Complex & Intelligent Systems (2022) 8:141–161
https://doi.org/10.1007/s40747-021-00354-5

ORIGINAL ARTICLE

A memetic discrete differential evolution algorithm for the distributed permutation flow shop scheduling problem

Fuqing Zhao[1] · Xiaotong Hu[1] · Ling Wang[2] · Zekai Li[1]

Received: 24 September 2020 / Accepted: 25 March 2021 / Published online: 10 April 2021

Abstract
The distributed manufacturing has become a prevail production mode under the economic globalization. In this article, a memetic discrete differential evolution (MDDE) algorithm is proposed to address the distributed permutation flow shop scheduling problem (DPFSP) with the minimization of the makespan. An enhanced NEH (Nawaz–Enscore–Ham) method is presented to produce potential candidate solutions and Taillard's acceleration method is adopted to ameliorate the operational efficiency of the MDDE. A new discrete mutation strategy is introduced to promote the search efficiency of the MDDE. Four neighborhood structures, which are based on job sequence and factory assignment adjustment mechanisms, are introduced to prevent the candidates from falling the local optimum during the search process. A neighborhood search mechanism is selected adaptively through a knowledge-based strategy which focuses on the adaptive evaluation for the neighborhood selection. The optimal combinations of parameters in the MDDE algorithm are testified by the design of experiment. The computational results and comparisons demonstrated the effectiveness of the MDDE algorithm for solving the DPFSP.

Keywords Distributed permutation flow shop scheduling · Neighborhood structures · Differential evolution · Knowledge · Makespan

Introduction

In the context of globalization, intelligent manufacturing has become a common production mode. With the increasing cooperation between factories, distributed manufacturing systems have been widely applied [1,2]. The distributed scheduling contributes to reduce the management risk and improve the efficiency of transportation [3,4]. Among various types of scheduling problems, the flow shop scheduling problem has attracted widespread attention from researchers [5,6]. The permutation flow shop scheduling problem (PFSP) is one of the classic combinatorial optimization problems, which has been proved as an NP-hard problem [7,8]. Due to its importance in engineering applications and academic, certain methods have been proposed to solve the PFSP [9,10].

The distributed permutation flow shop scheduling problem (DPFSP) is a generalization of classical permutation flow shop scheduling problem, which is used to fill the gap between practical application and academic research [5,11], as shown in Fig. 1. The DPFSP is expressed as DP/prmu/C_{max} , where DP represents the distributed factory; prmu represents permutation features, and C_{max} represents the target is to minimize the maximum completion time. In the DPFSP, there are F factories with the same m machines. In addition to the constraints of PFSP, DPFSP also stipulates that once a job is assigned to a factory, it is not allowed to be re-distributed. The processing times of the jobs do not differ between factories [12,13]. This additional dimension minimizes the maximum global makespan of F factories to determine the jobs that should be assigned to each factory. Therefore, from the specific situation of $F = 1$ in [14], DPFSP is a typical PFSP problem. For DPFSP, several integer linear programming models and two

✉ Fuqing Zhao
Fzhao2000@hotmail.com

Xiaotong Hu
huxt826@163.com

Ling Wang
wangling@tsinghua.edu.cn

Zekai Li
410379065@qq.com

[1] School of Computer and Communication Technology, Lanzhou University of Technology, Lanzhou 730050, China

[2] Department of Automation, Tsinghua University, Beijing 10084, China

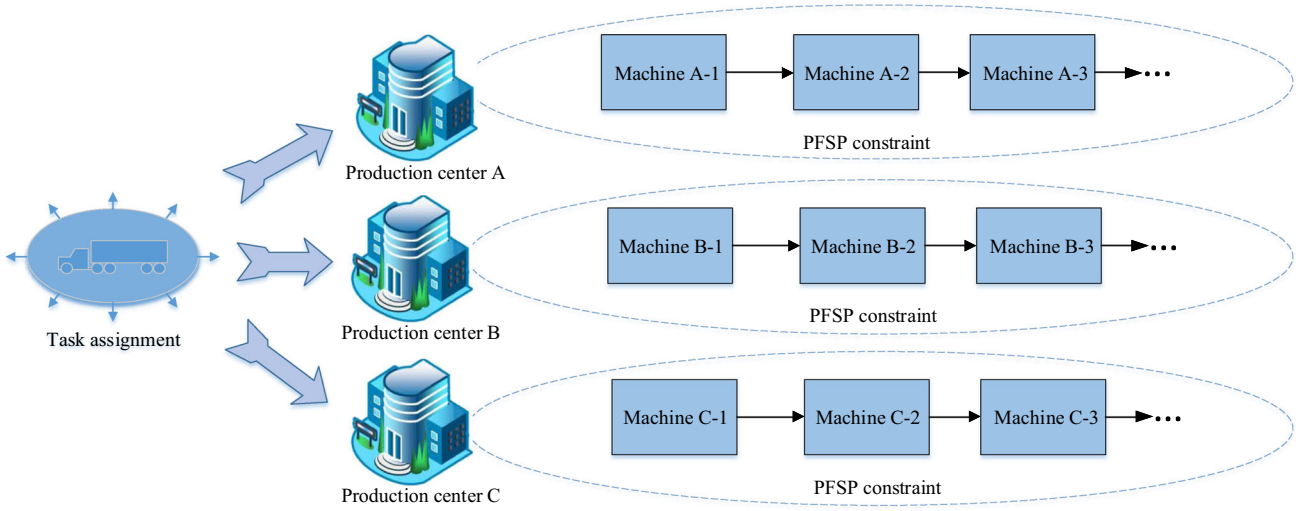

Fig. 1 The production model in distributed environment

factory assignment methods based on dispatching rules were proposed by Naderi [11]. Recently, several meta-heuristics have been developed to solve the DPFSP, including Iterated Greedy Algorithm (IG) [15,16], Estimation of Distributed Algorithm (EDA) [1], Discrete Artificial Bee Colony (ABC) and Iterated Local Search Algorithm (ILS) [13]. Differential Evolution Algorithms have rarely been studied to solve the DPFSP.

Differential evolution (DE) algorithm [17] is an intelligent optimization algorithm, including three critical components, mutation, crossover, and selection. In DE, three operations are repeated after population initialization until the termination condition is satisfied. The convergence result of DE is mainly determined by the mutation operation and the crossover operation [18,19]. To improve the performance of DE, researchers have proposed a series of improvement measures from the aspects of mutation strategy, population size, parameter design and fusion of other algorithms.

In terms of mutation strategy, Zhang and Sanderson [20] proposed a new mutation strategy "DE/current-to-pbest " to update the scale factor in an adaptive way to enhance the optimization performance (JADE). The jSO [21] is a variant of iL-SHADE algorithm [22] and it is the winner algorithm in the CEC2017 benchmark. In the jSO, a mutation strategy named "DE/current-to-pBest-w/1" is proposed. For the size of the population, Tanabe and Fukunaga proposed an adaptive algorithm (LSHADE) [23], which is based on linear population size reduction (LPSR). In LSHADE, the LPSR rule is applied to save expensive evaluations. For the parameters, Mohamed, Hadi, Fattouh and Jambi proposed LSHADE-SPACMA [24], which adopted a semi-parametric adaptive method based on randomization and self-adaptation. Furthermore, a hybridization framework is introduced between the modified versions of LSHADE-SPA and the Covariance Matrix Adaptation Evolution Strategy (CMA-ES). In recent years, DE has been fused with other algorithms to improve its performance. The search mechanism of DE is introduced into the Biogeography-Based Optimization algorithm (BBO) to improve the anti-rotation of the algorithm (TDBBO) [25]. The simulation results of TDBBO based on the CEC2017 benchmark test suit show that TDBBO has better accuracy and convergence speed than the most advanced BBO. In the Gravitational Search Algorithm (GSA), the crossover and mutation operations of DE are used to realize the diversity of population, which is named SGSADE [26].

DE and its variant algorithms have been widely used in production scheduling problems. The IDESAA algorithm which was integrated with the simulated annealing (SA) and DE was proposed by Xiuli and Xiajing [27] to solve the distributed flexible job-shop scheduling problem. A hybrid DE algorithm based on set model (eEDA) with distribution estimation algorithm is proposed by Zhou et al. [28] to solve the scheduling problem of reentrant hybrid flow shop. A discrete differential evolution (DDE) algorithm was designed by Zhang et al. [29] to solve the distributed blocking flow-shop scheduling. In the following research work, the author designed a CDE algorithm based on DDE to solve the distributed limited-buffer flow shop scheduling problem [30]. Wang et al. [31] proposed a novel hybrid discrete differential evolution (HDDE) algorithm for solving the blocking flow-shop scheduling problem. In HDDE, the permutations between jobs are discrete, thus the algorithm is directly operated in the discrete domain. Furthermore, a local search strategy based on the inserted neighborhood structure is introduced into the algorithm to enhance the ability of local search.

Inspired by the framework of MDDE, the MDDE is proposed to address the DPFSP. The search operator of the

general heuristic is randomly searched during the prime search process, which leads to the phenomenon of undesirable search efficiency. In general, the effectiveness in the search process is promoted by the previously acquired prior knowledge. An effective method of decomposition and coordination is formed by cooperative optimization. First, a constructive method named DNEH+T is used to create initial population for the MDDE. Second, a discrete mutation strategy is proposed to generate new solutions. Finally, four neighborhood search mechanisms and local search strategies are embedded in the MDDE to further improve the quality of solutions. Meanwhile, an appropriate neighborhood search mechanism is adaptively selected by a knowledge-based optimization strategy and is applied to promote the global search capacity of the MDDE.

The effectiveness of the four integration strategies in the MDDE are demonstrated by the experimental results. The contributions in this study are summarized as follows.

- An improved NEH method is designed to generate potential candidate solution and the Taillard's acceleration method is adopted to ameliorate the efficiency of the insertion process for jobs.
- A new discrete mutation strategy is proposed for MDDE algorithm to ameliorate the exploration ability. In the discrete mutation strategy, the best solution information is used to balance the diversity of population and the greediness of mutation.
- Four neighborhood structures based on job sequence and factory assignment adjustment mechanisms are designed to enhanced the search around the current best candidate solution. Furthermore, an appropriate neighborhood search mechanism is adaptively selected by a knowledge-based optimization strategy.

The structure of this study is as follows. The definition and details of DPFSP and the traditional DE are introduced in "Background". The proposed MDDE algorithm are introduced in "MDDE for DPFSP with minimizing makespan". The related experimental analysis and discussion are in "The experiments and comparisons". The conclusion and future works are suggested in "Conclusions and future research".

Background

The distributed permutation flow shop scheduling problem

According to the literature [32], the DPFSP is described as follows. A set of jobs J is processed in F factories, where $J = j_1, j_2, \ldots, j_n$ and $F = f_1, f_2, \ldots, f_l$. Each factory includes an identical flow shop with M machines, $M = m_1, m_2, \ldots, m_m$. In each factory, the operation of jobs j on machine i is denoted as O_{ij}. One job is processed on the only one machine at a time and each job has to go through all the machines in the factory to which it is assigned. In addition, each operation is not interrupted once it started. The setting time of the machine and the transmission time between operations are not considered. The notations in this study are shown as follows.

i	Index for machines where $i = 1, 2, \ldots, m$
ω	The scaling factor
j	Index for jobs where $j = 1, 2, \ldots, n$
F	The number of factories
f	Index for factories where $f = 1, 2, \ldots, l$
Cr	Crossover rate
n	The number of jobs
NP	The population size
m	The number of machines
P_{l}	Neighborhood search random value
$P_{j,i}$	The processing time of operation $O_(j, i)$ on machine M_i
Avg	The average
$I_{i,k,f}$	The idle time
$W_{i,k,f}$	Indicates the completion time of the job in position k on machine i in factory f
$X_{j,k,f}$	Binary variable that takes values 1 if job j occupies position k in factory f , and 0 otherwise

In this study, the purpose is to minimize the maximum completion time (completion time) with the sequence of jobs in the assigned factory. DPFSP detailed mixed-integer programming model is shown in the literature of Naderi and Ruiz [11]. Therefore, the maximum completion time minimization is calculated as follows.

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

$$\sum_{k=1}^{n}\sum_{f=1}^{F} X_(j,k,f) = 1, \forall_j \tag{2}$$

$$\sum_{j=1}^{n}\sum_{f=1}^{F} X_(j,k,f) = 1, \forall_k \tag{3}$$

$$I_{i,k,f} + \sum_{j=1}^{n} \cdot P_{j,i} + W_{i,k+1,f} - W_{i,k,f} - \sum_{j=1}^{n} X_(j,k,f) \cdot P_{j,i+1} - I_{i+1,k,f} = 0, \forall_{k<n, i<m, f} \tag{4}$$

$$C_f = \sum_{i=1}^{m-1}\sum_{j=1}^{n} X_(j,1,f) \cdot P_{j,i} + \sum_{k=1}^{n-1} I_(m,k,f) + \sum_{j=1}^{n}\sum_{k=1}^{n} X_(j,k,f) \cdot P_{j,m}, \forall_f \tag{5}$$

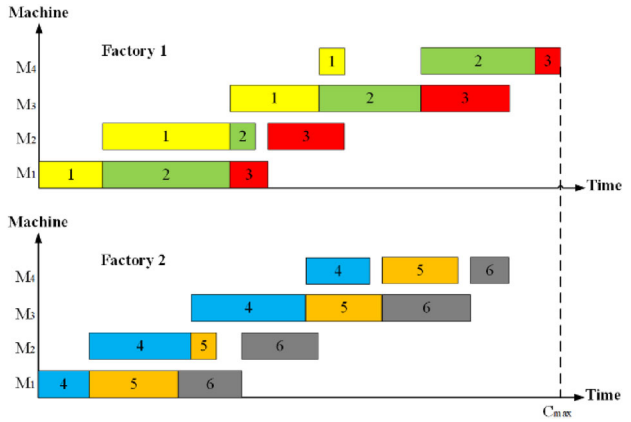

Fig. 2 The example of DPFSP

$$C_{max} \geq C_f, \forall_f \tag{6}$$
$$I_{i,k,f} \geq 0, \forall_{i,k<n,f} \tag{7}$$
$$W_{i,k,f} \geq 0, \forall_{k,i<m,f} \tag{8}$$
$$X_{j,k,f} \in \{0, 1\}, \forall_{j,k,f} \tag{9}$$

The constraint set (2) ensures that each job is assigned to only one location in a factory. All n positions that must occupy nF are indicated in the constraint set (3). The constraint set (4) is expressed as the idle time of the machine i after the job execution is completed at the position k of the factory f and before the job processing is started at the position $k+1$, where $k \leq n$. The completion time of different factories is expressed in the constraint set (5), while the completion time of the entire problem is represented by the constraint set (6). Constraint sets (7)–(9) define the decision variables. The constraint sets (7) and (8) ensure that the completion time of each job and the machine idle time before the next job which is processed are non-negative.

Compared with traditional single factory scheduling problem, scheduling problem in distributed factory is more complex. n jobs need to be assigned to F suitable factories. The possible combination for assigning jobs to factory f_1 is $\binom{n}{f_1}$ and there are $f_1!$ sequences in each of the combinations. f_h represents the number of jobs assigned to factory h.The result of n jobs assigned to F factories is $\binom{n}{f_1} f_1! \times \binom{n-f_2}{f_2} f_2! \times \cdots \times \binom{n-\sum_{h=1}^{F-1} f_h}{f_F} f_F!$. The result of the problem is $n!$.

In addition, there are different ways to assign n jobs to F different factories. The number of these methods is $\binom{n+F-1}{F-1}$.

So, the complexity of the DPFSP is $\binom{n+F-1}{F-1} n!$.

Factory 1: The maximum completion time C_{max} of the job sequence 1, 2, 3 is 41.

Factory 2: The maximum completion time C_{max} of the job sequence 4, 5, 6 is 37. So, the maximum completion time in the example is 41.

When the number of jobs is 6 and the number of factories is 2, the DPFSP problem complexity is 5040.

$$f_{(1)}(6, 2) = \binom{6+2-1}{2-1} 6! = \binom{7}{1} 6! = 7 * 720 = 5040$$

Brief introduction to DE algorithm

Traditional DE algorithm is an intelligent metaheuristic method introduced by Storn and Price [17]. The convergence result of DE is mainly determined by the mutation operator and the crossover operator. Starting from the random initialization of individuals in the population, it creates new candidate solutions by mutation strategy and crossover operator, and then the selection operators are used to determine the next generation of target individuals. Furthermore, the control parameters of DE play an important role in balancing the convergence speed of the algorithm and diversity of population, such as population size NP, scaling factor ω, crossover rate Cr. The basic DE algorithm process is procedure as follows.

Step 1: Initialization phase. The algorithm randomly generates NP particles to form the initial population and set the parameters.

Step 2: Mutation phase. Mutation vector $\mathbf{v}_{i,g}$ is created in DE using the "DE/$rand$/1" mutation strategy. Two vectors are randomly selected in "DE/$rand$/1". The difference of the two vectors is multiplied by the scaling factor ω and added to a third randomly selected vector as follows.

$$\mathbf{v}_{i,j} = \mathbf{x}_{r_1,j} + \omega \cdot (\mathbf{v}_{r_2,j} - \mathbf{v}_{r_3,j}), r_1 \neq r_2 \neq r_3 \neq i \tag{10}$$

where r_1, r_2, and r_3 are different from each other and randomly selected indexes of the $[1, NP]$.

Step 3: Crossover phase. The mutation vector $\mathbf{v}_{i,g}^g$ generated by the mutation operation is used for crossover operation to generate a trial vector $\mathbf{u}_{i,g}^g$.

$$\mathbf{u}_{i,j}^g = \begin{cases} \mathbf{v}_{i,j}^g, \text{if}(rand(0,1) \leq Cr \ or \ j = j_{rand}) \\ \mathbf{x}_{i,j}^g, \quad \text{otherwise} \end{cases} \tag{11}$$

where $i \in [1, NP]$ and $j \in [1, D]$, $Cr \in [0, 1]$ is crossover rate, a random integer uniformly distributed in $[1, D]$ is represented by j_rand, D is the dimension of the search space.

Step 4: Selection phase. The options are as follows.

$$\mathbf{x}_i^{g+1} = \begin{cases} \mathbf{u}_i^g, \text{if}(f(u_i^g) < f(x_i^g)) \\ \mathbf{x}_i^g, \quad \text{otherwise} \end{cases} \tag{12}$$

Step 5: If a stopping criterion is not satisfied, the result is output; otherwise it returns to step 2.

MDDE for DPFSP with minimizing makespan

The DE algorithm was originally designed to solve the continuous optimization problem, it is not used to directly generate discrete job sequences. Therefore, the MDDE algorithm for solving the problem of blocking flow shop scheduling is proposed by Wang et al. [27]. Inspired by the HDDE algorithm, the MDDE algorithm for solving the DPFSP with a makespan criterion is presented. The details of MDDE algorithm details include population initialization, mutation and crossover operator design, local search and variable neighborhood search, and elite retention strategies.

Initial population of DNEH+T

The principle of distributing two kinds of jobs to the factories is proposed by Naderi and Ruiz [11]. The first is to assign the job to the factory with the lowest current makespan, excluding the inserting job (denoted as NEH1). The second is the lowest current makespan (denoted NEH2) after the job is assigned to the factory. The "NEH" is the Nawaz–Enscore–Ham algorithm which is one of the most effective heuristic algorithms. An improved NEH (DNEH + Dipak) was proposed by Shao et al. [4] to solve the distributed no-wait flow shop scheduling problem (DNWFSP).

In this study, a modified version of DNEH + Dipak is applied to construct the initial population, which is named as DNEH + T. The detail of DNEH + T method is described as follows.

Supposing s is a sequence of k jobs to be assigned, assigning the k jobs to two factories, the steps are as follows.

Step 1: According to the principle of the shortest processing time (SPT Rule), the total processing time of each job is arranged in ascending order, obtaining a new sequence $s^{'}$ of jobs to be assigned.
Step 2: Insert the first two jobs of $s^{'}$ into the two factories, respectively.
Step 3: Insert the third job of $s^{'}$ into all possible locations in the two factories, respectively. Find the best factory location for the job.
Step 4: Repeat the Step 3 until all jobs of $s^{'}$ have been allocated.

In addition, the insertion process of jobs using Taillard's acceleration method [33] greatly increase the speed of operation. The pseudo-code of initial population is shown in Algorithm 1. The detailed application rules of DNEH + T are shown in Fig. 3.

Algorithm 1 DNEH+T

Calculate $P_j = \sum_{i=1}^{m} p_{i,j}$, $j = 1, 2, \ldots, n$
Sort all jobs in descending order according to P_j, denoted as $\sigma = [\sigma_1, \sigma_2, \cdots, \sigma_n]$
for $k = 1 : n$
  insert the job σ_k in all possible
  if the job number of the factory >2
    for each job in this factory
      insert the job in other positions of the factory
      place the job in the position of the current factory with the lowest makespan
    end for
  end if
end for

The job permutation-based mutation operator

Mutation operator is one of the significant operators of DE algorithms. A good mutation operator can prevent the algorithm from falling into the local optimal solution, and makes the algorithm perform well in solving speed and precision. Inspired by the LSAHDE [21] algorithm, the mutation strategy $current-to-pbest/1$ is used in the MDDE algorithm.

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} \oplus \omega \otimes (\mathbf{v}_{pb,g} - \mathbf{x}_{i,g}) \oplus \omega \otimes (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \tag{13}$$

where $x_{pb,g}$ is the best individual at the current generation g. $x_{r_1,g}$ and $x_{r_2,g}$ are two different individuals at the current population. $x_{i,g}$ is the individual involved in mutation. $v_{i,g}$ is the offspring individual of $x_{i,g}$ by the mutation operation. The definitions of '$-$', '$\otimes$' and '$\oplus$' in Eq. (13) are separately described as follows.

$$\begin{aligned} \triangle_1 &= \omega \otimes (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \\ &\Leftrightarrow \sigma_j = \begin{cases} \mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}, & \text{if } rand(0,1) < \omega \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{14}$$

$$\begin{aligned} \triangle_2 &= \omega \otimes (\mathbf{x}_{pb,g} - \mathbf{x}_{i,g}) \\ &\Leftrightarrow \pi_j = \begin{cases} \mathbf{x}_{pb,g} - \mathbf{x}_{i,g}, & \text{if } rand(0,1) < \omega \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{15}$$

where $\triangle_1$ and $\triangle_2$ are temporary vector.

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} \oplus \triangle_1 \oplus \triangle_2 = \mathrm{mod}((\mathbf{x}_{i,g} + \triangle_1 + \triangle_2 + n), n) \tag{16}$$

where 'mod' represents the mathematical operation of Mod. n is the dimension of the individual. The detailed pseudo-code of mutation operator is shown in Algorithm 2.

From Algorithm 2, $v_{i,g}$ does not represent a complete sequence, as individual jobs are repeated multiple times or lost. Mutant individuals are used to enhance the perturbation to the target individuals to increase the exploitation and

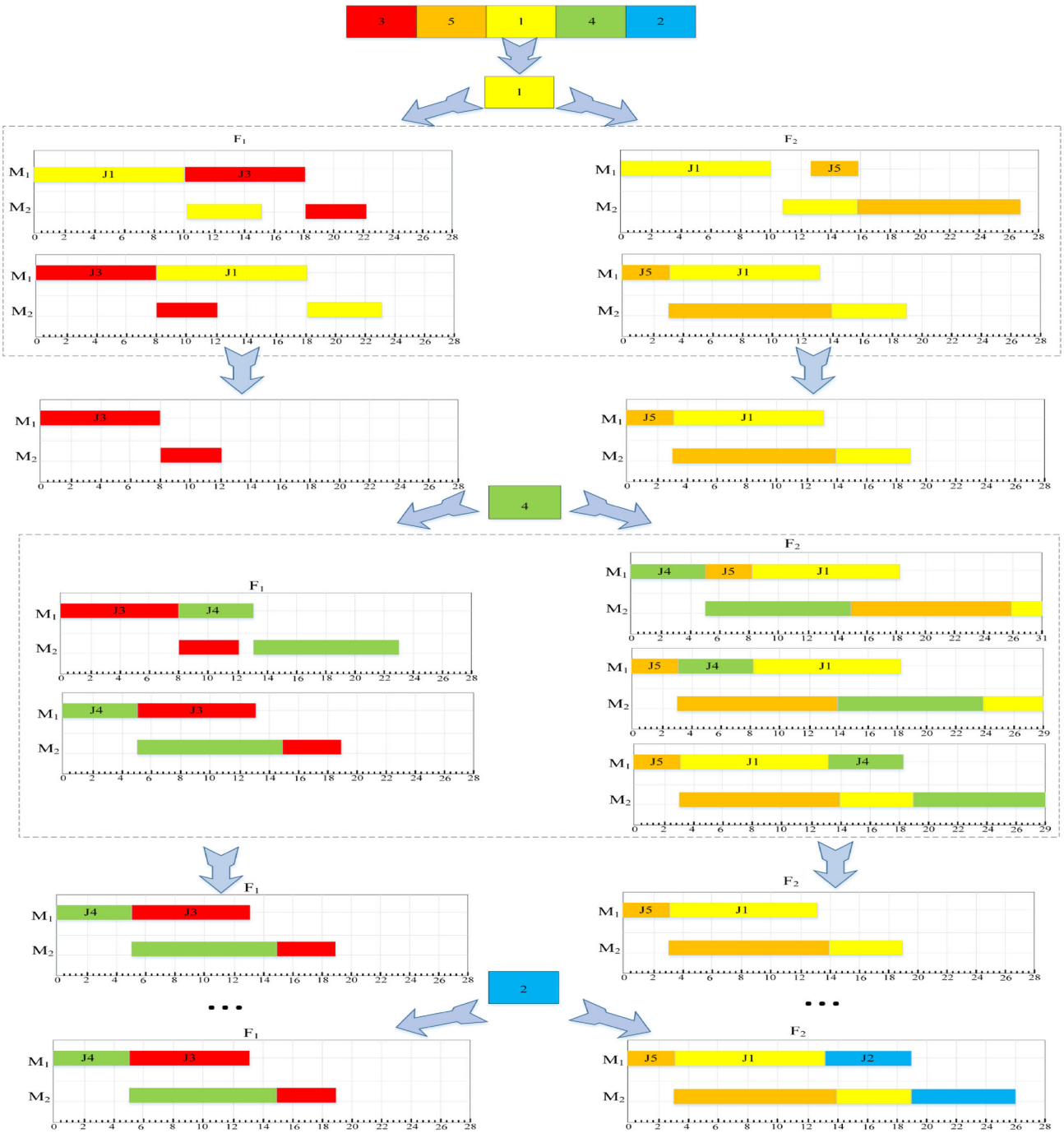

Fig. 3 The examples of Initial population

exploration ability of the algorithm. Therefore, the legitimate target individuals are obtained through the following crossover operators.

Crossover operator

The mutant individual $V_{i,g}$ and the target individual $x_{i,g-1}$ are combined to generate a trial individual $u_{i,g}$ by crossover operator. The purpose of crossover operation is to transform the unreasonable sequence of jobs produced by mutation operation into reasonable sequence of jobs. The details are as follows.

Assuming that v_i is a sequence of repeated jobs produced by the mutation operation, and x_i is a sequence containing non-repeated jobs. The length of v_i and x_i is n.

Algorithm 2 Mutation operator

for $pop = 1 : NP$
The $r1, r2$ are randomly selected in the population NP, and $r1 \neq r2$;
  if $rand() < \omega$
    Calculate the values of σ_j and π_j;
  end if
  The values of σ_j and π_j are 0;
  Calculate $v_{i,g}$ according to Eq.(16);
end for

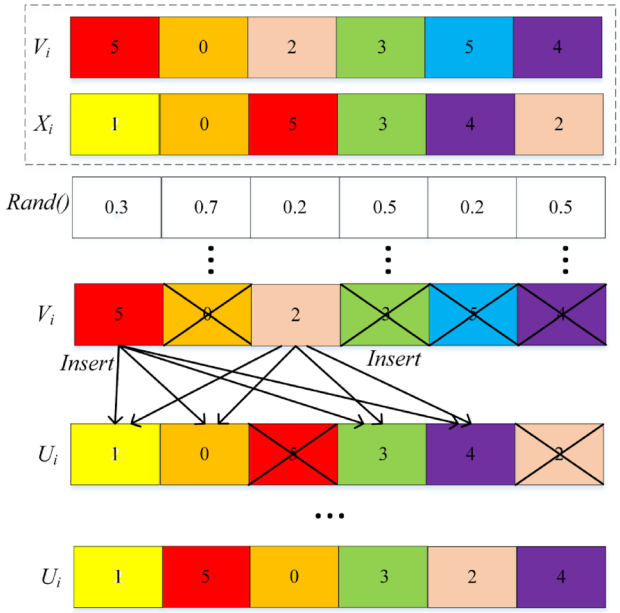

Fig. 4 The examples of crossover operators

Step 1: From $j = 1$ to n, if $rand() > Cr$ or the job is not the first time appear in v_i, the job is deleted from v_i. Assigning v_i that completes the step to V_i, that is, $V_i = v_i$.
Step 2: Remove jobs contained in V_i from x_i and assigning the x_i that completes the step to U_i, that is, $U_i = x_i$.
Step 3: Each job is fetched from V_i and re-inserted to any possible location in U_i, until the best location for the job is found.

The detailed process is shown in Fig. 4. Assuming that $u_i = (5, 0, 2, 3, 5, 4)$ is the mutation sequence.

Switching mechanism based on neighborhood structures

The main ideas of neighborhood search are divided into the following aspects. First, a local minimum in a neighborhood structure is not necessarily applied to another neighborhood structure. Second, the global optimal value is the local optimal value involving all possible neighborhood structures. Finally, for various problems, the local optimal values of multiple neighborhoods are relatively close to each other [4]. In the design of the neighborhood, supposing that if the schedule in the factory of the largest makespan (represented as the critical factory f_c) is not changed, the solution makespan is not reduced. In MDDE, four neighborhood structures are introduced in this paper, improving the exploitation and exploration ability of the algorithm. These neighborhood structures are divided into two categories. One is assignment based on factory, including Critical-swap-multi (N_1) and Critical-insert-multi (N_2). The other is adjustment based on job order, including Critical-swap-single (N_3) and Critical-insert-single (N_4). An example of the four neighborhood structures is shown in Fig. 5. The detailed descriptions of the four neighborhood structures are as follows.

- N_1: A job J is randomly selected from the f_c factory. Randomly select a job J_f from other factories, and then exchange the positions of J and J_f respectively. Then, $F-1$ new solutions are generated, and the current optimal solution is selected as the next neighborhood for local search.
- N_2: A job J is randomly selected from the factory f_c, and insert it into a randomly selected position in other factories.
- N_3: Randomly select l jobs from the factory f_c. A job J is randomly selected from l jobs and exchanged with other $l-1$ jobs in the factory.

Neighborhood structure based local search

The local search takes a significant effect in the MDDE algorithm. Four local search methods are used to improve the solution as shown in Fig. 6. LS_insert_critical_factory1 and LS_insert_critical_factory2 indicate that a job is taken from the critical factory f_c and inserted into all other factories or critical factory. The location of the best makespan is chosen for that factory. Each job in the critical factory is exchanged with each job of another factory as shown in LS_swap. LS_insert means that a pair of job taken from critical factory and non-critical factory and then they are inserted into other factories respectively to find the best completion time. The local search is terminated when the maximum makespan is no longer improved.

The MDDE algorithm

The general diagram of MDDE algorithm is shown in Fig. 7. The detailed pseudo-code of MDDE is summarized in Algorithm 3.

In the initialization stage, the $DNEH+T$ method is introduced to generate potential candidate solution. The improved mutation strategies are used to assist in the exploration and

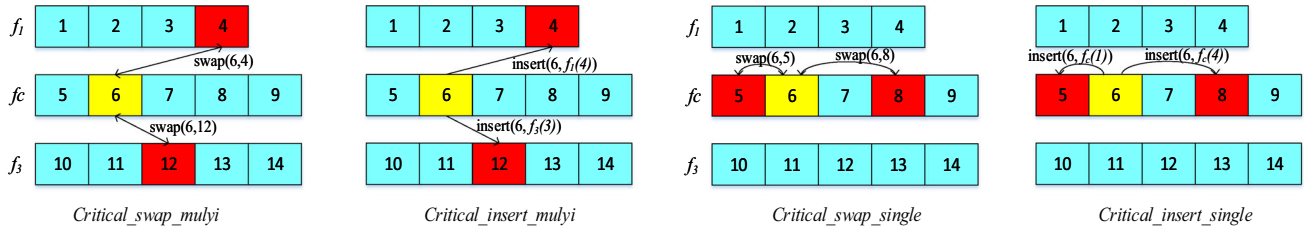

Fig. 5 Neighborhood structures of MDDE

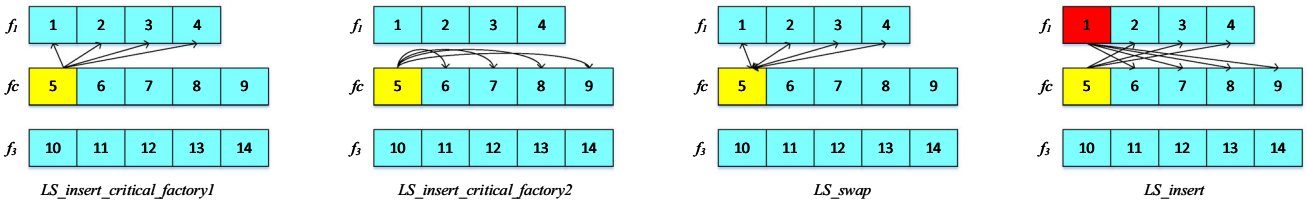

Fig. 6 Local search methods of MDDE

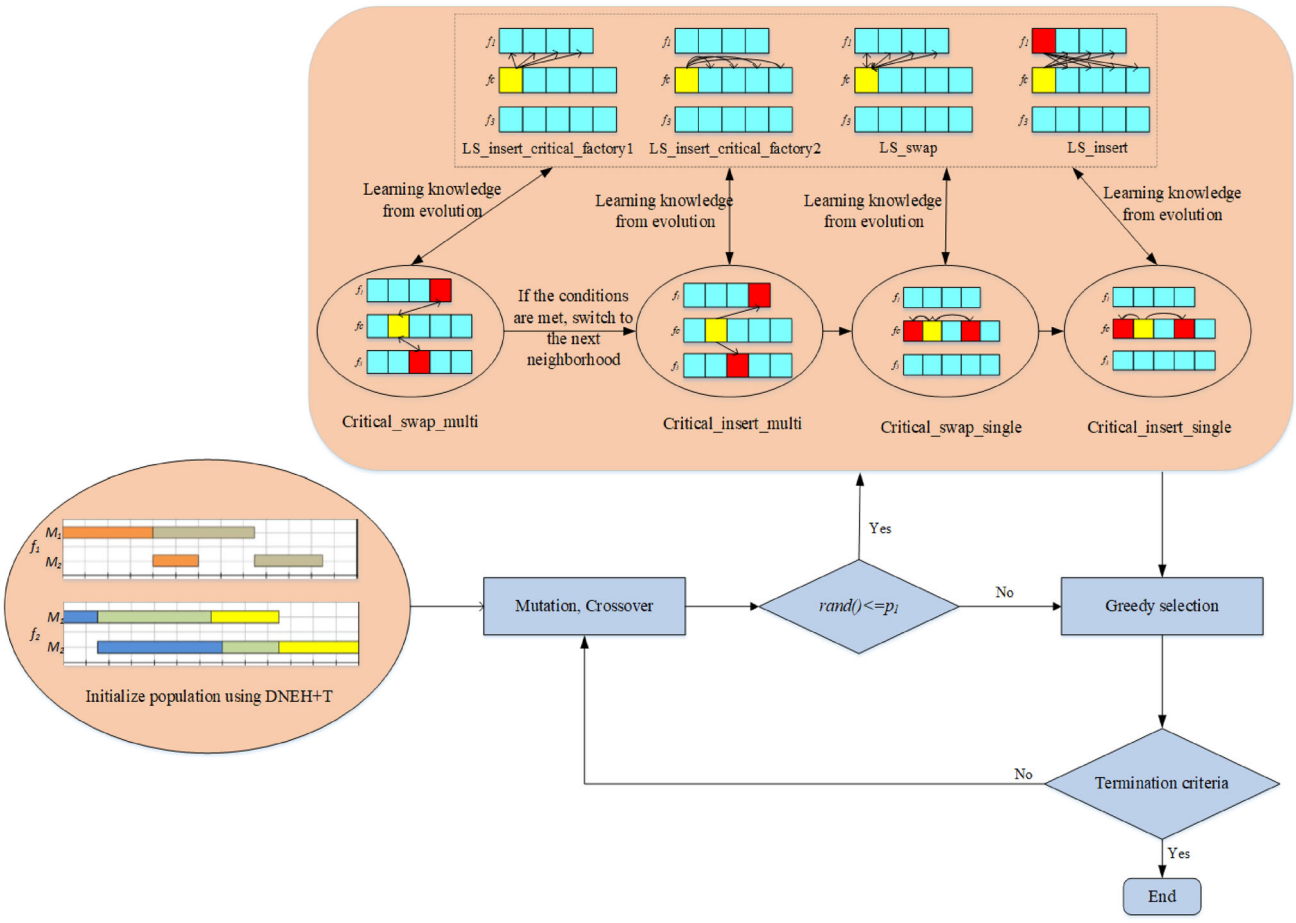

Fig. 7 Generic diagram of MDDE

Algorithm 3 The main procedure of MDDE

```
Input:Test function
Initialize population with DNEH+T strategy
Initialize operator ω, Cr and P1
while t ≤ T
  for i = 1 : Np
    Mutation operation
    The trial individual u_{i,g} is generated by the crossover operator.
    if rand() ≤ P1
      while k < K_max
        Select variable neighborhood search
        while(flag)
          Select local search
        end while
        k = k + 1
      end while
    end if
    Greedy selection is used to preserve the local optimum.
  end for
end while
Output C_max
```

exploitation of the algorithms. A switching mechanism is designed to determine whether to enter variable neighborhood search after the crossover operation of MDDE to reduce the complexity of the algorithm. A switch probability P_1 is given and a random number between 0 and 1 is generated after each mutation operation of the population. If the random number is less than probability P_1, the individual enters variable neighborhood search and local search. The selection of neighborhood is also one of the important factors to improve the search ability of the algorithm. Finally, the target individuals in the population are selected for the next generation by greedy selection.

In this paper, the knowledge used to guide search is divided into two kinds: knowledge of problem and knowledge of algorithm optimization.

For the distributed permutation flow-shop scheduling problem, optimal sequence of jobs needs to be found to minimize the maximum completion time. In the process of algorithm search, it is easy to find multiple local optimal solutions. In order to further optimize the current solution, it is necessary to set the neighborhood structure of the solution. A local optimum is optimal in a certain neighborhood, but it may not be optimal after changing the neighborhood. At this point, the solution has room for improvement. Therefore, the design of reasonable neighborhood structure based on problem knowledge can guide the evolution of individuals.

The local search of variable neighborhood is the key operation in this algorithm. In this section, four neighborhood structures are designed and a knowledge-based neighborhood selection strategy is proposed. The feedback results of local search in the neighborhood during the evolution of the algorithm are extracted to decide whether to switch neighborhoods or not. When there are still potential candidate solutions in the neighborhood, the local search is continued in the neighborhood. The algorithm switches to the next neighborhood to continue searching, when potential candidate solution is not found in this neighborhood. The historical information in the evolution process of the algorithm is extracted as a kind of guiding knowledge to switch neighborhood search, which helps to further enhance the exploitation ability of MDDE.

The experiments and comparisons

In this section, the optimal combinations of parameters in the MDDE algorithm are testified by the DOE. Afterwards, MDDE is compared with the other four state-of-the-art algorithms. The computational simulation used the benchmark proposed by Naderi and Ruiz [11], which is extension of the Taillard [34] benchmark. The small instance problem of DPFSP has been well solved by various algorithms, so the algorithm in this study is used to solve the large instance of DPFSP. The number of factories is set to $F = 2, 3, \ldots, 7$. More details are found at http://soa.iti.es.

Design of the experiments

The same experimental conditions are used in the experiment, including the same stopping criteria and the same programming language. All algorithms are programmed with MATLAB (2016b). The experiments run on a PC with 3.4 GHz Intel (R) Core i7-6700 CPU, 16GB RAM and 64-bit OS to ensure the fairness of the algorithm. The average relative percentage deviation (ARPD) index is used to measure the results and is calculated as follows.

$$\mathrm{ARPD} = \frac{1}{R} \cdot \sum_{i=1}^{R} \frac{C_i - C_{\mathrm{opt}}}{C_{\mathrm{opt}}} \cdot 100 \tag{17}$$

where C_i represents the solution generated by the specific algorithm in the ith experiment of the given instance. R is expressed as the number of runs. C_{opt} is represented as the optimal value of the results of all algorithms. ARPD represents the minimum found by all the algorithms in this paper.

In this study, the termination time of all comparison algorithms is set to $T_{\max} = n \times m \times f \times C$ millisecond (ms), and the time-level $C = 5, 15, 30$. All comparison algorithms are run independently 10 times on the benchmark function.

Parameters analysis

The parameter calibration experiments contribute to improve the efficiency and performance of the algorithm because the control parameters have an important influence in DE [35]. In this section, the DOE method is used to determine a suitable

Table 1 ANOVA results for parameter of MDDE

Source	Sum of squares	Degrees of freedom	Mean square	F-ratio	P value
Cr	25.86	4	12.63	19.3	**0.0044**
ω	37.93	3	18.96	15.8	**0.0216**
P_1	2.15	2	1.07	0.9	0.4145
NP	2.22	2	0.11	0.09	0.9116
$Cr * \omega$	5.02	4	1.26	1.05	0.3927
$Cr * P_1$	7.05	4	1.76	1.47	0.2257
$Cr * NP$	5.16	4	1.29	1.08	0.3787
$\omega * P_1$	0.51	4	0.13	0.11	0.9797
$\omega * NP$	1.81	4	0.45	0.38	0.8229
$P_1 * NP$	4.68	4	1.17	0.98	0.4291
Residual	57.53	48	1.99		
Total	88.18	80			

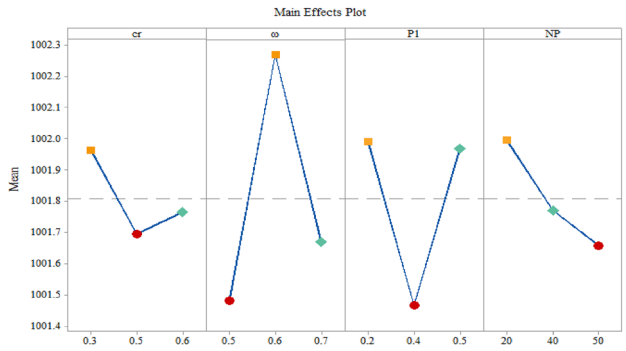

Fig. 8 Main effects plot of parameters

set of parameters for MDDE [32]. MDDE has four critical parameters: Cr (crossover rate), ω (scaling factor), P_1, NP (population size). The selected parameters are as follows: $Cr \in 0.3, 0.5, 0.6$, $\omega \in 0.5, 0.6, 0.7$, $P_1 \in 0.2, 0.4, 0.5$, $NP \in 20, 40, 50$. The configurations of all possible combinations of the four parameters are $3 \times 3 \times 3 \times 3 = 81$. Since the calibration algorithm using the same example result in over fitted results [36], 52 instances of randomly generated with various numbers of jobs, machines, and factories. Each parameter combination is executed 5 times. Following the literature [37], the experimental results are analyzed by the multivariate analysis of variance (ANOVA). In the DOE experiment, the parameters have a distinctiveness effect on the algorithm, when the confidence level of the P value is less than 0.05. The results of ANOVA are reported in Table 1.

After running all parameter configurations, the results show that the parameters P_1 and NP have no significantly effect on MDDE. According to the results from Table 1, the P values of parameters Cr and ω are less than the confidence level ($\alpha = 0.05$), indicating that these parameters have a greater impact on MDDE than other parameters. Meanwhile, the parameters Cr corresponds to the greatest F-ratio. It suggests that the parameters Cr have the greatest effect on the average performance of MDDE among all factors of consideration. From Fig. 8, the parameters are selected as follows: $Cr = 0.5$, $\omega = 0.5$, $P_1 = 0.4$, and $NP = 50$.

Analysis and discussion

In this study, the proposed MDDE algorithm is compared with the four optimal competitive algorithms for FSFP. The details of four comparison algorithms are described as follows.

- In the HDDE algorithm [31], discrete mutation and crossover operators, and local search are used to solve the problem of blocking flow shop scheduling.
- An efficient hybrid DDE algorithm is used to solve distributed blocking flow shop scheduling problem, including a unique elitist retain strategy and a biased selection operator [29].
- The CDE algorithm is used to solve the distributed finite buffer flow shop scheduling problem, using two constructive heuristics to create excellent initialization [30].
- A two-stage Iterated greedy algorithm [5] is used to solve the distributed permutation flow shop scheduling problem, which includes construction improvement, destruction procedures, and a local search.

The MDDE algorithm is used to test on 720 large-scale instances. The ARPD values grouped by the number of factories F are listed in Table 2, 3 and 4, where the best results are shown in boldface. From Table 2, 3 and 4, the performance of MDDE is better than other variants on the most instances. The reason is that MDDE algorithm uses a knowledge-based ensemble strategy, which improves the search accuracy of the algorithm. The mean and 95% Fisher's least-significant difference (LSD) interval for MDDE is presented in Fig. 9, 10 and 11. Therefore, the proposed MDDE algorithm is slightly better than the other four comparison algorithms.

Statistical tests show that MDDE algorithm has significant improvement compared with the comparison algorithms. In this study, Wilcoxon's sign rank test [38] is selected for non-parametric statistical tests. From Table 5, R^+ represents that the sum of ranks for the functions of MDDE is better than the comparison algorithm in the row, and R^- represents that the sum of ranks the functions of the comparison algorithm is superior to the MDDE.

It is worth noting that in pair-wise comparison, MDDE is the first algorithm in the row. The statistical results are listed in Table 5, MDDE is significantly better than the other four comparison algorithms in the case of $C = 15$.

Table 2 Calculation results of ARPD value with $C = 5$

n*m	F2					F3					F4				
	HDDE	DDE	CDE	IG	MDDE	HDDE	DDE	CDE	IG	MDDE	HDDE	DDE	CDE	IG	MDDE
20*5	1.482	1.539	1.812	0.296	**0.213**	1.55	2.448	1.83	0.309	**0.282**	1.832	1.774	2.585	0.558	**0.202**
20*10	2.004	2.143	2.623	**0.293**	0.31	2	1.353	1.954	**0.18**	0.274	2.08	1.632	2.199	0.349	**0.158**
20*20	1.118	1.885	1.585	**0.137**	0.241	1.374	1.463	1.421	**0.241**	0.284	1.05	2.078	1.852	0.325	**0.213**
50*5	2.408	1.293	0.658	0.488	**0.423**	3.549	1.14	0.829	0.723	**0.217**	4.252	0.705	1.406	**0.207**	0.319
50*10	2.713	1.068	1.739	0.656	**0.215**	3.528	0.962	1.672	**0.173**	0.287	3.814	1.674	1.688	**0.233**	0.319
50*20	3.163	1.564	1.633	1.003	**0.081**	2.849	1.297	1.055	0.252	**0.213**	3.215	1.598	1.577	0.346	**0.264**
100*5	1.42	0.75	0.499	0.28	**0.022**	2.548	0.483	0.62	0.397	**0.168**	3.797	0.838	0.975	0.523	**0.325**
100*10	3.154	1.272	0.979	0.792	**0.038**	4.006	0.966	1.348	0.84	**0.126**	4.572	1.042	1.574	0.654	**0.225**
100*20	3.34	1.61	1.063	1.286	**0.062**	3.565	1.102	0.943	0.945	**0.057**	3.677	0.975	1.017	0.551	**0.199**
200*10	2.416	0.644	0.825	0.659	**0.095**	3.448	0.889	1.059	0.868	**0.079**	4.064	0.905	1.101	0.825	**0.076**
200*20	3.205	1.033	1.006	1.001	**0.025**	3.949	1.216	0.986	1.026	**0.004**	3.995	0.742	0.724	1.054	**0.055**
500*20	3.01	**0.777**	1.183	1.023	0.793	2.963	0.423	0.406	0.529	**0.34**	3.819	0.525	0.795	0.72	**0.219**
Avg	2.45	1.3	1.3	0.66	**0.21**	2.94	1.15	1.18	0.54	**0.19**	3.35	1.21	1.46	0.53	**0.21**
n*m	F5					F6					F7				
	HDDE	DDE	CDE	IG	MDDE	HDDE	DDE	CDE	IG	MDDE	HDDE	DDE	CDE	IG	MDDE
20*5	1.395	2.24	2.341	0.681	**0.347**	1.777	4.442	4.606	1.789	**0.219**	1.624	2.925	3.394	4.786	**0.137**
20*10	1.717	1.841	3.712	0.665	**0.22**	1.738	2.882	3.514	2.336	**0.288**	1.479	4.48	5.013	4.756	**0.195**
20*20	0.894	2.819	2.093	0.762	**0.343**	1.084	2.95	2.705	1.933	**0.236**	0.705	2.093	4.185	3.71	**0.266**
50*5	5.307	2.222	1.244	**0.087**	0.598	5.486	2.058	2.513	**0**	0.604	5.78	2.548	2.161	**0.037**	0.462
50*10	4.482	1.895	1.505	**0.147**	0.49	4.364	2.294	1.93	**0.127**	0.529	4.45	2.441	1.583	**0.069**	0.391
50*20	3.517	1.555	1.687	**0.2**	0.448	3.627	1.836	1.909	**0.067**	0.482	3.628	1.537	1.649	**0.021**	0.538
100*5	4.468	0.676	0.875	**0.19**	0.364	5.142	0.964	1.057	**0**	0.517	5.724	1.437	1.811	**0.134**	0.301
100*10	4.73	1.057	0.768	**0.198**	0.254	5.004	1.167	1.3	**0.053**	0.411	5.128	1.576	1.163	**0.031**	0.612
100*20	3.997	0.994	1.007	0.352	**0.046**	4.1	1.264	1.492	**0.169**	0.328	4.031	1.228	1.29	**0.037**	0.392
200*10	4.299	0.849	0.787	0.625	**0.127**	4.721	0.74	1.351	0.561	**0.059**	4.896	0.767	1.006	0.32	**0.16**
200*20	4.164	0.92	0.835	0.879	**0.097**	4.215	0.697	0.821	0.479	**0.181**	4.299	0.811	0.916	0.474	**0.06**
500*20	4.037	0.806	0.889	0.552	**0.141**	3.831	0.573	0.946	0.454	**0.277**	4.166	0.624	0.903	0.607	**0.159**
Avg	3.58	1.49	1.48	0.44	**0.29**	3.76	1.82	2.01	0.66	**0.34**	3.83	1.87	2.09	1.25	**0.31**

In 120 instances, the ARPD of MDDE is smaller than that of the HDDE, DDE, CDE and IG, which means that MDDE has obtained a better solution. The classical trend graphs of MDDE and the comparison algorithms are shown in Figs. 12, 13, 14, 15, 16 and 17. In these figures, the horizontal axis represents the index of the instances in the test suit, and the vertical axis represents the test results obtained by MDDE and the four comparison algorithms on all the instances. In this paper, the algorithm is tested on 120 instances when $F = 2$, $F = 3$, $F = 4$, $F = 5$, $F = 6$, $F = 7$ respectively. There are 120 points on the horizontal axis, and each point corresponds to the ARPD value of the five algorithms. Since the 120 instances are not related to each other, the ARPD value on each instance of the algorithm has no correlation.

Bonferroni–Dunn test and Friedman's test are addressed to further illustrate the significant differences between MDDE and the four comparison algorithms, as shown in Figs. 18, 19, 20 and 22. The ARPD values between different plants are compared in Figs. 18, 19, 20 and 22, and the maximum runtime is set to $T_{\max} = n \times m \times f \times 15$ ms. To assess the significance level of MDDE and other comparison algorithms, the Bonferroni–Dunn test is used to calculate the critical differences between the algorithms to compare their differences with $\alpha = 0.05$ and $\alpha = 0.1$. Although there is no significant difference between MDDE and the comparison algorithms in individual cases, the rank of MDDE is the smallest among all the test cases. From Figs. 18, 19, 20, 21 and 22, the proposed MDDE algorithm ranks first in all dimensions.

The effectiveness of discrete mutation strategy, the neighborhood structures and local search method are validated by the above experimental analysis. Furthermore, compared with the state-of-the-art algorithms from the literatures, the

Table 3 Calculation results of ARPD value with $C = 15$

n*m	F2					F3					F4				
	HDDE	DDE	CDE	IG	MDDE	HDDE	DDE	CDE	IG	MDDE	HDDE	DDE	CDE	IG	MDDE
20*5	1.129	2.475	1.151	**0.401**	0.611	1.186	2.137	1.272	**0.018**	0.458	1.801	2.883	2.677	1.366	**0.872**
20*10	1.867	2.41	2.199	**0.334**	0.403	1.397	2.645	2.499	**0.193**	0.373	1.358	2.06	2.587	0.162	**0.16**
20*20	0.823	2.152	1.632	**0.092**	0.419	1.022	2.132	1.516	**0.165**	0.151	0.834	1.874	2.298	**0.213**	0.26
50*5	1.866	0.748	1.194	**0.169**	0.415	3.267	1.481	1.486	**0.315**	0.069	4.039	1.515	1.577	**0.036**	0.427
50*10	2.527	1.574	2.014	**0.278**	0.318	3.328	1.954	1.456	**0.178**	0.403	3.785	1.737	1.071	**0.017**	0.447
50*20	2.266	1.523	1.17	**0.085**	0.297	2.901	1.403	1.716	**0.157**	0.432	3.199	1.766	1.746	**0.151**	0.439
100*5	1.227	0.396	0.424	**0.144**	0.149	2.361	0.63	1.025	0.469	**0.148**	3.119	0.814	0.644	0.21	**0.164**
100*10	2.655	0.993	0.683	0.719	**0.099**	3.605	1.215	1.488	0.67	**0.009**	4.47	1.365	1.29	0.401	**0.26**
100*20	2.82	0.922	0.857	0.955	**0.018**	3.044	0.793	1.193	0.501	**0.01**	3.299	0.828	1.114	**0.089**	0.221
200*10	2.193	0.805	0.442	0.605	**0.022**	3.062	0.596	0.781	0.636	**0**	3.895	0.997	0.939	0.82	**0.065**
200*20	3.286	0.991	0.963	1.232	**0.103**	3.764	1.064	0.815	0.918	**0.073**	3.728	0.44	1.16	0.513	**0.089**
500*20	2.9	1.026	0.561	0.969	**0.731**	3.294	0.868	0.833	1.034	**0.02**	3.684	0.809	1.1	0.855	**0.052**
Avg	2.13	1.33	1.11	0.5	**0.3**	2.69	1.41	1.34	0.44	**0.18**	3.1	1.42	1.52	0.4	**0.29**
n*m	F5					F6					F7				
	HDDE	DDE	CDE	IG	MDDE	HDDE	DDE	CDE	IG	MDDE	HDDE	DDE	CDE	IG	MDDE
20*5	0.969	2.908	2.341	1.073	**0.239**	1.545	3.746	4.416	2.99	**0.279**	1.63	4.98	4.918	3.65	**0.531**
20*10	1.712	2.159	2.642	0.686	**0.115**	1.792	3.366	3.62	2.477	**0.031**	1.274	3.512	5.4	3.133	**0.035**
20*20	0.676	2.507	2.659	1.118	**0.07**	0.56	2.026	2.924	1.779	**0.147**	1.028	4.179	4.986	4.279	**0.324**
50*5	5.048	1.812	1.953	**0**	0.755	5.627	2.961	2.036	**0.016**	0.752	5.556	2.296	2.896	**0.038**	0.84
50*10	4.103	2.015	1.565	**0.058**	0.496	4.568	2.18	1.955	**0.044**	0.448	4.607	2.084	2.319	**0.046**	0.414
50*20	3.327	1.737	1.77	**0**	0.414	3.756	1.574	1.562	**0.053**	0.515	3.922	2.135	1.705	**0.343**	0.534
100*5	4.209	0.734	1.104	**0.06**	0.128	4.657	1.226	1.247	**0.047**	0.342	5.661	1.253	1.901	**0.023**	0.388
100*10	4.748	1.339	0.996	0.234	**0.132**	4.923	1.53	1.511	**0.135**	0.34	5.257	1.879	0.879	**0.276**	0.518
100*20	3.888	0.979	1.078	**0.108**	0.24	3.991	1.043	1.462	**0.025**	0.248	4.223	1.127	1.44	**0**	0.363
200*10	4.548	0.987	0.868	0.815	**0.24**	4.432	0.754	0.975	0.275	**0.113**	4.891	0.924	0.847	**0.14**	0.175
200*20	4.013	0.976	0.643	0.392	**0.147**	4.254	1.21	0.617	0.247	**0.145**	4.502	1.06	1.611	**0.099**	0.468
500*20	4.124	0.968	0.922	0.714	**0.022**	4.172	1.156	1.082	0.844	**0.069**	4.153	0.865	0.995	0.597	**0.106**
Avg	3.45	1.59	1.55	0.44	**0.25**	3.69	1.9	1.95	0.74	**0.29**	3.89	2.19	2.49	1.05	**0.39**

proposed MDDE algorithm for solving the DPFSP is effective. The main reasons are summarized as follows.

First, the DE algorithm is an efficient intelligent optimization algorithm, which has been applied in various fields. Although DE is sensitive to parameters including the crossover rate and mutation factor, it generally provides excellent results on various complex problems. In addition, an excellent initial solution assists the algorithm in increasing the diversity of the population to discover the potential candidate solution. A poor initial population unnecessarily increases the number of searches or causes the algorithm to converge at local optima. In MDDE, the constructive DNEH + T method is used to initialize the population to increase the population diversity, and the Taillard's method is used to speed up the operational efficiency of the algorithm.

Third, the neighborhood search is one of the important factors which affect the performance of the algorithm. In MDDE, an appropriate neighborhood search mechanism is adaptively selected by a knowledge-based optimization strategy, which enhances the global search ability of the proposed algorithm. The experiments show that each local search method plays a different role in the overall performance of the algorithm. Since various local search methods are explored in different solution spaces, the mixing local search helps to avoid local optimum. At the same time, the results prove that this integration strategy is better than a single local search strategy.

It is noteworthy that the ensemble strategy focuses on improving the global search capability in the stage of exploration, including the discrete mutation strategy, neighborhood search and local search. Individuals in the population

Table 4 Calculation results of ARPD value with $C = 30$

n*m	F2					F3					F4				
	HDDE	DDE	CDE	IG	MDDE	HDDE	DDE	CDE	IG	MDDE	HDDE	DDE	CDE	IG	MDDE
20*5	0.556	2.296	1.593	**0.122**	0.25	0.789	1.524	2.281	**0.248**	0.53	1.073	1.909	2.269	0.519	**0.297**
20*10	0.871	1.783	2.188	**0.084**	0.6	0.953	2.78	2.595	**0.025**	0.664	0.679	1.858	2.633	**0.081**	0.508
20*20	0.871	2.157	2.07	**0**	0.428	0.739	1.699	1.509	**0.014**	0.614	0.578	1.511	2.041	**0.177**	0.544
50*5	1.706	1.271	1.694	0.467	**0.089**	2.812	1.288	2.054	0.353	**0.133**	3.993	1.635	1.994	**0.177**	0.036
50*10	2.523	2.41	1.706	0.452	**0.029**	3.339	1.555	1.937	**0.254**	0.356	3.497	2.141	1.891	**0.044**	0.594
50*20	2.347	1.516	1.866	0.265	**0.043**	2.705	1.525	1.517	**0.051**	0.229	2.93	1.646	1.956	**0**	0.588
100*5	1.213	0.706	0.689	0.347	**0.004**	2.07	0.81	0.788	0.388	**0**	3.221	1.009	1.183	0.28	**0.057**
100*10	3.11	1.601	1.658	1.051	**0**	3.873	1.625	1.646	0.852	**0**	4.225	1.814	1.765	0.367	**0**
100*20	3.709	2.338	2.375	1.878	**0**	3.817	2.236	2.321	1.346	**0**	4.185	1.763	1.654	0.839	**0.06**
200*10	2.754	1.176	1.201	1.269	**0.108**	3.664	1.754	1.667	1.476	**0.054**	4.336	1.755	1.61	1.528	**0.156**
200*20	4.071	2.051	1.911	2.178	**0.024**	4.411	1.828	1.943	1.98	**0.017**	4.609	1.845	2.039	1.567	**0**
500*20	2.575	0.756	1.233	0.872	**0.037**	3.171	1.018	1.32	0.949	**0.018**	3.59	0.869	1.428	0.815	**0.008**
Avg	2.19	1.67	1.68	0.75	**0.13**	2.7	1.64	1.8	0.66	**0.22**	3.08	1.65	1.87	0.53	**0.24**
n*m	F5					F6					F7				
	HDDE	DDE	CDE	IG	MDDE	HDDE	DDE	CDE	IG	MDDE	HDDE	DDE	CDE	IG	MDDE
20*5	1.245	2.404	2.798	0.631	**0.238**	2.059	2.735	4.105	2.441	**0.111**	2.334	3.94	4.501	3.747	**0.211**
20*10	1.036	1.787	2.241	0.416	**0.281**	2.546	2.881	3.752	2.182	**0.029**	3.222	3.515	4.716	3.648	**0.084**
20*20	1.094	1.807	1.736	0.601	**0.256**	1.784	1.938	2.274	1.687	**0.269**	3.681	3.398	3.431	4.208	**0.458**
50*5	4.3	2.064	1.848	**0**	0.448	4.665	1.758	2.062	**0**	0.59	5.497	2.812	2.639	**0**	0.773
50*10	4.074	2.021	2.083	**0.199**	0.695	4.008	2.636	2.03	**0**	0.592	4.433	2.394	2.11	**0.034**	0.865
50*20	2.993	1.571	1.634	**0**	0.526	3.331	1.945	2.657	**0**	0.787	3.581	2.189	2.292	**0.007**	0.639
100*5	4.013	1.316	1.164	0.138	**0.122**	4.565	1.443	1.53	0.098	**0.08**	5.21	1.485	1.829	**0.081**	0.225
100*10	5.051	1.891	1.65	**0.637**	0.699	4.86	1.765	1.689	**0.022**	0.216	5.263	1.831	2.359	**0.307**	0.604
100*20	4.102	1.784	2.172	0.643	**0.051**	4.351	1.535	2.039	0.334	**0.085**	4.255	1.961	1.649	0.242	**0.069**
200*10	4.979	2.153	1.899	1.522	**0.139**	5.086	1.923	1.896	0.868	**0**	5.468	2.28	1.784	0.808	**0.057**
200*20	4.557	1.541	1.938	1.067	**0.003**	4.536	1.69	1.494	0.791	**0.129**	4.749	1.706	1.857	0.852	**0**
500*20	3.571	0.626	1.363	0.439	**0.101**	3.977	0.685	1.451	0.682	**0.055**	4.782	1.861	1.647	1.625	**0.091**
Avg	3.42	1.75	1.88	0.52	**0.3**	3.81	1.91	2.25	0.76	**0.25**	4.37	2.45	2.57	1.3	**0.34**

become similar when the MDDE algorithm falls into the local optimal solution. Mutation strategies and crossover operations will not produce new individuals and will lead to population stagnation. For example, when MDDE algorithm falls into local optimum, even if new mutation strategy is applied in MDDE, none of the mutation strategies can help MDDE escape from the local optimal. However, the neighborhood search mechanism helps the algorithm to search the direction that is introduced to another neighborhood.

At the same time, the local search mechanism is complementary to the neighborhood search mechanism, which increases the small-range search capability of the algorithm. In addition, the current suitable neighborhood search mechanism is adaptively selected by the knowledge-based optimization strategy, which enables the algorithm to implement a closed-loop control system. Significant differences between the MDDE algorithm and the comparison algorithms are shown in Table 5. Results with significant differences are marked 'yes' and highlighted in bold. The experiment results show that MDDE outperforms other comparison algorithms. According to Table 5, MDDE proves to be significantly different from the majority of comparison algorithms by using Wilcoxon's test. The convergence accuracy and stability of MDDE are superior to other comparison algorithms, as shown in Figs. 9, 10 and 11. From Figs. 18, 19, 20, 21, 22 and 23, MDDE is superior to other comparison algorithms because it ranks higher than all comparison algorithms.

Performance analysis of each component of MDDE

In this section, the contributions of each strategy and mechanism to the MDDE algorithm are analyzed experimentally. The MDDE mainly consists of $DNEH$ initialization method, discrete mutation strategy and adaptive neighbor-

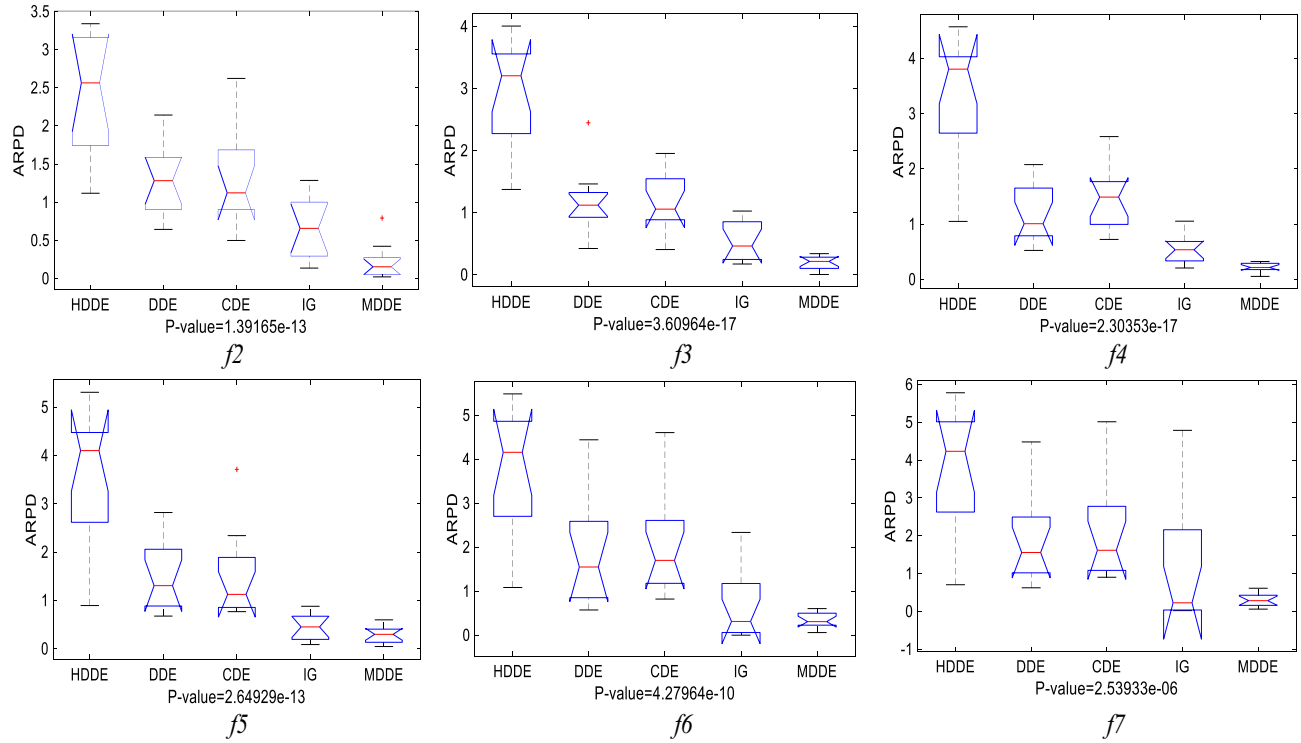

Fig. 9 The variance plot of the algorithms at $C = 5$

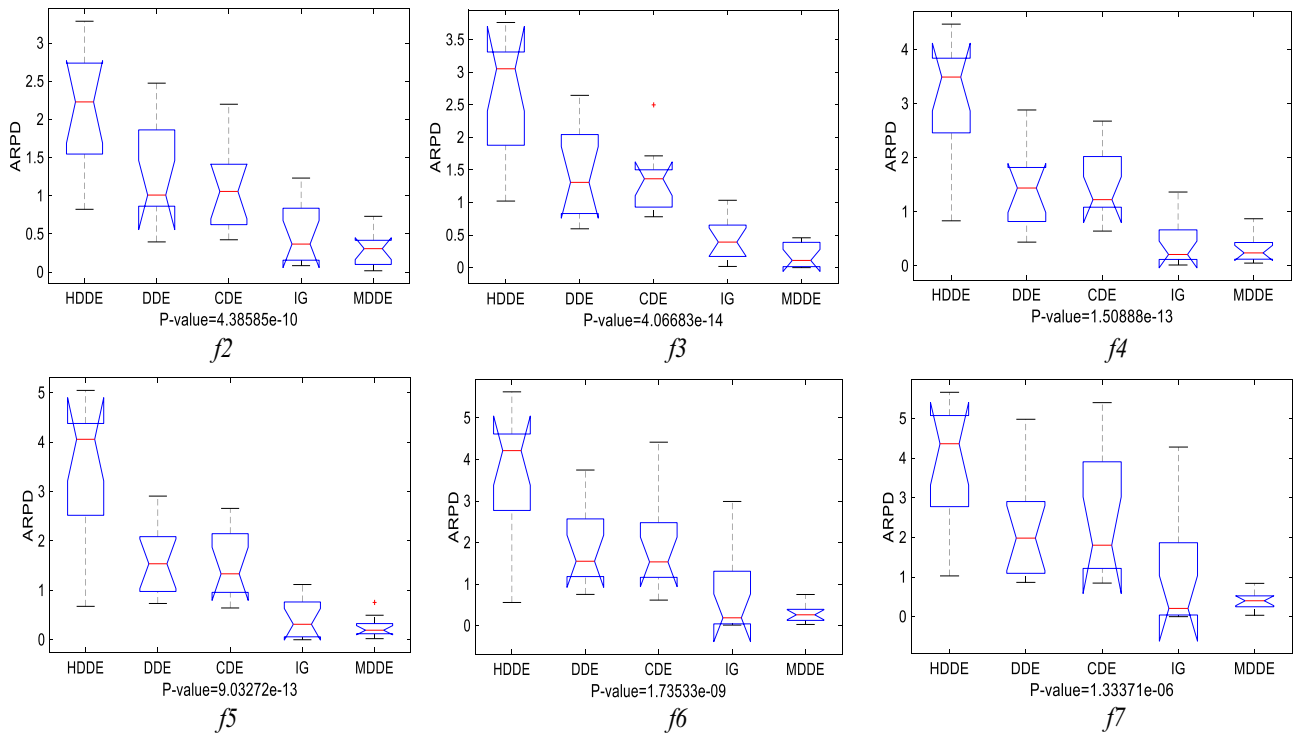

Fig. 10 The variance plot of the algorithms at $C = 15$

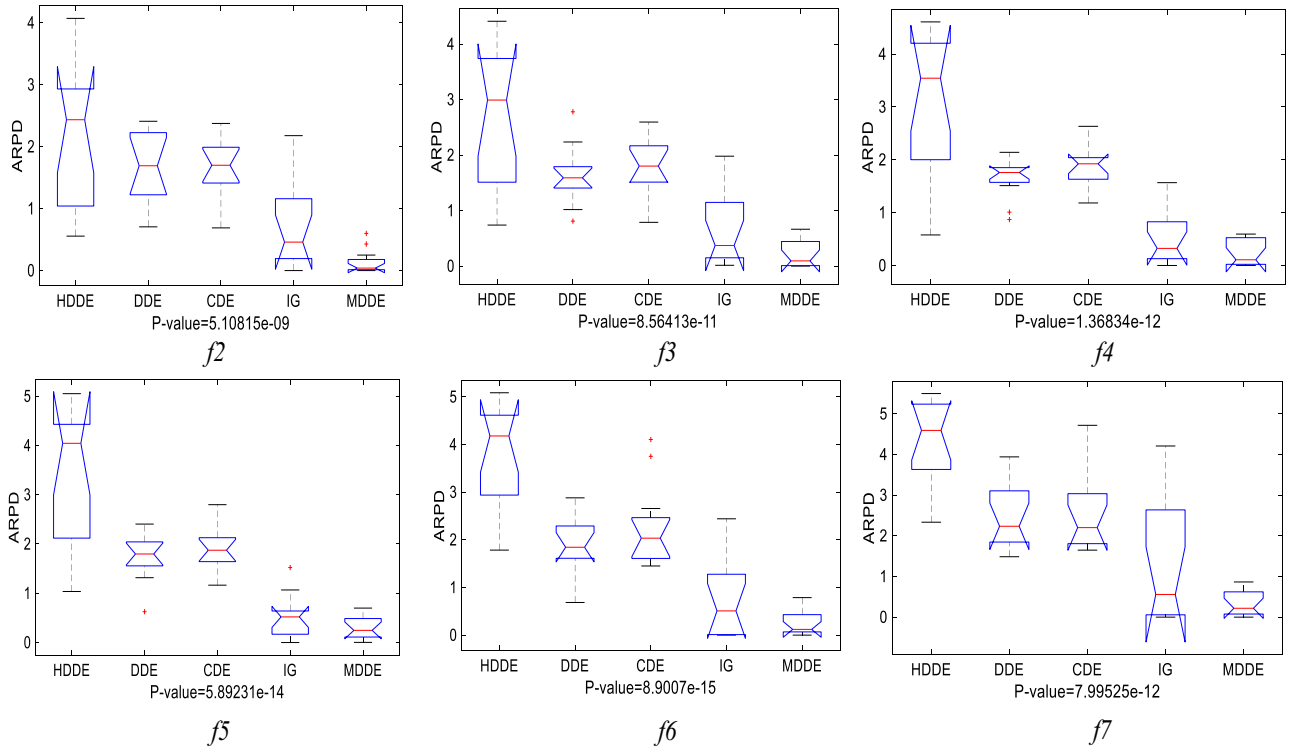

Fig. 11 The variance plot of the algorithms at $C = 30$

Table 5 Rankings obtained through the Wilcoxon signed-rank test

F	MDDE VS	R^+	R^-	Z	P value	$\alpha = 0.05$	$\alpha = 0.01$
2	HDDE	6443	112	−8.957	3.35E−19	**Yes**	**Yes**
	DDE	5623.5	371.5	−7.96	1.73E−15	**Yes**	**Yes**
	CDE	5176	602	−7.134	9.75E−13	**Yes**	**Yes**
	IG	3108.5	896.5	−4.548	0.000005	**Yes**	**Yes**
3	HDDE	7236	24	−9.449	3.42E−21	**Yes**	**Yes**
	DDE	6364	77	−9.029	1.73E−19	**Yes**	**Yes**
	CDE	5963.5	141.5	−8.7	3.31E−18	**Yes**	**Yes**
	IG	3370	546	−5.905	3.54E−09	**Yes**	**Yes**
4	HDDE	6955	66	−9.254	2.16E−20	**Yes**	**Yes**
	DDE	6031	185	−8.635	5.86E−18	**Yes**	**Yes**
	CDE	5767	228	−8.395	4.65E−17	**Yes**	**Yes**
	IG	2218.5	1184.5	−2.407	0.016092	**Yes**	**Yes**
5	HDDE	6648	22	−9.25	2.25E−20	**Yes**	**Yes**
	DDE	6080.5	135.5	−8.785	1.56E−18	**Yes**	**Yes**
	CDE	5711	284	−8.225	1.94E−16	**Yes**	**Yes**
	IG	2667.5	1427.5	−2.523	0.011649	**Yes**	**Yes**
6	HDDE	6406	35	−9.128	6.95E−20	**Yes**	**Yes**
	DDE	6303	138	−8.852	8.61E−19	**Yes**	**Yes**
	CDE	5621	157	−8.506	1.80E−17	**Yes**	**Yes**
	IG	2976.5	1488.5	−2.817	0.004852	**Yes**	**Yes**
7	HDDE	6963	58	−9.273	1.81E−20	**Yes**	**Yes**
	DDE	6589.5	196.5	−8.824	1.10E−18	**Yes**	**Yes**
	CDE	6431.5	354.5	−8.382	5.22E−17	**Yes**	**Yes**
	IG	2632	2024	−1.115	0.264898	No	No

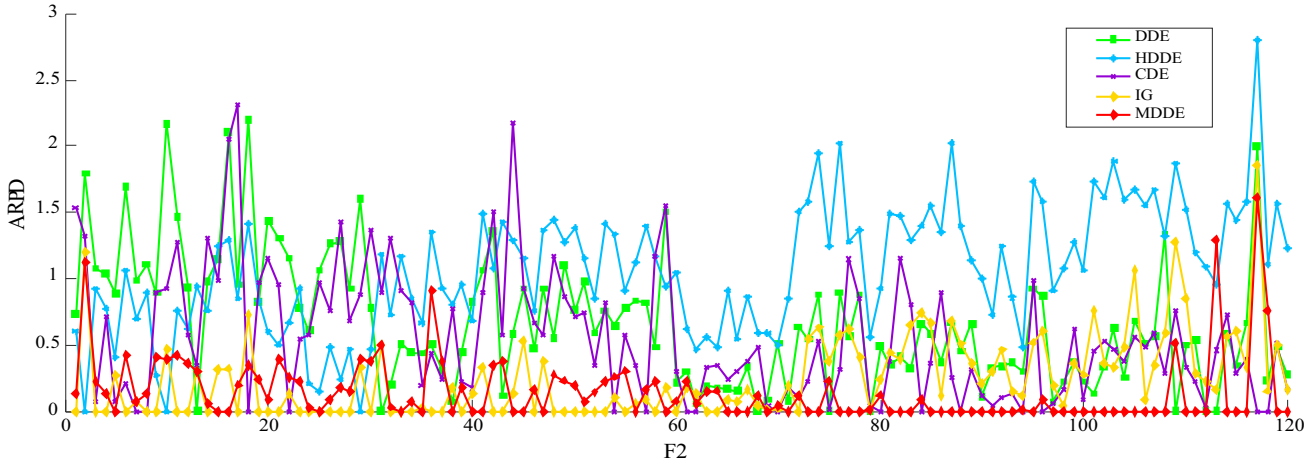

Fig. 12 ARPD of F=2

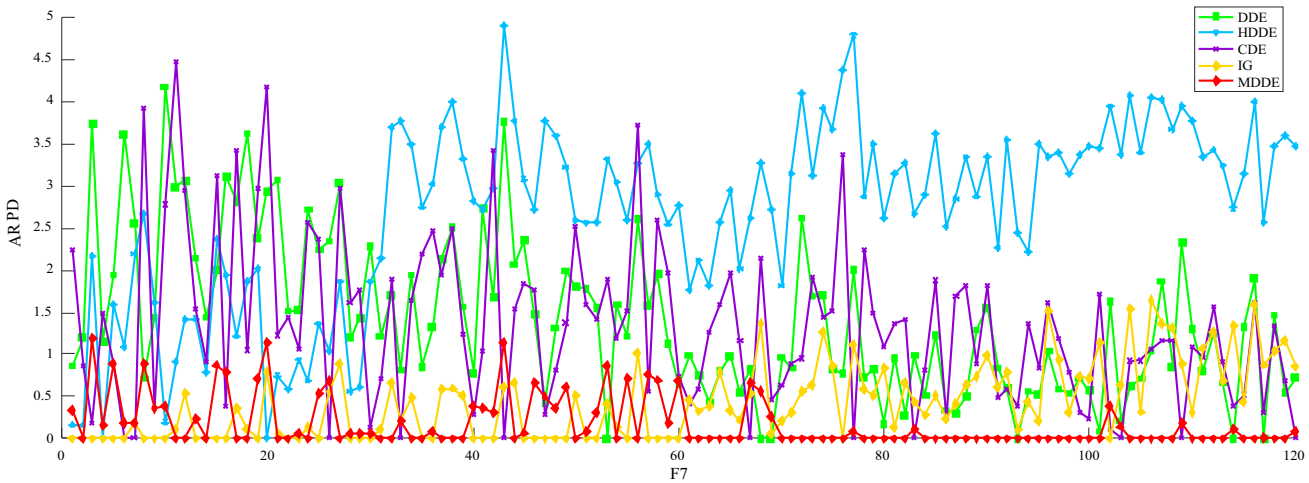

Fig. 13 ARPD of F=3

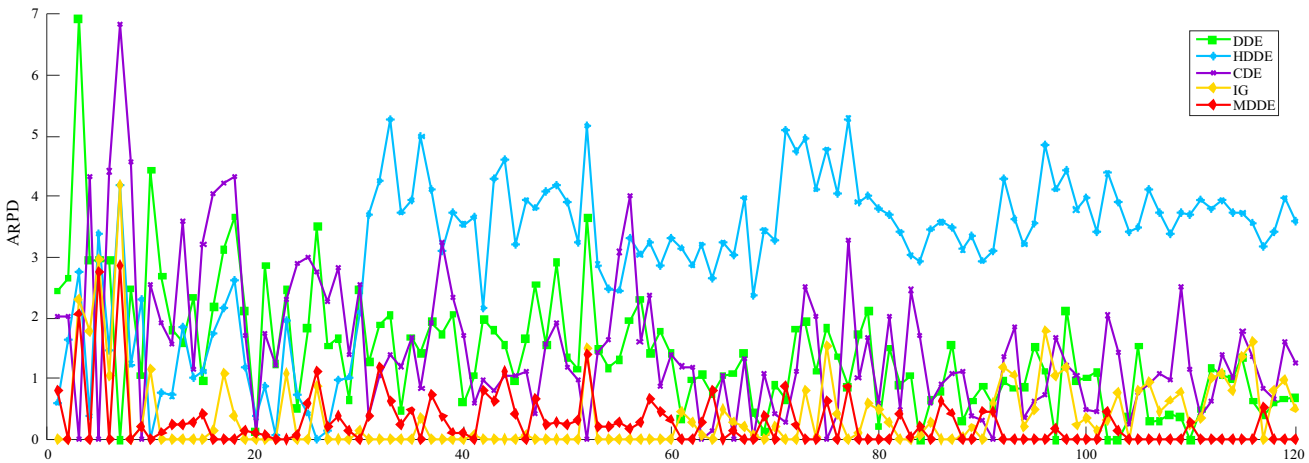

Fig. 14 ARPD of F=4

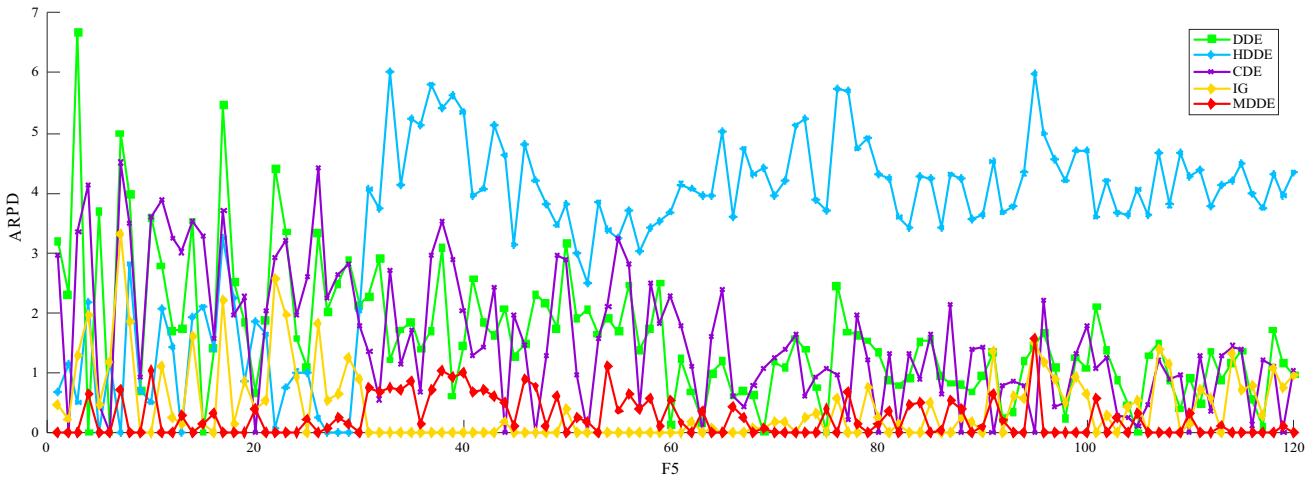

Fig. 15 ARPD of F=5

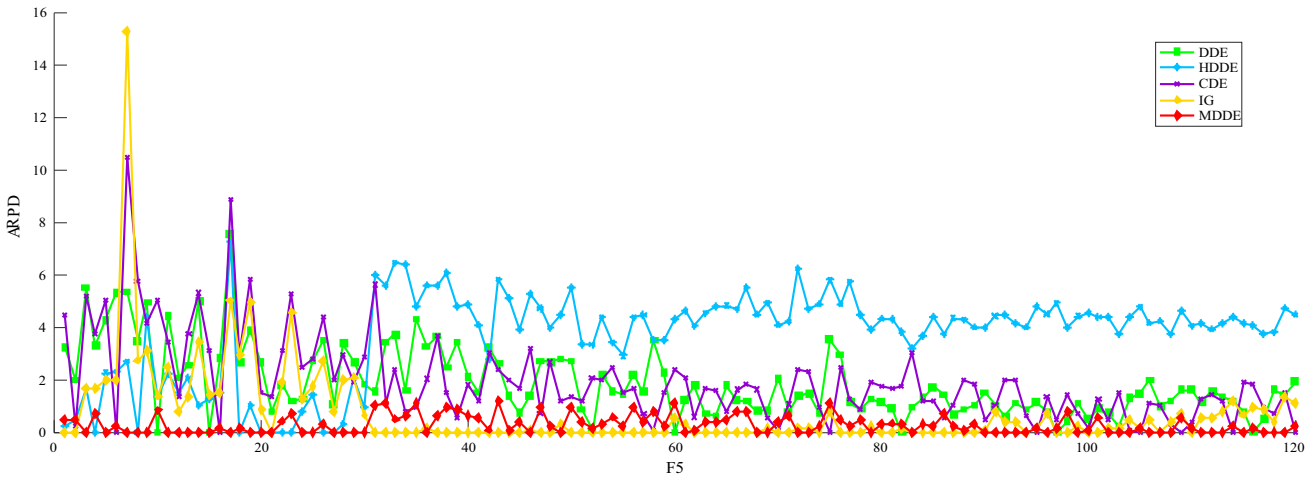

Fig. 16 ARPD of F=6

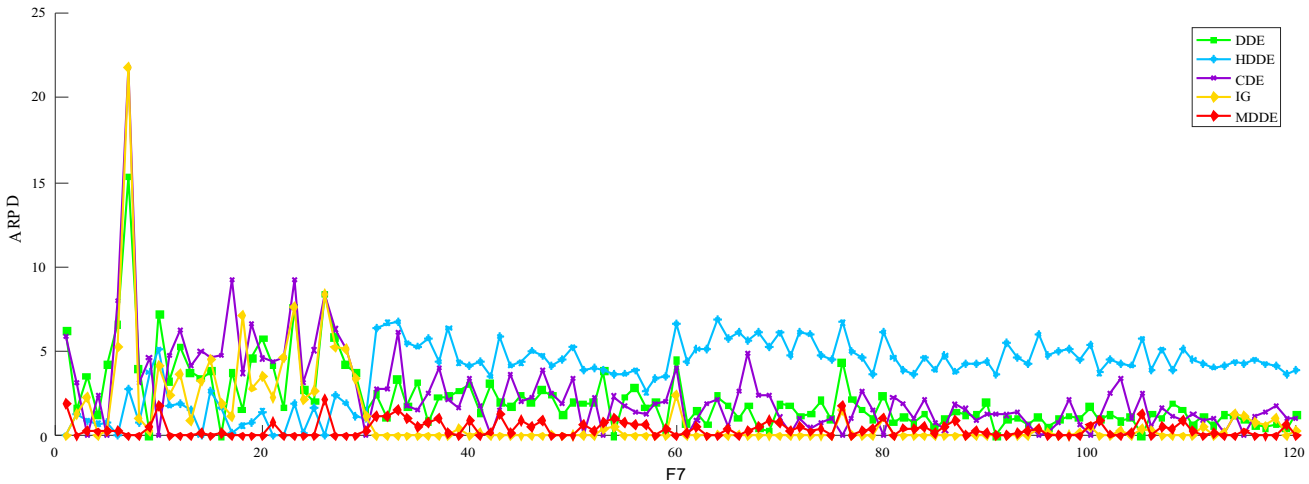

Fig. 17 ARPD of F=7

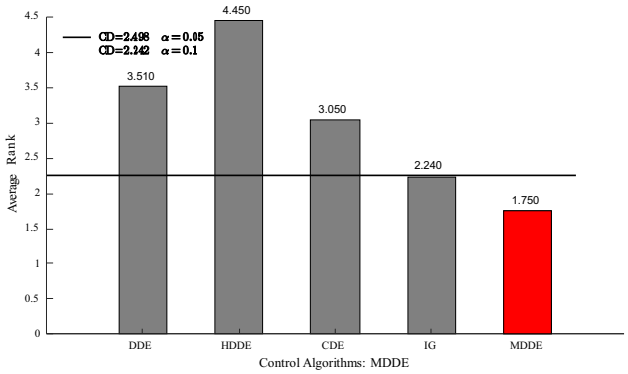

Fig. 18 Rankings for F=2

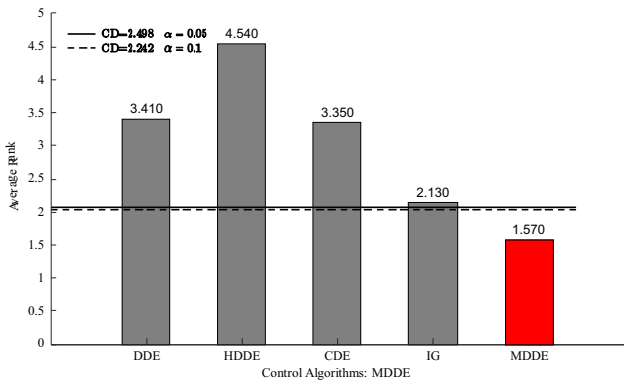

Fig. 19 Rankings for F=3

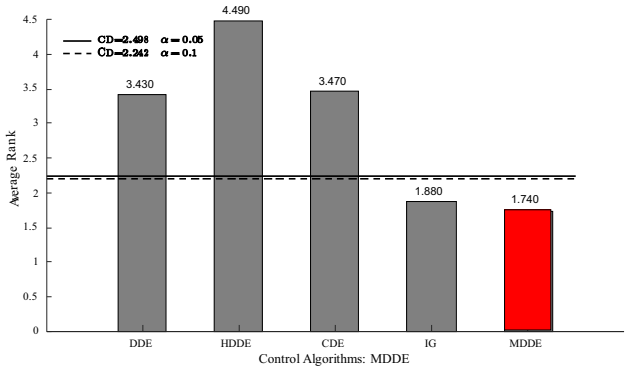

Fig. 20 Rankings for F=4

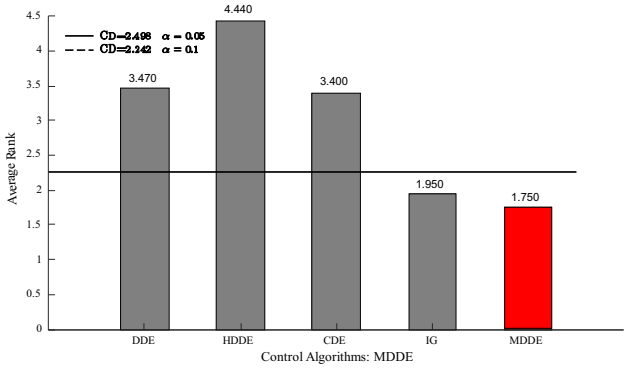

Fig. 21 Rankings for F=5

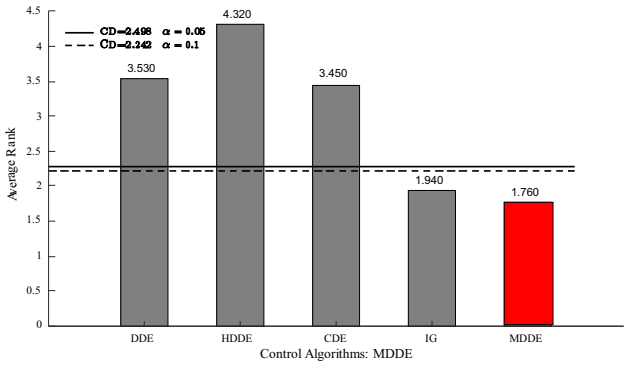

Fig. 22 Rankings for F=6

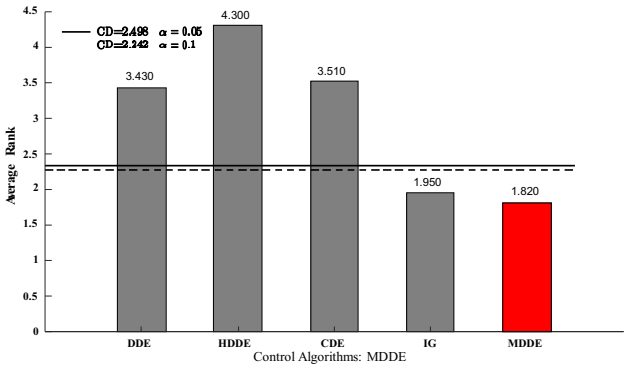

Fig. 23 Rankings for F=7

hood search strategy. Three sets of experiments are conducted to verify the contribution of these strategies. In the first experiment (N-DNEH), the improved NEH initialization method is removed. In the second experiment (N-Mutation), the discrete mutation and crossover strategy is removed. The adaptive neighborhood search strategy is removed in the third experiment (N-LocalSearch). The three algorithms are run on each instance 10 times and the terminal time is $30 * m * n$. Table 6 is obtained according to the calculation method of ARPD. In each instance, the smallest value corresponds to the best result. The value with the best result is highlighted in bold.

It can be seen from the Table 6 that the results of MDDE are better than MDDE(N-DNEH), so it can be concluded that the improved NEH initialization method contributes to the quality of the initial solution of the algorithm. The results of MDDE are also superior to the MDDE(N-Mutation) because the discrete variation strategies guarantee the feasibility of solutions. The results of the MDDE(N-LocalSearch) are poor, because the local search method effectively improves the exploitation of the algorithm and helps the algorithm to find the global optimal solution quickly.

Table 6 The computational results of all variants of MDDE

Instance	MDDE (N-DNEH)	MDDE (N-Mutation)	MDDE (N-LocalSearch)	MDDE
20*5	**0**	0.401	1.203	0.401
20*10	0.948	0.19	1.327	**0**
20*20	1.314	0.657	1.434	**0**
50*5	0.85	0.071	1.134	**0**
50*10	2.751	0.168	2.695	**0**
50*20	3.429	0.394	2.838	**0**
100*5	0.887	0.035	0.887	**0**
100*10	3.21	**0**	3.146	0.578
100*20	2.387	0.289	2.728	**0**
Average	1.753	0.245	1.932	**0.109**

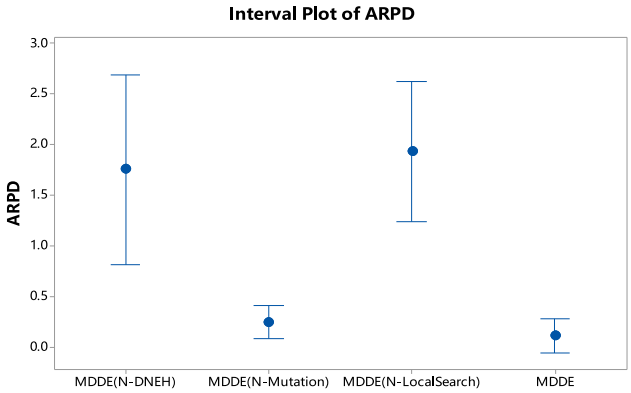

Fig. 24 The 95% confidence interval plot

The 95% confidence interval plot is shown in Fig. 24. As can be seen from the figure, the MDDE performs better than the other variables in all cases, indicating that each component has a significant impact on the proposed algorithm.

Flow shop scheduling problem has $n!$ possible sequences. It's expensive to list all the possible sequences. In the flow-shop scheduling problem, all the jobs must pass through all the machines in the same order, and the jobs with higher total machining time have higher priority than those with lower total machining time. The NEH method is a classical initialization method. The method can generate a very good sequence of jobs and solve the large-scale flow-shop scheduling problem well in both static and dynamic scheduling. The DNEH+T method is an improved version of NEH method, which is used to initialize the population. It can generate a good sequence of jobs and effectively solve the permutation flow shop scheduling problem.

DE algorithm has an efficient global optimization capability. The mutation strategy is an operation in the DE algorithm. The current individual and the differential vector are used to form the mutation strategy, in which the scaling factor is used to control the influence of the differential vector. This method can avoid falling into local optimum in the process of solving and improve the speed and accuracy.

In the process of iteration, multiple local optimal solutions are generated. The goal is to find a global optimal solution. Since a local optimal solution may not be optimal in all neighborhoods, the neighborhood needs to be changed to obtain the global optimal solution. In general, variable neighborhood search is a local search method, which can further optimize the current optimal solution.

In the process of solving, DNEH+T is used to generate a good initial solution, which is conducive to the further search of the algorithm. The mutation strategy in DE avoids the local optimum and increases the global searching ability of the algorithm. The variable neighborhood search strategy further optimizes the current optimal solution and enhances the local search ability of the algorithm. The combination of the three works balances the exploration and exploitation capabilities of the algorithm.

Conclusions and future research

In this paper, a memetic discrete differential evolutionary algorithm is proposed to solve the DPFSP problem. In the proposed MDDE, the neighborhood structure based on knowledge exchange mechanism strengthens the exploration and exploitation ability of MDDE. MDDE algorithm adopts knowledge-based integration strategy, which improves the search precision of the algorithm. MDDE algorithm is used to test 720 large instances. The results show that in most instances, MDDE performs better than other variants of DE. Wilcoxon's sign rank test and Friedman test are used to analyze the data. The results show that in the case of $C = 15$, MDDE is superior to the other four comparison algorithms in terms of solution diversity and convergence accuracy.

The future research is to develop the multi-target MDDE and design a certain self-learning method to improve the search ability of MDDE. The MDDE algorithm is used

to solve other distributed scheduling problems, such as the distributed dynamic scheduling problems. Furthermore, adopting MDDE to real-world optimization is also the main direction of future research.

Acknowledgements This work was financially supported by the National Key Research and Development Plan under grant number 2020YFB1713600 and the National Natural Science Foundation of China under grant numbers 62063021. It was also supported by the Lanzhou Science Bureau project (2018-rc-98), Public Welfare Project of Zhejiang Natural Science Foundation (LGJ19E050001), and Project of Zhejiang Natural Science Foundation (LQ20F020011), respectively.

References

1. Chen JF, Wang L, Peng ZP (2019) A collaborative optimization algorithm for energy-efficient multi-objective distributed no-idle flow-shop scheduling. Swarm Evol Comput 50:100557
2. Shao WS, Pi DC, Shao ZS (2018) A pareto-based estimation of distribution algorithm for solving multi objective distributed no-wait flow-shop scheduling problem with sequence-dependent setup time. IEEE Trans Autom Sci Eng 16:1344–1360
3. Sang HY, Pan QK, Li JQ, Wang P, Han YY, Gao KZ, Duan P (2019) Effective invasive weed optimization algorithms for distributed assembly permutation flow shop problem with total flowtime criterion. Swarm Evol Comput 44:64–73
4. Shao WS, Pi DC, Shao ZS (2017) Optimization of makespan for the distributed no-wait flow shop scheduling problem with iterated greedy algorithms. Knowl Based Syst 137:163–181
5. Ruiz R, Pan QK, Naderi B (2019) Iterated Greedy methods for the distributed permutation flow shop scheduling problem. Omega Int J Manag Sci 83:213–222
6. Zhao FQ, Qin S, Zhang Y, Ma WM, Zhang C, Song HB (2019) A hybrid biogeography-based optimization with variable neighborhood search mechanism for no-wait flow shop scheduling problem. Expert Syst Appl 126:321–339
7. Zhao FQ, Liu H, Zhang Y, Ma WM (2018) A discrete water wave optimization algorithm for no-wait flow shop scheduling problem. Expert Syst Appl 91:347–363
8. Zhao FQ, Xue FL, Zhang Y, Ma WM, Zhang C, Song HB (2019) A discrete gravitational search algorithm for the blocking flow shop problem with total flow time minimization. Appl Intell 49:3362–3382
9. Meng T, Pan QK, Wang L (2019) A distributed permutation flow shop scheduling problem with the customer order constraint. Knowl Based Syst 184:104894
10. Tasgetiren MF, Pan Q-K, Kizilay D, Velez-Gallego MC (2019) A variable block insertion heuristic for permutation flow shops with makespan criterion. 2017 IEEE congress on evolutionary computation, vol 12
11. Naderi B, Ruiz R (2010) The distributed permutation flow shop scheduling problem. Comput Oper Res 37:754–768
12. Fernandez-Viagas V, Perez-Gonzalez P, Framinan JM (2018) The distributed permutation flow shop to minimise the total flowtime. Comput Ind Eng 118:464–477
13. Pan QK, Gao L, Wang L, Liang J, Li XY (2019) Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flow shop problem. Expert Syst Appl 124:309–324
14. Wang JJ, Wang LA (2020) Knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop. IEEE Trans Syst Man Cybern Syst 50:1805–1819
15. Li WH, Li JQ, Gao KZ, Han YY, Niu B, Liu ZM, Sun Q (2019) Solving robotic distributed flow shop problem using an improved iterated greedy algorithm. Int J Adv Robot Syst 16:1729881419879819
16. Pan QK, Gao L, Li XY, Jose FM (2019) Effective constructive heuristics and meta-heuristics for the distributed assembly permutation flow shop scheduling problem. Appl Soft Comput 81:105492
17. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11:341–359
18. Meng Z, Pan J-S, Kong L (2018) Parameters with adaptive learning mechanism (PALM) for the enhancement of differential evolution. Knowl Based Syst 141:92–112
19. Meng Z, Pan J-S, Tseng K-K (2019) PaDE: an enhanced differential evolution algorithm with novel control parameter adaptation schemes for numerical optimization. Knowl Based Syst 168:80–99
20. Zhang J, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13:945–958
21. Brest J, Maucec MS, Boskovic B, (2017) IEEE (2017) Single objective real-parameter optimization: algorithm jSO. 2017 IEEE congress on evolutionary computation, pp 1311–1318
22. Brest J, Maucec MS, Boskovic B, (2016) IEEE (2016) iL-SHADE: improved L-SHADE algorithm for single objective real-parameter optimization. 2016 IEEE congress on evolutionary computation, pp 1188–1195
23. Tanabe R, Fukunaga AS, (2014) Improving the search performance of shade using linear population size reduction. 2014 IEEE congress on evolutionary computation. IEEE, pp 1658–1665
24. Mohamed AW, Hadi AA, Fattouh AM, Jambi KM, (2017) IEEE (2017) LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems, 2017 IEEE congress on evolutionary computation, pp 145–152
25. Zhao F, Qin S, Zhang Y, Ma W, Zhang C, Song H (2019) A two-stage differential biogeography-based optimization algorithm and its performance analysis. Expert Syst Appl 115:329–345
26. Zhao F, Xue F, Zhang Y, Ma W, Zhang C, Song H (2018) A hybrid algorithm based on self-adaptive gravitational search algorithm and differential evolution. Expert Syst Appl 113:515–530
27. Wu X, Liu X (2018) An Improved Differential Evolution Algorithm for Solving a Distributed Flexible Job Shop Scheduling Problem. In: Reveliotis S, Cappelleri D, Dimarogonas DV et al. (eds) 2018 IEEE 14th international conference on automation science and engineering, IEEE international conference on automation science and engineering. pp 968–973
28. Zhou B-H, Hu L-M, Zhong Z-Y (2018) A hybrid differential evolution algorithm with estimation of distribution algorithm for reentrant hybrid flow shop scheduling problem. Neural Comput Appl 30:193–209
29. Zhang G, Xing K, Cao F (2018) Discrete differential evolution algorithm for distributed blocking flow shop scheduling with makespan criterion. Eng Appl Artif Intell 76:96–107

30. Zhang G, Xing K (2019) Differential evolution metaheuristics for distributed limited-buffer flow shop scheduling with makespan criterion. Comput Oper Res 108:33–43
31. Wang L, Pan Q-K, Suganthan PN, Wang W-H, Wang Y-M (2010) A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems. Comput Oper Res 37:509–520
32. Montgomery DC (2006) Design and analysis of experiments, 9th edn. Wiley, New York
33. Taillard E (1990) Some efficient heuristic methods for the flow shop sequencing problem. Eur J Oper Res 47:65–74
34. Taillard E (1993) Benchmarks for basic scheduling problems. Eur J Oper Res 64:278–285
35. Stanovov V, Akhmedova S, Semenkin E, (2018) IEEE (2018) LSHADE algorithm with rank-based selective pressure strategy for solving CEC 2017 benchmark problems. 2018 IEEE congress on evolutionary computation, pp 1–8
36. Pan Q-K, Ruiz R (2014) An effective iterated greedy algorithm for the mixed no-idle permutation flow shop scheduling problem. Omega Int J Manag Sci 44:41–50
37. Shao Z, Pi D, Shao W, Yuan P (2019) An efficient discrete invasive weed optimization for blocking flow-shop scheduling problem. Eng Appl Artif Intell 78:124–141
38. Garcia S, Molina D, Lozano M, Herrera F (2009) A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization. J Heuristics 15:617–644