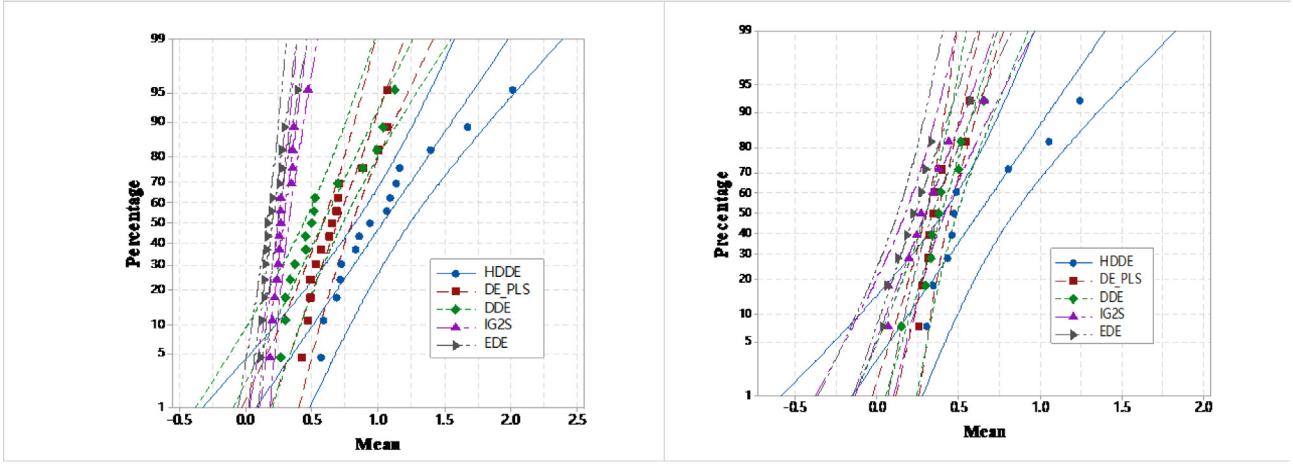
f	EDE vs	Total SS	Total df	MS	F	p-value	$\alpha = 0.05$	$\alpha = 0.10$
2	IG2S	0.34072	23	0.00213	0.14	0.7136	NO	NO
3	IG2S	0.19404	23	0.2561	3.35	0.081	NO	**Yes**
4	IG2S	0.13557	23	0.2001	3.81	0.0638	NO	**Yes**
5	IG2S	0.37196	23	0.00338	0.2	0.6575	NO	NO
6	IG2S	0.47771	23	0.05861	3.08	0.0934	NO	**Yes**
7	IG2S	1.26319	23	0.05501	1.01	0.3278	NO	NO

Table 12
The One-way Analysis of Variance (ANOVA) table of EDE and IG2S in time-level $T = 15$.

f	EDE vs	Total SS	Total df	MS	F	p-value	$\alpha = 0.05$	$\alpha = 0.10$
2	IG2S	0.11108	23	0.02089	5.09	0.0343	**Yes**	**Yes**
3	IG2S	0.14629	23	0.05482	13.18	0.0015	**Yes**	**Yes**
4	IG2S	0.17882	23	0.06955	14	0.0011	**Yes**	**Yes**
5	IG2S	0.18335	23	0.06376	11.73	0.0024	**Yes**	**Yes**
6	IG2S	0.1037	23	0.02912	8.59	0.0077	**Yes**	**Yes**
7	IG2S	0.14089	23	0.03635	7.65	0.0113	**Yes**	**Yes**

Table 13
The One-way Analysis of Variance (ANOVA) table of EDE and IG2S in time-level $T = 30$.

f	EDE vs	Total SS	Total df	MS	F	p-value	$\alpha = 0.05$	$\alpha = 0.10$
2	IG2S	0.24122	23	0.05665	6.75	0.0164	**Yes**	**Yes**
3	IG2S	0.13868	23	0.00393	13.27	0.0014	**Yes**	**Yes**
4	IG2S	0.14979	23	0.2227	3.84	0.0628	No	**Yes**
5	IG2S	0.11701	23	0.04683	14.44	0.001	**Yes**	**Yes**
6	IG2S	0.15855	23	0.07616	20.34	0.0002	**Yes**	**Yes**
7	IG2S	0.12659	23	0.04735	13.15	0.0015	**Yes**	**Yes**



(a) Jobs n (b) Machines m

Fig. 20. Probability plot of the compared algorithms.

Table 14
The ARPD results for the compared algorithm in different job *n*.

T	Machine	HDDE	DE_PLS	DDE	IG2S	EDE
5	20	1.138	0.471	0.515	0.262	**0.187**
	50	0.934	0.704	0.526	0.354	**0.275**
	100	0.860	0.649	0.501	0.366	**0.288**
	200	0.689	0.532	0.453	0.356	**0.276**
	500	0.586	0.486	0.452	0.471	**0.390**
15	20	2.018	1.072	1.125	0.340	**0.254**
	50	1.673	1.072	1.034	0.265	**0.202**
	100	1.393	0.996	0.988	0.259	**0.163**
	200	1.090	0.882	0.883	0.234	**0.141**
	500	0.725	0.683	0.695	0.200	**0.117**
30	20	1.166	0.696	0.369	0.22	**0.137**
	50	1.060	0.630	0.338	0.252	**0.163**
	100	0.827	0.568	0.297	0.250	**0.148**
	200	0.713	0.492	0.301	0.247	**0.152**
	500	0.571	0.423	0.267	0.181	**0.112**

5. Conclusions and future research

In this paper, an ensemble different evolution (EDE) algorithm is presented to solve the distributed blocking flowshop scheduling problem with the objective of minimizing makespan. First, a large number of experimental results show that the hidden knowledge from knowledge specific problem is used to design efficient scheduling rules by constructing the initial solution, which is significantly better than those methods that do not use domain knowledge. Second, the integration of different mutation strategies has a significant effect and plays an important role in EDE, which achieve the effect of ' 1 + 1 > 2 ' instead of simple combination. Finally, the experimental results compared with the state-of-arts show that EDE is significantly better than the comparison algorithm. Therefore, EDE is an effective method to solve the DBFSP.

Although the optimization effect of EDE is significant compared with the performance of classical algorithm, a large number of practical application cases are still needed to test the effectiveness of the algorithm. For future work, it is necessary to further implement sensitivity analysis for algorithm parameters and to research the mutation strategy selection. Several directions are suggested. First, the proposed EDE algorithm is applied to other distributed scheduling problems or other optimization problems, such as the distributed no-idle flowshop scheduling problems and traveling salesman problem. Second, different constraints, such as energy constraints and limit buffer, are considered to be appended for the scheduling problem. Finally, it is desirable to add an adaptive mechanism for parameter adjustment.

CRediT authorship contribution statement

Fuqing Zhao: Conceptualization, Funding acquisition, Supervision, Writing - review & editing. **Lexi Zhao:** Data curation, Formal analysis, Writing - original draft. **Ling Wang:** Methodology, Resources. **Houbin Song:** Investigation, Resources, Software, Validation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was financially supported by the National Key Research and Development Plan under grant number 2018YFB1703105 and National Natural Science Foundation of China under grant numbers 61663023 and 61873328, China. It is also supported by the Key Research Programs of Science and Technology Commission Foundation of Gansu Province (2017GS10817), Lanzhou Science Bureau project (2018-rc-98), Public Welfare Project of Zhejiang Natural Science Foundation (LGJ19E050001) and Wenzhou Public Welfare Science and Technology project (G20170016), respectively.

References

Baioletti, M., Milani, A., & Santucci, V. (2020). Variable neighborhood algebraic Differential Evolution: An application to the Linear Ordering Problem with Cumulative Costs. *Information Sciences, 507*, 37–52.
Behnamian, J., & Ghomi, S. M. T. F. (2016). A survey of multi-factory scheduling. *Journal of Intelligent Manufacturing, 27*, 231–249.
Brest, J., Maučec, M. S., & Bošković, B. (2017). Single objective real-parameter optimization: Algorithm jSO. In 2017 IEEE Congress on Evolutionary Computation (CEC) (pp. 1311-1318): IEEE.
Fernandez-Viagas, V., & Framinan, Jose M. (2015). A bounded-search iterated greedy algorithm for the distributed permutation flowshop scheduling problem. *International Journal of Production Research, 53*, 1111–1123.
Fernandez-Viagas, V., Perez-Gonzalez, P., & Framinan, Jose M. (2018). The distributed permutation flow shop to minimise the total flowtime. *Computers and Industrial Engineering, 118*, 464–477.
Gong, H., Tang, L. X., & Duin, C. W. (2010). A two-stage flow shop scheduling problem on a batching machine and a discrete machine with blocking and shared setup times. *Computers and Operations Research, 37*, 960–969.
Graham, R. L., Lawler, E. L., Lenstra, J. K., & KanRinnooy, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics, 5*, 287–326.
Hall, N. G., & Sriskandarajah, C. (1996). A survey of machine scheduling problems with blocking and no-wait in process. *Operations Research, 44*, 510–525.
He, D. W., Kusiak, A., & Artiba, A. (1996). A scheduling problem in glass manufacturing. *AIIE Transactions, 28*, 129–139.
Li, X. P., Yang, Z., Ruiz, R., Chen, T., & Sui, S. C. (2018). An iterated greedy heuristic for no-wait flow shops with sequence dependent setup times, learning and forgetting effects. *Information Sciences, 453*, 408–425.
Lin, S. W., & Ying, K. C. (2013). Minimizing makespan in a blocking flowshop using a revised artificial immune system algorithm. *Omega, 41*, 383–389.
Lin, S. W., & Ying, K. C. (2016). Minimizing makespan for solving the distributed no-wait flowshop scheduling problem. *Computers and Industrial Engineering, 99*, 202–209.
Maccarthy, B. L., & Liu, J. Y. (1993). Addressing the gap in scheduling research: A review of optimization and heuristic methods in production scheduling. *International Journal of Production Research, 31*, 59–79.
McCormick, S. T., Pinedo, M. L., Shenker, S., & Wolf, B. (1989). Sequencing in an Assembly Line with Blocking to Minimize Cycle Time. *Operations Research, 37*, 925–935.
Merchan, A. F., & Maravelias, C. T. (2016). Preprocessing and tightening methods for time-indexed MIP chemical production scheduling models. *Computers and Chemical Engineering, 84*, 516–535.
Montgomery, D. C. (2017). *Design and analysis of experiments*. John Wiley & Sons.
Naderi, B., & Ruiz, R. (2010). The distributed permutation flowshop scheduling problem. *Computers and Operations Research, 37*, 754–768.
Naderi, B., & Ruiz, R. (2014). A scatter search algorithm for the distributed permutation flowshop scheduling problem. *European Journal of Operational Research, 239*, 323–334.
Nawaz, M., Enscore, J., Emory, E., & Ham, I. (1983). A heuristic algorithm for the m -machine, n -job flow-shop sequencing problem. *Omega, 11*, 91–95.
Nayak, S. K., Rout, P. K., Jagadev, A. K., & Swarnkar, T. (2017). Elitism based Multi-Objective Differential Evolution for feature selection: A filter approach with an efficient redundancy measure. *Journal of King Saud University – Computer and Information Sciences, 2017*.
Pan, Q. K., Gao, L., Wang, L., Liang, J., & Li, X. Y. (2019). Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem. *Expert Systems with Applications, 124*, 309–324.
Pan, Q. K., Gao, L., Li, X. Y., & Jose, F. M. (2019). Effective constructive heuristics and meta-heuristics for the distributed assembly permutation flowshop scheduling problem. *Applied Soft Computing, 124*, 309–324.
Pan, Q. K., & Wang, L. (2011). Effective heuristics for the blocking flowshop scheduling problem with makespan minimization. *Omega, 40*, 218–229.
Price, K. V. (1996). Differential evolution: a fast and simple numerical optimizer. In Proceedings of North American Fuzzy Information Processing (524–527).
Ribas, I., Companys, R., & Tort-Martorell, X. (2015). An efficient Discrete Artificial Bee Colony algorithm for the blocking flow shop problem with total flowtime minimization. *Expert Systems with Applications, 42*, 6155–6167.
Ribas, I., Companys, R., & Tort-Martorell, X. (2017). Efficient heuristics for the parallel blocking flow shop scheduling problem. *Expert Systems with Applications, 74*, 41–54.
Ribas, I., Companys, R., & Tort-Martorell, X. (2019). An Iterated Greedy Algorithm for Solving the Total Tardiness Parallel Blocking Flow Shop Scheduling Problem. *Expert Systems with Applications, 121*, 347–361.

Ronconi, D. P. (2004). A note on constructive heuristics for the flowshop problem with blocking. *International Journal of Production Economics, 87*, 39–48.

Ronconi, D. P., & Armentano, V. A. (2001). Lower bounding schemes for flowshops with blocking in-process. *Journal of the Operational Research Society, 52*, 1289–1297.

Ruiz, R., Pan, Q. K., & Naderi, B. (2019). Iterated Greedy methods for the distributed permutation flowshop scheduling problem. *Omega, 83*, 213–222.

Shao, Z. S., Pi, D. C., & Shao, W. S. (2019). A novel multi-objective discrete water wave optimization for solving multi-objective blocking flow-shop scheduling problem. *Knowledge-Based Systems, 165*, 110–131.

Shao, Z. S., Pi, D. C., & Shao, W. S. (2020). Hybrid enhanced discrete fruit fly optimization algorithm for scheduling blocking flow-shop in distributed environment. *Expert Systems with Applications, 145*, 113–147.

Shao, Z. S., Pi, D. C., Shao, W. S., & Yuan, P. S. (2019). An efficient discrete invasive weed optimization for blocking flow-shop scheduling problem. *Engineering Applications of Artificial Intelligence, 78*, 124–141.

Taillard (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research, 64*, 278–285.

Tasgetiren, M. F., Pan, Q. K., Kizilay, D., & Suer, G. (2015). A populated local search with differential evolution for blocking flowshop scheduling problem. In 2015 IEEE Congress on Evolutionary Computation (CEC) (2789–2796).

Vallada, E., Ruiz, R., & Framinan, Jose M. (2015). New hard benchmark for flowshop scheduling problems minimising makespan. *European Journal of Operational Research, 240*, 666–677.

Wang, J. J., & Wang, L. (2018). A knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop. *IEEE Transactions on Systems Man and Cybernetics Systems*, 1–15.

Wang, L., Pan, Q. K., Suganthan, P. N., Wang, W. H., & Wang, Y. M. (2010). A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems. *Computers and Operations Research, 37*, 509–520.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*, 67–82.

Ying, K. C., Lin, S. W., Cheng, C. Y., & He, C. D. (2017). Iterated reference greedy algorithm for solving distributed no-idle permutation flowshop scheduling problems. *Computers & Industrial Engineering, 110*, 413–423.

Zamuda, A., & Sosa, J. D. H. (2019). Success history applied to expert system for underwater glider path planning using differential evolution. *Expert Systems with Applications, 119*, 155–170.

Zhang, G. H., & Xing, K. Y. (2018). Memetic social spider optimization algorithm for scheduling two-stage assembly flowshop in a distributed environment. *Computers and Industrial Engineering, 125*, 423–433.

Zhang, G. H., & Xing, K. Y. (2019). Differential evolution metaheuristics for distributed limited-buffer flowshop scheduling with makespan criterion. *Computers and Operations Research, 108*, 33–43.

Zhang, G. H., Xing, K. Y., & Cao, F. (2018). Discrete differential evolution algorithm for distributed blocking flowshop scheduling with makespan criterion. *Engineering Applications of Artificial Intelligence, 76*, 96–107.

Zhao, F. Q., Liu, H., Zhang, Y., Ma, W. M., & Zhang, C. (2018). A discrete Water Wave Optimization algorithm for no-wait flow shop scheduling problem. *Expert Systems with Applications, 91*, 347–363.

Zhao, F. Q., Liu, Y., Zhang, Y., Ma, W. M., & Zhang, C. (2017). A hybrid harmony search algorithm with efficient job sequence scheme and variable neighborhood search for the permutation flow shop scheduling problems. *Engineering Applications of Artificial Intelligence, 65*, 178–199.

Zhao, F. Q., Qin, S., Yang, G. Q., Ma, W. M., Zhang, C., & Song, H. B. (2019). A factorial based particle swarm optimization with a population adaptation mechanism for the no-wait flow shop scheduling problem with the makespan objective. *Expert Systems with Applications, 126*, 41–53.

Zhao, F. Q., Qin, S., Zhang, Y., Ma, W. M., Zhang, C., & Song, H. B. (2019). A hybrid biogeography-based optimization with variable neighborhood search mechanism for no-wait flow shop scheduling problem. *Expert Systems with Applications, 126*, 321–339.

Zhao, F. Q., Xue, F. L., Zhang, Y., Ma, W. M., Zhang, C., & Song, H. B. (2018). A hybrid algorithm based on self-adaptive gravitational search algorithm and differential evolution. *Expert Systems with Applications, 113*, 515–530.