Engineering Applications of Artificial Intelligence 133 (2024) 108491

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

A hybrid estimation of distribution algorithm for solving assembly flexible job shop scheduling in a distributed environment

Baigang Du [a,b], Shuai Han [a,b], Jun Guo [a,b,*], Yibing Li [a,b]

[a] School of Mechanical and Electronic Engineering, Wuhan University of Technology, Wuhan, 430070, China
[b] Hubei Digital Manufacturing Key Laboratory, Wuhan, 430070, China

ARTICLE INFO

Keywords:
Distributed assembly flexible job shop scheduling problem
Estimation of distribution algorithm
Differential evolution
Variable neighborhood search

ABSTRACT

This paper proposes a novel distributed assembly flexible job shop scheduling problem (DAFJSP), which involves three stages: production stage, assembly stage, and delivery stage. The production stage is accomplished in a few flexible job shops, the assembly stage is accomplished in a few single-machine factories, and the delivery stage is to deliver the obtained products to the corresponding customers. To address the problem, a hybrid estimation of distribution algorithm based on differential evolution operator and variable neighborhood search (HEDA-DEV) is proposed with the goal of minimizing the total cost and tardiness. Firstly, a new multidimensional coding method is designed based on the features of the DAFJSP. Secondly, two mutation operators and the similarity coefficient based on the probability matrix are put forward to implement the dynamic mutation. Thirdly, five types of neighborhood structures satisfying cooperative search strategies are employed to adequately improve the local exploitation ability. Finally, the comparison experiment results suggest that the proposed HEDA-DEV has competitive performance compared to the selected efficient algorithms. Moreover, a real case study is used to demonstrate that HEDA-DEV is an effective method for solving DAFJSP.

1. Introduction

As economic globalization becomes popular, the conventional single-factory production mode is unable to meet the increasing market demand. So, more and more enterprises have begun to set up multiple factories in different places to improve their production efficiency (Naderi and Azab, 2014; Hsu et al., 2016; Hamzadayı, 2020; Behnamian and Fatemi Ghomi, 2016). Different from the single-factory production mode, a distributed manufacturing system must address two interrelated decisions, i.e., the allocation of jobs among factories and the scheduling of production within each factory (Zhang et al., 2018). It is much more complex than the conventional single-factory production mode because it has to deal with the problem of optimizing the scheduling across multiple factories simultaneously (Zhang and Xing, 2018).

Recently, the problem of distributed scheduling has attracted extensive attention from many scholars due to its theoretical significance and practical applications. Most researchers focus on the distributed scheduling optimization problem under various processing modes: distributed parallel machine scheduling system (Lei et al., 2021; Mönch and Shen, 2021), distributed job shop scheduling system (Şahman, 2021; Hsu et al., 2016), distributed flow-shop scheduling system (Cai et al., 2020; Huang et al., 2021; Rifai et al., 2016; Shao et al., 2020; Ying and Lin, 2018), distributed flexible manufacturing system (Chang and Liu, 2017; Lin et al., 2020; Wu et al., 2017; Guo et al., 2020). These distributed systems can be regarded as the extension of classical production systems. However, in practice, most practical products are assembled from multiple components which need to be processed first. And the actual manufacturing process includes two stages: production and assembly (Pan et al., 2019). Therefore, it is of significant relevance to study the distributed assembly scheduling problem (DASP).

The research of DASP has focused on the distributed assembly permutation flow-shop scheduling problem (DAPFSP) in recent years. The DAPFSP was first proposed by Hatami et al. (2013), combining distributed manufacturing with conventional PFSP. In the subsequent research, Hatami et al. (2015) extended the problem and combined the sequence-dependent setup times with DAPFSP. Subsequently, many scholars have put forward many effective algorithms to solve DAPFSP. Maria Gonzalez-Neira et al. (2017) designed a metaheuristic approach that integrated biased randomization and simulation techniques to solve DAPFSP with stochastic processing times. Zhang et al. (2022) designed a new three-dimensional probability model for the estimation of distribution algorithm to better handle the DAPFSP with an energy-saving

* Corresponding author. School of Mechanical and Electronic Engineering, Wuhan University of Technology, Wuhan, 430070, China.
E-mail address: junguo@whut.edu.cn (J. Guo).

https://doi.org/10.1016/j.engappai.2024.108491
Received 9 December 2022; Received in revised form 16 March 2024; Accepted 19 April 2024
Available online 29 April 2024

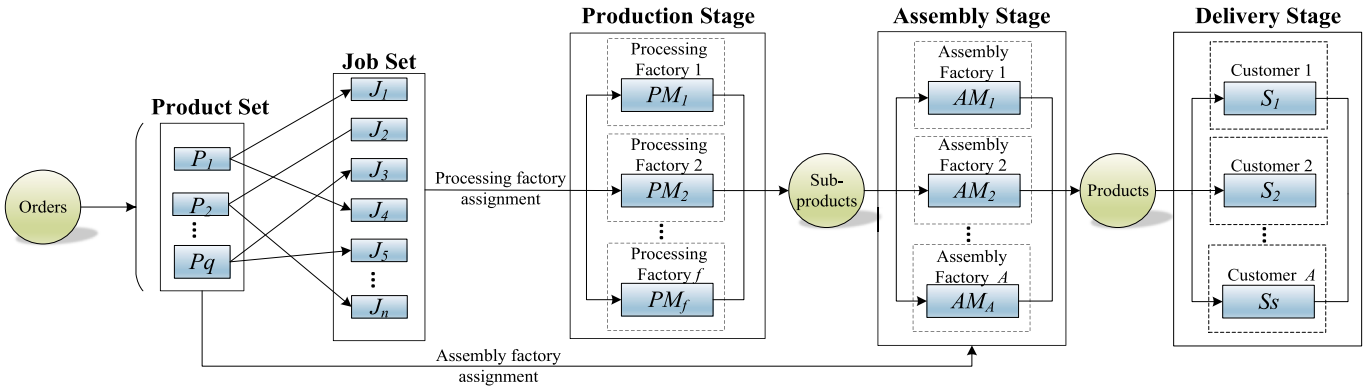

Fig. 1. The main flow chart of DAFJSP.

Table 1
Notations.

Indices	
r	Index of processing factory, $r = \{1,2, \ldots,f\}$
c	Index of assembly factory, $c = \{1,2, \ldots,a\}$
j, p	Index of machines, $j,p = \{1,2, \ldots,m\}$
i, h	Index of jobs, $i,h = \{1,2, \ldots,n\}$
k,g	Index of operation sequence, $k = \{1,2, \ldots, O_i\}, g = \{1,2, \ldots, O_h\}$
O_{ik}	Index of operation k of job i
b	Index of products, $b = \{1,2, \ldots,q\}$
d	Index of customers, $d = \{1,2, \ldots,s\}$
Parameters	
f	Total number of processing factory
a	Total number of the assembly factory
m	Total number of machines
n	Total number of jobs
O_i	Total number of operations of job i
q	Total number of products
s	Total number of customers
t^{JP}_{ikrj}	The processing time of O_{ik} on machine j of processing factory r
t^{JT}_{irc}	The transportation time of job i from processing factory r to assembly factory c
t^{PA}_{bc}	The assembly time of product b at assembly factory c
t^{PT}_{bdc}	The transportation time of product b from assembly factory c to customer d
C^{JP}_{ikrj}	The processing unit cost of O_{ik} on machine j of processing factory r
C^{JT}_{rc}	The transportation unit cost from processing factory r to assembly factory c
C^{PA}_{bc}	The assembly unit cost of product b in assembly factory c
C^{PT}_{cd}	The transportation unit cost from assembly factory c to customer d
W_{ib}	The assembly relationship of job i and product b which equals 1 if job i is a sub-product of product b and 0, otherwise
V_{bd}	The attribution relationship of product b and customer d which equals 1 if product b is ordered by customer d and 0, otherwise
D_b	The delivery date of product b
M	A large positive number
Decision variables	
S_{ikrj}	The start time of O_{ik} on machine k of the processing factory r
S_{bc}	The start time of product b in assembly factory c
T_b	The arrival time of product b to its corresponding customer
C	The total cost
TD	The total tardiness time
X_{ikrj}	A binary variable which equals 1 if O_{ik} is processed on machine j of processing factory r and 0, otherwise
H_{ikhg}	A binary variable which equals 1 if O_{ik} is scheduled later than O_{hg} and 0, otherwise
H_{bec}	A binary variable which equals 1 if product b is assembled later than product e in assembly factory c and 0, otherwise
Y_{ir}	A binary variable which equals 1 if job i is assigned to processing factory r and 0, otherwise
Z_{bc}	A binary variable which equals 1 if product b is assigned to assembly factory c and 0, otherwise

strategy. Compared to the DAPFSP, the assembly flexible job shop scheduling in a distributed environment (DAFJSP) is more complicated because it needs to deal with the arrangement of operations with variable processing sequences and the selection of multiple assembly factories simultaneously. However, few studies have been done on the DAFJSP. Wu et al. (2019) presented an improved differential evolution algorithm combined with the simulated annealing algorithm to address the DAFJSP. In the research of DAFJSP, few literatures consider the delivery of products to the customers after the completion of the assembly stage. However, in reality, we should not only consider producing the required products as quickly as possible but also consider delivering these products to customers with minimum delay and cost (Yang and Xu, 2021). Therefore, this paper proposes a novel DAFJSP considering the delivery of products to customers.

Given that FJSP is an NP-hard problem, DAFJSP, as an extension of FJSP is NP-hard. In the past few years, there has been an increasing trend of using meta-heuristic algorithms to address multi-objective optimal scheduling problems (He et al., 2019, 2021; Li et al., 2021; L. Sun et al., 2019; X. Sun et al., 2019). Among various meta-heuristic algorithms, the estimation of distribution algorithm (EDA) attracts our attention. It adopts the ideas of statistics and evolutionary algorithms for reference, updates the probability model by making up the dominant population in the previous generation, and uses the established probability model to guide the generation of the next generation (Hao et al., 2017). So far, many scholars have applied EDA to solve a variety of job shop scheduling problems and obtained good results, such as flow-shop scheduling (Wang et al., 2015), flexible job shop scheduling (Ge et al., 2016), etc. The differential evolution (DE) algorithm (Storn and Price, 1997) can disturb the population by information exchange between individual vectors and is widely applied for the advantages of easy implementation and high local search ability. Many scholars have applied DE to various shop scheduling areas. For example, He et al. (2022a) applied an improved adaptive DE algorithm to the energy-efficient open shop scheduling problem (EOSSP). Subsequently, they proposed an improved population-based DE algorithm to solve the EOSSP with multiple AGVs and deteriorated operations (He et al., 2022b). EDA has good global search ability but is poor at localized searching. It is easy to fall into local optimization in the later stages of the algorithm due to the reduction of population diversity. While DE can make up for the shortage of local searching ability of EDA. In addition, variable neighborhood search (VNS) is also used in conjunction with EDA because it can improve the quality of the solutions obtained by EDA (Liu et al., 2019). Inspired by these works, we propose a HEDA-DEV algorithm for the addressed problem. As far as we know, it is the first attempt to apply HEDA-DEV to DAFJSP. Firstly, we establish a mathematical model for the DAFJSP with the goal of minimizing the total cost and tardiness. Secondly, based on the problem characteristics, a novel multidimensional coding method is proposed. Thirdly, for enhancing the exploitation ability and operational efficiency of the algorithm, two mutation operators and a similarity coefficient are designed to achieve the dynamic mutation within the EDA framework. Then, considering both the characteristics of the problem and the two objectives, five neighborhood structures that satisfy the cooperative search strategies are designed. Finally, comparison experiments between HEDA-DEV and the selected efficient

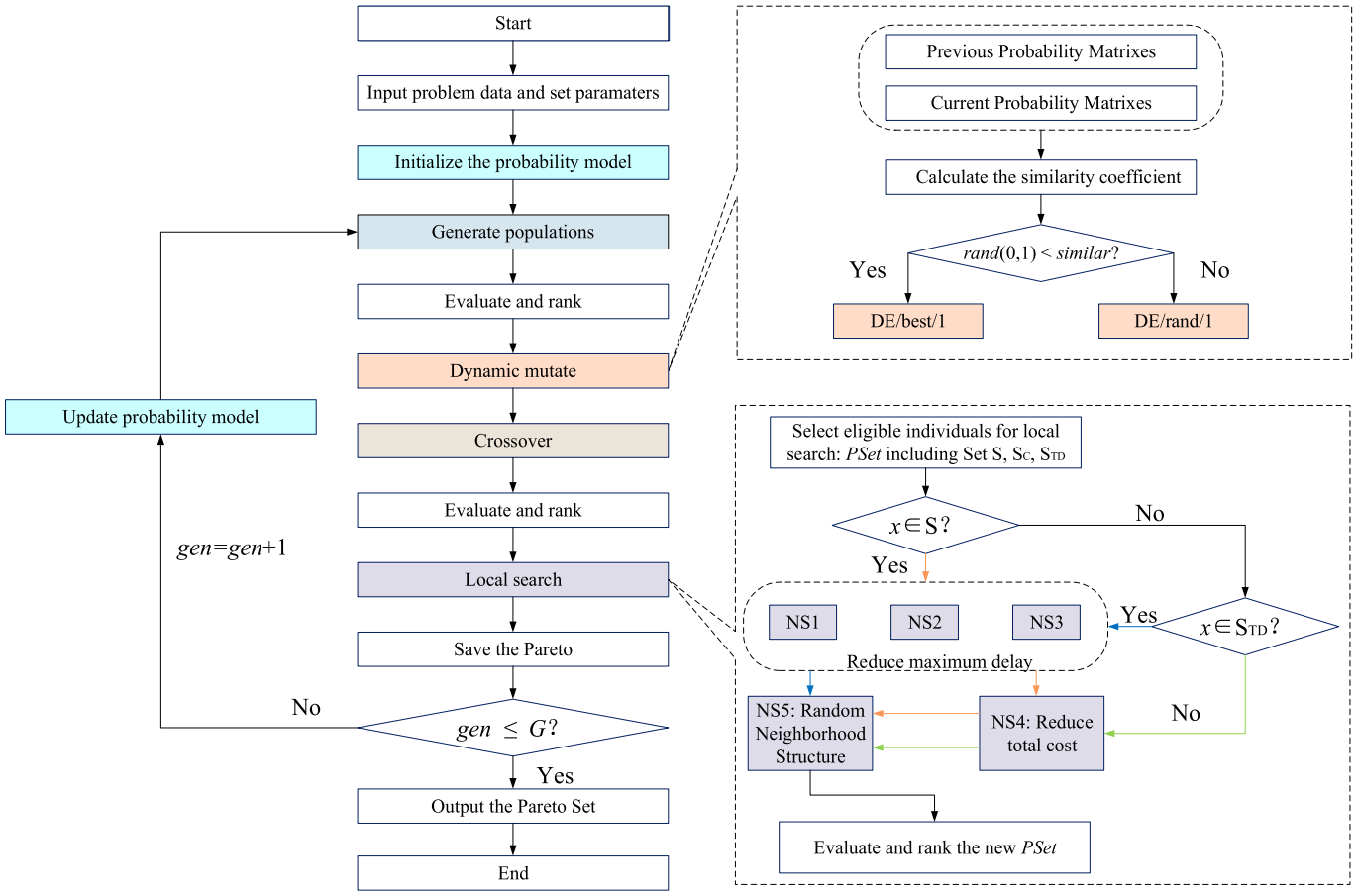

Fig. 2. The framework of the proposed HEDA-DEV.

algorithms are carried out to verify the efficiency of HEDA-DEV.

The rest of the paper is structured as follows: The DAFJSP is described in Section 2. Section 3 proposes a HEDA-DEV to solve the DAFJSP. Section 4 gives the algorithm comparison experiment and the result analysis. Finally, the study is summarized, and future research is prospected in Section 5.

2. The formulation of DAFJSP

2.1. Problem description

The main flow chart of DAFJSP is shown in Fig. 1. It consists of the following three stages: production stage, assembly stage, and delivery stage.

At first, the manufacturer received orders from s customers $\{S_1, S_2, \ldots, S_s\}$ with a total of q products $\{P_1, P_2, \ldots, P_q\}$. The assembly structure of the products is the single-layer tree structure and they are assembled from n jobs $\{J_1, J_2, \ldots, J_n\}$. All jobs need to be assigned to each processing factory before starting processing. In the production stage, each processing factory is a regular FJSP. A job J_i consists of a series of operations $\{O_{i1}, O_{i2}, \ldots, O_{ik}\}$ that need to be processed in a certain sequence on the available machines. In the assembly stage, each assembly factory can be considered as a single-machine factory. A product can only be assembled when all of its jobs are finished in the production factory and transported to the assembly factory. In the delivery stage, the finished products will be delivered to the corresponding customers.

In addition, according to the characteristics of DAFJSP, the following assumptions should be made.

1) Each job/product can be assigned to only one available processing/assembly factory.
2) Job/product shall not be transferred between factories during processing/assembly.
3) Once the job starts processing, it cannot be interrupted until the end of the process.
4) One job can only be processed on one machine simultaneously, and one machine can only process one job at a time.
5) Buffer capacity is infinite, and machine start time is not considered.

The notations employed in the paper are displayed in Table 1.

2.2. Mathematical formulation

The mathematical model of DAFJSP is presented as follows.

$$\min \quad C = \sum_{i=1}^{n} \sum_{k=1}^{O_i} \sum_{r=1}^{f} \sum_{j=1}^{m} t_{ikrj}^{JP} \times C_{ikrj}^{JP} \times X_{ikrj} + \sum_{r=1}^{f} \sum_{i=1}^{n} \sum_{c=1}^{a} \sum_{b=1}^{q} t_{irc}^{JT} \times C_{rc}^{JT} \times Y_{ir} \times Z_{bc} \times W_{ib} + \sum_{b=1}^{q} \sum_{c=1}^{a} t_{bc}^{PA} \times C_{bc}^{PA} \times Z_{bc} + \sum_{b=1}^{q} \sum_{d=1}^{s} \sum_{c=1}^{a} t_{bdc}^{PT} \times C_{cd}^{PT} \times Z_{bc} \times V_{bd} \tag{1}$$

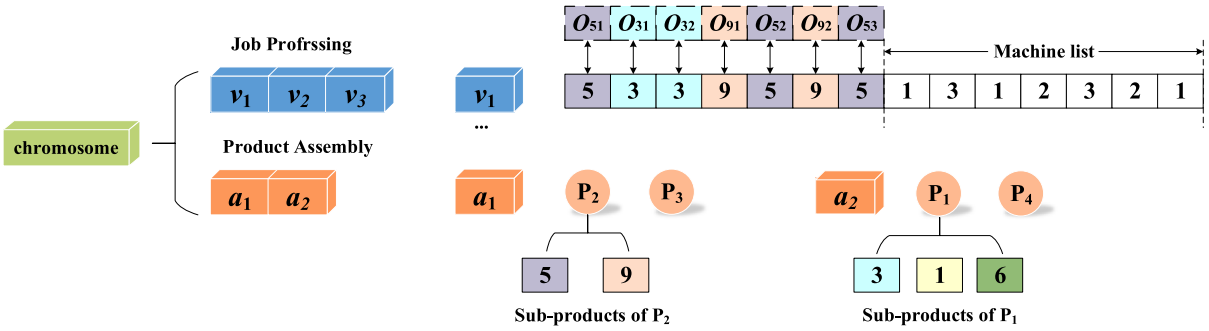

Fig. 3. An example of chromosome.

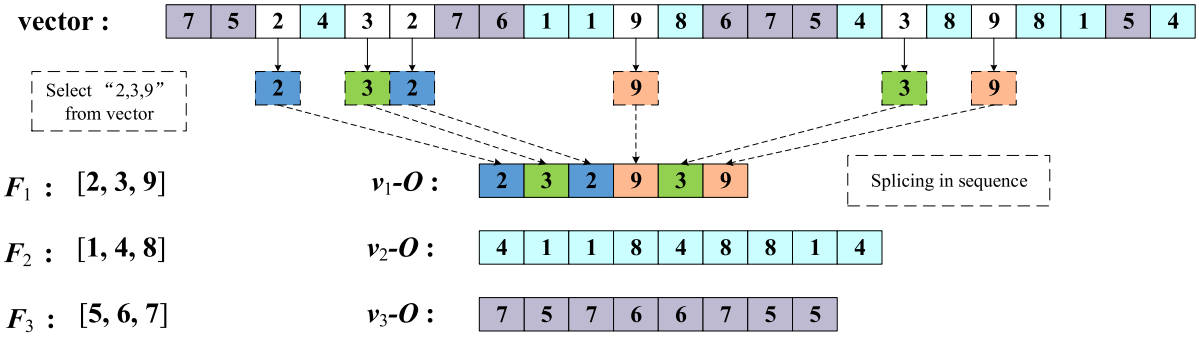

Fig. 4. An example of factory internal scheduling.

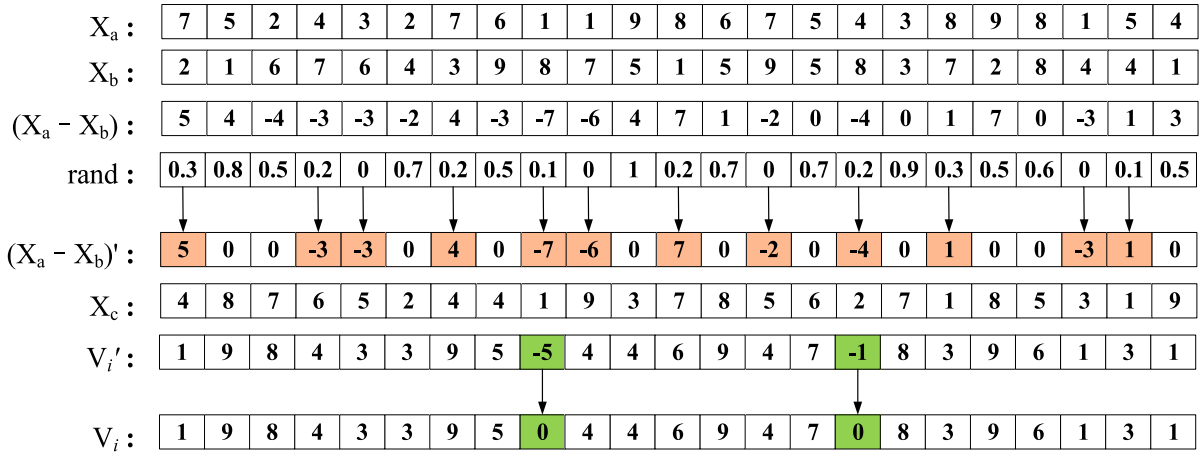

Fig. 5. An example of DE/rand/1.

$$\min \quad TD = \sum_{b=1}^{q} \max\{T_b - D_b, 0\} \tag{2}$$

s.t

$$S_{ikrj} - S_{i(k-1)rp} - t^{JP}_{i(k-1)rp} \geq M \times \left(X_{ikrj} + X_{i(k-1)rp} - 2\right), \forall i, k > 1, j, p, r \tag{3}$$

$$S_{ikrj} - S_{hgrj} - t^{JP}_{hgrj} \geq M \times \left(X_{ikrj} + X_{hgrj} + H_{ikhg} - 3\right), \forall k \in [1, O_i], g \in [1, O_h], i \neq h, j, r \tag{4}$$

$$H_{ikhg} + H_{hgik} \geq 1 + M \times \left(X_{ikrj} + X_{hgrj} - 2\right), \forall k \in [1, O_i], g \in [1, O_h], i \neq h, j, r \tag{5}$$

$$H_{ikhg} + H_{hgik} \leq 1 - M \times \left(X_{ikrj} + X_{hgrj} - 2\right), \forall k \in [1, O_i], g \in [1, O_h], i \neq h, j, r \tag{6}$$

$$\sum_{r=1}^{f} Y_{ir} = 1, \forall i \tag{7}$$

$$\sum_{c=1}^{q} Z_{bc} = 1, \forall \mathrm{b} \tag{8}$$

$$\sum_{r=1}^{f} \sum_{j=1}^{m} X_{ikrj} = 1, \forall k \in [1, O_i], i \tag{9}$$

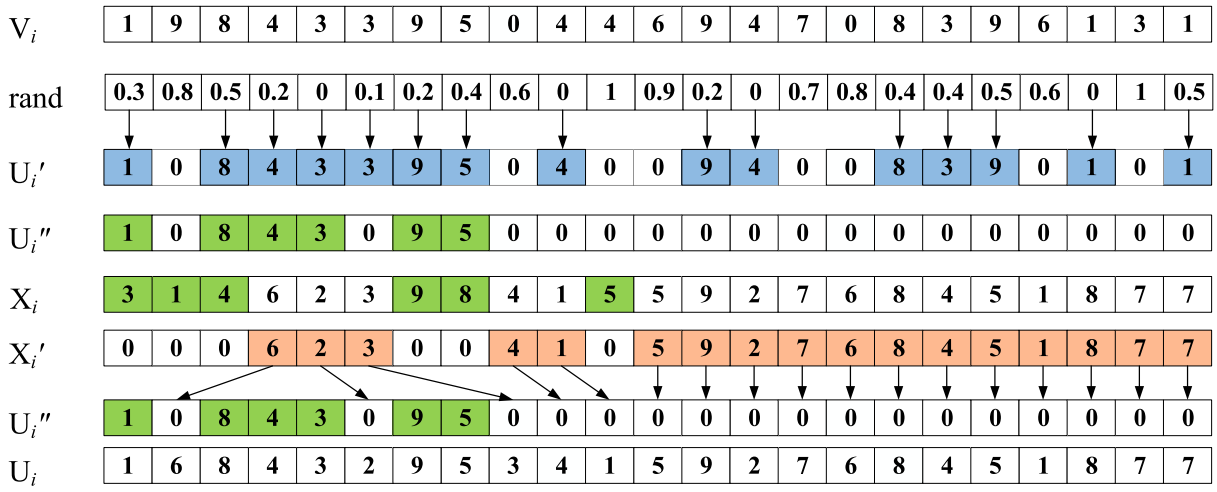

Fig. 6. An example of crossover.

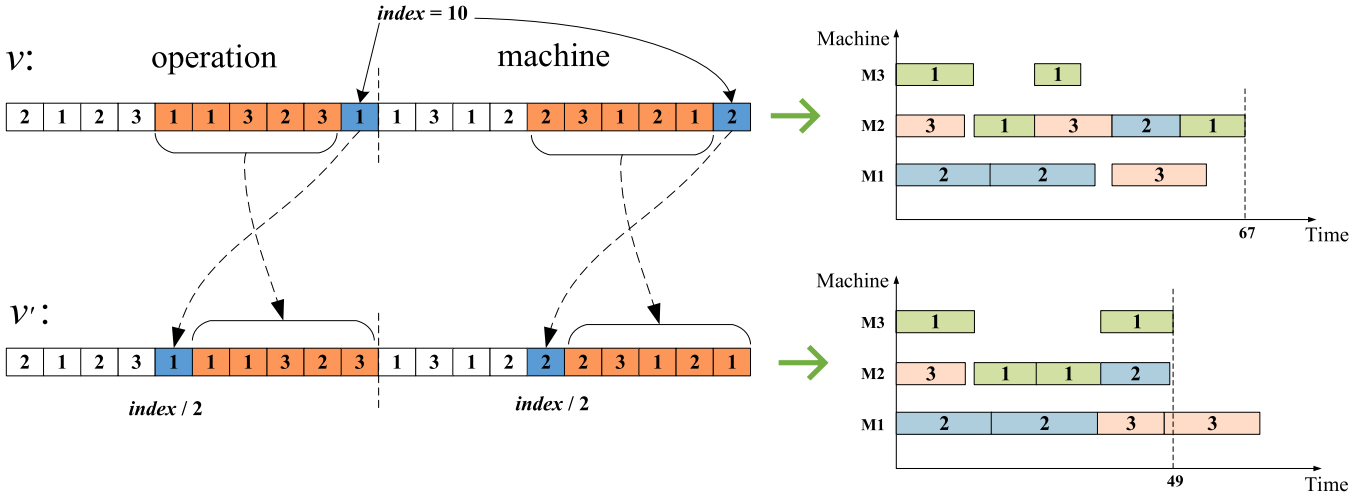

Fig. 7. Neighborhood structure 1.

Table 2
Details of the example.

Jobs	Job1				Job2			Job3		
operation	O_{11}	O_{12}	O_{13}	O_{14}	O_{21}	O_{22}	O_{23}	O_{31}	O_{32}	O_{33}
M1	12	10	9	10	18	21	10	15	11	18
M2	14	9	11	13	16	17	12	12	15	14
M3	15	12	7	14	20	16	14	10	19	20

$$S_{bc} - S_{iO_irj} - t^{JP}_{iO_irj} - t^{JT}_{irc} \geq M \times \left(Y_{ir} + Z_{bc} + W_{ib} + X_{iO_irj} - 4\right), \forall b, c, i, r, j \tag{10}$$

$$S_{bc} - S_{ec} - t^{PA}_{ec} \geq M \times (Z_{bc} + Z_{ec} + H_{bec} - 3), \forall b, e, c, b \neq e \tag{11}$$

$$H_{bec} + H_{ebc} \geq 1 + M \times (Z_{bc} + Z_{ec} - 2), \forall b, e, c, b \neq e \tag{12}$$

$$H_{bec} + H_{ebc} \leq 1 - M \times (Z_{bc} + Z_{ec} - 2), \forall b, e, c, b \neq e \tag{13}$$

$$T_b = \sum_{c=1}^{a} \left(S_{bc} + t^{PA}_{bc}\right) \times Z_{bc} + \sum_{d=1}^{s} \sum_{c=1}^{a} t^{PT}_{bdc} \times Z_{bc} \times V_{bd}, \forall b \tag{14}$$

Equation (1) and equation (2) are two objective functions of the DAFJSP. Equation (1) is designed to minimize the total cost, including the total processing cost of all jobs, the cost of transportation between the processing and assembly factories, the total assembly cost of all products, and the cost of transportation between the assembly factories and the customers. Equation (2) is formulated to minimize the total tardiness. Equation (3) ensures the job can only be processed according to its process route. Equation (4) guarantees that a machine can only process one job simultaneously. Equations (5) and (6) limit the processing sequence of the different jobs. Equations (7) and (8) limit one job/product can only be assigned to one processing/assembly factory. Equation (9) limits a job to be processed on only one machine at a time. Equation (10) guarantees that the assembly start time for each product is behind the arrival time of all jobs for this product at the assembly factory. Equation (11) guarantees that an assembly factory can only assemble one product simultaneously. Equations (12) and (13) limit the assembly sequence of the different products. Equation (14) calculates the arrival time of the product.

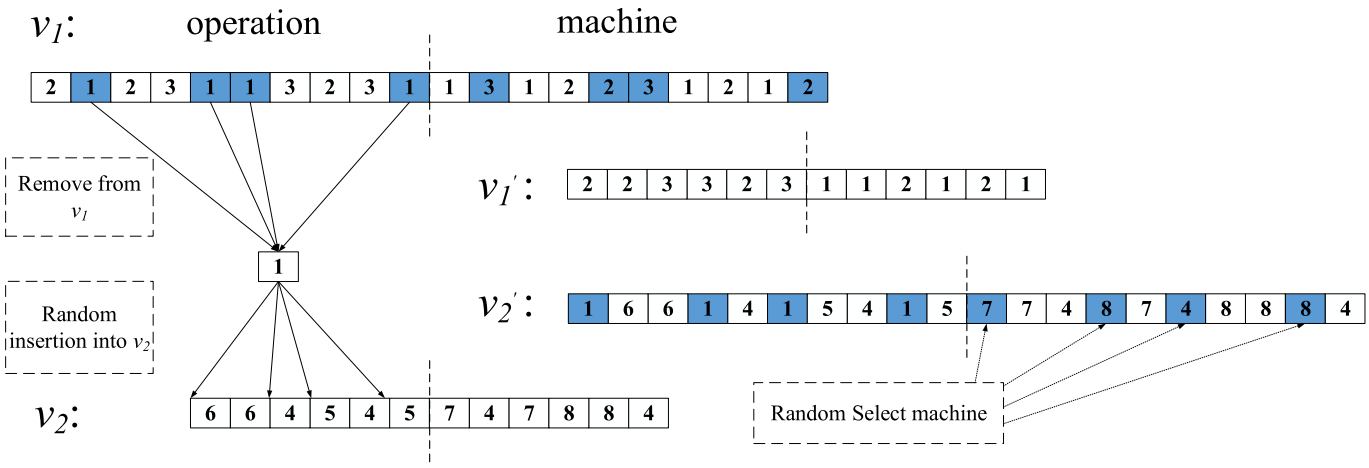

Fig. 8. Neighborhood structure 2.

Table 3
The parameters of Test Set 1.

Instance	Scale	O_i	t_{ikrj}^{JP}	t_{bc}^{PA}	$t_{irc}^{JT}/t_{bdc}^{PT}$	C_{rc}^{JT}/C_{cd}^{PT}	$C_{ikrj}^{JP}/C_{bc}^{PA}$	D_b
MK01	10_6_3_2_3_2	[5, 7]	[1, 7]	[6, 12]	[12, 20]	[1, 3]	[5, 10]	[30, 45]
MK02	10_6_3_2_3_2	[5, 7]	[1, 7]	[2, 15]	[12, 20]	[1, 3]	[5, 10]	[45, 60]
MK03	15_8_3_2_4_2	[10, 10]	[1, 20]	[5, 20]	[5, 20]	[1, 3]	[5, 10]	[210, 270]
MK04	15_10_4_3_3_3	[3, 10]	[1, 10]	[5, 20]	[5, 20]	[1, 3]	[5, 10]	[80, 120]
MK05	15_10_4_3_3_3	[5, 10]	[5, 10]	[5, 20]	[5, 20]	[1, 3]	[5, 10]	[100, 150]
MK06	10_10_4_3_3_3	[15, 15]	[1, 10]	[5, 20]	[5, 20]	[1, 3]	[5, 10]	[80, 150]
MK07	20_10_4_3_5_3	[5, 5]	[1, 20]	[15, 25]	[5, 20]	[1, 3]	[5, 10]	[120, 180]
MK08	20_12_4_3_5_3	[5, 10]	[5, 20]	[15, 25]	[5, 20]	[1, 3]	[5, 10]	[200, 250]
MK09	20_10_4_3_5_3	[10, 15]	[5, 20]	[15, 25]	[5, 20]	[1, 3]	[5, 10]	[300, 400]
MK10	20_12_4_3_5_3	[10, 15]	[5, 20]	[15, 30]	[5, 20]	[1, 3]	[5, 10]	[300, 350]
MK11	30_10_4_3_7_3	[5, 8]	[10, 30]	[5, 20]	[5, 20]	[1, 3]	[5, 10]	[400, 500]
MK12	30_15_4_3_7_3	[5, 10]	[10, 30]	[5, 20]	[5, 20]	[1, 3]	[5, 10]	[300, 400]
MK13	30_15_4_3_7_3	[5, 10]	[10, 30]	[5, 25]	[5, 20]	[1, 3]	[5, 10]	[350, 450]
MK14	30_15_4_3_7_3	[8, 12]	[10, 30]	[5, 20]	[5, 20]	[1, 3]	[5, 10]	[400, 500]
MK15	30_15_4_3_5_4	[8, 12]	[10, 30]	[17, 33]	[5, 20]	[1, 3]	[5, 10]	[450, 550]

Table 4
The parameters of Test Set 2.

Instance	Scale	O_i	t_{ikrj}^{JP}	t_{bc}^{PA}	$t_{irc}^{JT}/t_{bdc}^{PT}$	C_{rc}^{JT}/C_{cd}^{PT}	$C_{ikrj}^{JP}/C_{bc}^{PA}$	D_b
LI01	100_15_4_3_15_4	[8, 12]	[10, 30]	[30, 50]	[10, 30]	[1, 3]	[5, 10]	[1500, 2000]
LI02	100_20_4_3_15_4	[8, 12]	[10, 30]	[30, 50]	[10, 30]	[1, 3]	[5, 10]	[1000, 2000]
LI03	100_20_4_3_20_4	[8, 12]	[10, 30]	[30, 50]	[10, 30]	[1, 3]	[5, 10]	[1000, 2000]
LI04	300_20_4_3_20_4	[8, 12]	[10, 30]	[30, 50]	[10, 30]	[1, 3]	[5, 10]	[3000, 4000]
LI05	300_25_4_3_30_4	[8, 12]	[10, 30]	[30, 50]	[10, 30]	[1, 3]	[5, 10]	[3000, 4000]
LI06	300_30_4_3_40_4	[8, 12]	[10, 30]	[30, 50]	[10, 30]	[1, 3]	[5, 10]	[2500, 3500]
LI07	500_30_4_3_40_4	[8, 12]	[10, 30]	[30, 50]	[10, 30]	[1, 3]	[5, 10]	[4000, 5000]
LI08	500_30_5_4_40_4	[8, 12]	[10, 30]	[30, 50]	[10, 30]	[1, 3]	[5, 10]	[3500, 4000]
LI09	500_40_6_4_45_4	[8, 12]	[10, 30]	[30, 50]	[10, 30]	[1, 3]	[5, 10]	[3000, 4000]
LI10	500_50_5_4_50_4	[8, 12]	[10, 30]	[30, 50]	[10, 30]	[1, 3]	[5, 10]	[2500, 3000]

Table 5
The parameter value and their levels.

parameters	Parameter level				
	1	2	3	4	5
Popsize	50	100	150	200	250
α	0.1	0.2	0.3	0.4	0.5
β	0.1	0.2	0.3	0.4	0.5
μ	0.1	0.2	0.3	0.4	0.5
F	0.1	0.3	0.5	0.7	0.9
CR	0.1	0.3	0.5	0.7	0.9

3. Proposed HEDA-DEV for DAFJSP

3.1. Framework of HEDA-DEV

The framework of the HEDA-DEV is shown in Fig. 2. The detailed steps of the HEDA-DEV are as follows.

Step 1. Input the problem data, such as the number of jobs and factories, processing and assembly time, processing and assembly cost, etc. Set the parameters of the DAFJSP, such as the size of population, the learning rate of probability matrices, the mutation and crossover probability, etc. The details are in Section 4.1.

Step 2. Initialize the probability model. It is initialized with uniform

Table 6
Orthogonal array and Response values.

Experiments	Parameters and factor levels						Response	
	Popsize	α	β	μ	*F*	*CR*	MK07	LI05
1	1	1	1	1	1	1	0.1684	0.2161
2	1	2	2	2	2	2	0.1731	0.1678
3	1	3	3	3	3	3	0.1608	0.1645
4	1	4	4	4	4	4	0.1767	0.1707
5	1	5	5	5	5	5	0.1890	0.1626
6	2	1	2	3	4	5	0.1642	0.1516
7	2	2	3	4	5	1	0.1578	0.0827
8	2	3	4	5	1	2	0.1544	0.0682
9	2	4	5	1	2	3	0.1485	0.2677
10	2	5	1	2	3	4	0.1394	0.1455
11	3	1	3	5	2	4	0.1502	0.1788
12	3	2	4	1	3	5	0.1186	0.0744
13	3	3	5	2	4	1	0.1383	0.0693
14	3	4	1	3	5	2	0.1386	0.0542
15	3	5	2	4	1	3	0.1339	0.0468
16	4	1	4	2	5	3	0.1318	0.1082
17	4	2	5	3	1	4	0.1297	0.0574
18	4	3	1	4	2	5	0.1306	0.0493
19	4	4	2	5	3	1	0.1374	0.0481
20	4	5	3	1	4	2	0.1081	0.1195
21	5	1	5	4	3	2	0.1375	0.1365
22	5	2	1	5	4	3	0.1209	0.0640
23	5	3	2	1	5	4	0.1000	0.0543
24	5	4	3	2	1	5	0.1073	0.0390
25	5	5	4	3	2	1	0.1187	0.0373

distribution. The details are in Section 3.3.1.

Step 3. Generate populations by sampling probability matrices and the generation method. After sampling the probability matrix, we can get the factory assignment scheme of the jobs and products, and then the detailed operation scheduling sequence in the factory can be obtained through the population generation method. For details, see Sections 3.3 and 3.4.

Step 4. Evaluate and rank. Evaluate the total cost and tardiness of the population, and record the rank of each individual after non-dominance sorting.

Step 5. Dynamic mutate. Calculate the similarity coefficient of the probability matrices between the two successive generations. Infer the stage of the algorithm through the similarity coefficient to choose different mutation operations. The details are shown in Section 3.4.

Step 6. Crossover. The new individual can be obtained by exchanging information between the mutated individual and the original individual. The details are shown in Section 3.5.

Step 7. Evaluate and rank. Make a new non-dominated sorting of the population after the dynamic mutate and crossover.

Step 8. Local search. Four targeted neighborhood operators and a random operator are designed, which can work together according to a search strategy. Repeat the procedure until the stop condition is met.

Step 9. Save the Pareto. If $gen \leq G$, go to step 10. Otherwise, set $gen = gen + 1$ and update the probability matrices through the superior individuals. Then, return to the step 3.

Step 10. Output the Pareto Set.

3.2. Multi-dimensional coding method

The DAFJSP contains three sub-problems, including factory assignment, machine selection, and operations scheduling. Therefore, we design a new multi-dimensional coding method, which is composed of two parts according to the manufacturing stage. The first part is job processing, and the second part is product assembly. Assuming that there are nine jobs, three processing factories, and two assembly factories, a feasible solution is shown in Fig. 3.

In Fig. 3, v_1, v_2, and v_3 represent the sequence of operations and machines in processing factories 1, 2, and 3 respectively. In the first part

Table 7
Means response table for each parameter.

Factor level	MK07						LI05					
	Popsize	α	β	μ	*F*	*CR*	*Popsize*	α	β	μ	*F*	*CR*
1	0.1736	0.1504	0.1396	0.1287	0.1387	0.1441	0.1763	0.1582	0.1058	0.1464	0.0855	0.0907
2	0.1529	0.1400	0.1417	0.1380	0.1442	0.1423	0.1431	0.0893	0.0937	0.1060	0.1402	0.1092
3	0.1359	0.1368	0.1368	0.1424	0.1387	0.1392	0.0847	0.0811	0.1169	0.0930	0.1138	0.1302
4	0.1275	0.1417	0.1400	0.1473	0.1416	0.1392	0.0765	0.1159	0.0918	0.0972	0.1150	0.1213
5	0.1169	0.1378	0.1486	0.1504	0.1434	0.1419	0.0662	0.1023	0.1387	0.1043	0.0924	0.0954
Delta	0.0567	0.0136	0.0118	0.0217	0.0055	0.0049	0.1101	0.0771	0.0469	0.0534	0.0547	0.0395
Rank	1	3	4	2	5	6	1	2	5	4	3	6

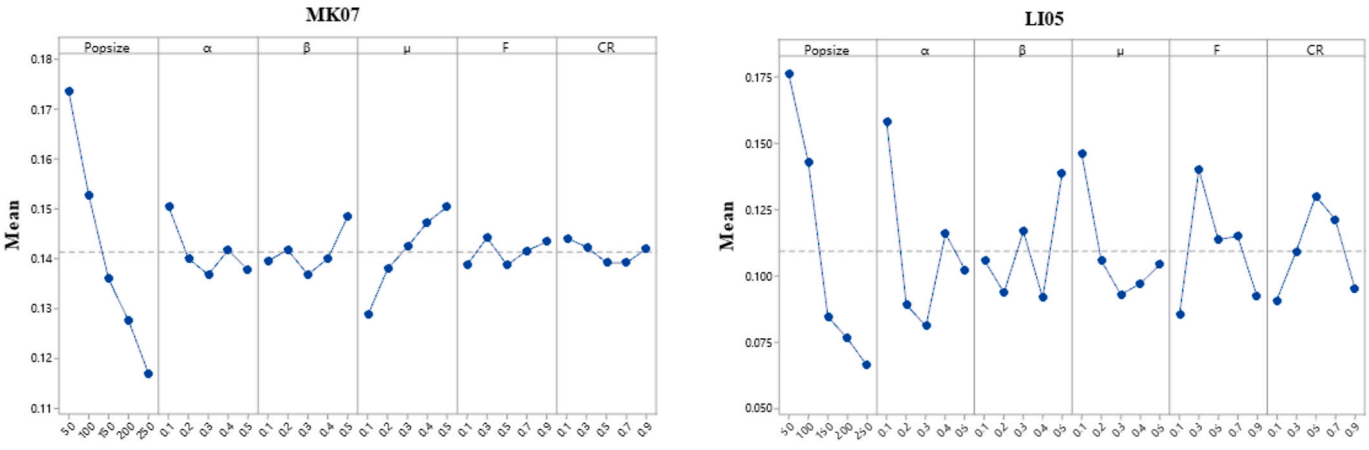

Fig. 9. Factor level trend for MK07 and LI05.

Table 8
Parameter setting of algorithms.

Algorithms	Test Set 1							Test Set 2						
	Popsize	α	β	μ	*F*	*CR*	*T*	*Popsize*	α	β	μ	*F*	*CR*	*T*
HEDA-DEV	250	0.3	0.3	0.1	0.1	0.5	–	250	0.3	0.4	0.3	0.1	0.1	–
EDA	200	0.3	0.4	0.4	–	–	–	200	0.3	0.2	0.3	–	–	–
DE	200	–	–	–	0.1	0.9	–	250	–	–	–	0.3	0.1	–
ABC	250	–	–	–	–	–	–	200	–	–	–	–	–	–
NSGA-II	250	–	–	–	0.3	0.1	–	250	–	–	–	0.3	0.3	–
MOEA/D	200	–	–	–	0.5	0.1	50	200	–	–	–	0.3	0.1	50

Table 9
Pareto solution for CPLEX and HEDA-DEV.

Pareto Solution	CPLEX		HEDA-DEV	
	Total Cost	Total Tardiness	Total Cost	Total Tardiness
1	629	54	629	54
2	638	53	638	53
3	642	51	656	51

of v_1, the number corresponds to the job index, i.e., jobs 3, 5, and 9 are allocated to factory 1. The occurrence sequence of the job represents the sequence of operations. For example, the first "5" refers to the first operation of job 5 (O_{51}), and the second "5" refers to the O_{52}. The second part of v_1 is the list of machines, which corresponds to the operations in the first part one by one. For example, the first "1" of the Machine list corresponds to O_{51} in the first part, indicating that O_{51} is processed on machine M_1. In addition, a_1 and a_2 represent the products allocated to assembly factories 1 and 2, respectively. The products "P_2" and "P_3" are allocated to assembly factory 1, and products "1" and "4" are allocated to assembly factory 2. Take products "P_1" and "P_2" for example, the sub-products of "P_1" are jobs 5 and 9, and the sub-products of "P_2" are jobs 3, 1, and 6.

In addition, it is very hard to guarantee a high quality of the initial population by random initialization. Therefore, we design two heuristic rules which act simultaneously to improve the quality of the initial population. Meanwhile, in consideration of maintaining the diversity of the population, the following rules affect only 50% of randomly selected individuals.

(1) Machine adjustment rule

According to the optimization goal, this paper generates individuals equally according to two strategies: for 50% of individuals, traverse the list of available machines in the factory, and give priority to the shortest processing time. If the machine is not the only one, choose the machine with the lowest processing cost. For the remaining 50% of individuals, the lowest processing cost is given priority, and the machine with the shortest processing time should be chosen when the machine is not unique.

(2) Factory assignment adjustment rule

Selecting the nearest assembly factory closest to the customer will reduce the transportation time and the total tardiness. After the product-assembly factory assignment is obtained by sampling probability matrix $\boldsymbol{\sigma}$, it is adjusted and assigned to the nearest assembly factory according to the customer's location where the product is to be sent.

3.3. Probability model and updating mechanism

The probability model and its updating mechanism are the core steps of EDA (Wang et al., 2013a). Two matrixes are used as probability model in this paper: the probability matrix $\boldsymbol{\rho}$ which represents the matching

Table 10
The IGD results of HEDA-DEV and its two variants on each instance.

Instance	HEDA-DEV			HEDA-DE			HEDA-VNS		
	Avg	SD	Med	Avg	SD	Med	Avg	SD	Med
MK01	**0.1209**	0.0304	**0.1129**	0.2202	0.0292	0.2214	0.1510	0.0318	0.1448
MK02	**0.1761**	0.0403	**0.1806**	0.2946	0.0531	0.3028	0.2284	0.0357	0.2294
MK03	**0.1446**	0.0336	**0.1423**	0.2394	0.0427	0.2366	0.1700	0.0283	0.1657
MK04	**0.1623**	0.0736	**0.1541**	0.3035	0.0535	0.3166	0.2289	0.0685	0.2211
MK05	**0.0873**	0.0250	**0.0790**	0.1553	0.0330	0.1474	0.1059	0.0270	0.0988
MK06	**0.1334**	0.0294	**0.1360**	0.2557	0.0507	0.2465	0.1673	0.0344	0.1642
MK07	**0.0872**	0.0175	**0.0826**	0.1859	0.0364	0.1891	0.1139	0.0275	0.1099
MK08	**0.1226**	0.0276	**0.1211**	0.2010	0.0257	0.1997	0.1492	0.0300	0.1513
MK09	**0.1231**	0.0215	**0.1225**	0.1993	0.0469	0.1983	0.1601	0.0299	0.1653
MK10	**0.1058**	0.0195	**0.1032**	0.2092	0.0478	0.2081	0.1503	0.0329	0.1455
MK11	**0.1247**	0.0322	**0.1277**	0.2005	0.0261	0.1954	0.1541	0.0356	0.1543
MK12	**0.0990**	0.0234	**0.0930**	0.1663	0.0367	0.1678	0.1242	0.0243	0.1238
MK13	**0.1099**	0.0228	**0.1081**	0.2133	0.0413	0.2128	0.1374	0.0332	0.1398
MK14	**0.1357**	0.0196	**0.1320**	0.2038	0.0277	0.2062	0.1578	0.0177	0.1623
MK15	**0.0811**	0.0194	**0.0820**	0.1554	0.0386	0.1423	0.1073	0.0252	0.1075
LI01	**0.0482**	0.0099	**0.0485**	0.0651	0.0135	0.0679	0.1032	0.0175	0.1029
LI02	**0.1027**	0.0186	**0.1050**	0.1288	0.0255	0.1203	0.1638	0.0217	0.1605
LI03	**0.0843**	0.0169	**0.0848**	0.1092	0.0156	0.1089	0.1342	0.0204	0.1373
LI04	**0.1111**	0.0167	**0.1082**	0.1407	0.0234	0.1343	0.2017	0.0317	0.2057
LI05	**0.1025**	0.0235	**0.1000**	0.1339	0.0218	0.1317	0.1659	0.0242	0.1615
LI06	**0.0892**	0.0167	**0.0882**	0.1110	0.0180	0.1148	0.1448	0.0236	0.1437
LI07	**0.0924**	0.0162	**0.0933**	0.1117	0.0163	0.1130	0.1593	0.0223	0.1551
LI08	**0.0994**	0.0215	**0.1022**	0.1231	0.0186	0.1246	0.1321	0.0198	0.1300
LI09	**0.0890**	0.0230	**0.0798**	0.1381	0.0229	0.1382	0.1741	0.0314	0.1679
LI10	**0.0670**	0.0163	**0.0685**	0.0861	0.0165	0.0862	0.1342	0.0256	0.1301

Table 11
The HV results of HEDA-DEV and its two variants on each instance.

Instance	HEDA-DEV			HEDA-DE			HEDA-VNS		
	Avg	SD	Med	Avg	SD	Med	Avg	SD	Med
MK01	**0.7621**	0.0490	**0.7604**	0.6032	0.0448	0.6050	0.6999	0.0537	0.7003
MK02	**0.7019**	0.0763	**0.6749**	0.5205	0.0602	0.5105	0.6140	0.0397	0.6128
MK03	**0.7611**	0.0398	**0.7548**	0.6058	0.0574	0.6123	0.7125	0.0379	0.7171
MK04	**0.8753**	0.0630	**0.8774**	0.7325	0.0519	0.7296	0.8013	0.0623	0.8144
MK05	**0.7556**	0.0417	**0.7560**	0.6561	0.0404	0.6623	0.7274	0.0443	0.7334
MK06	**0.7387**	0.0438	**0.7312**	0.5633	0.0442	0.5698	0.6958	0.0631	0.6886
MK07	**0.7498**	0.0362	**0.7364**	0.5849	0.0426	0.5839	0.6894	0.0451	0.6953
MK08	**0.7758**	0.0345	**0.7747**	0.6343	0.0425	0.6404	0.7118	0.0454	0.7109
MK09	**0.7766**	0.0425	**0.7813**	0.6591	0.0568	0.6689	0.7382	0.0421	0.7397
MK10	**0.7109**	0.0409	**0.7160**	0.5483	0.0673	0.5512	0.6311	0.0560	0.6345
MK11	**0.7492**	0.0532	**0.7423**	0.6314	0.0350	0.6337	0.7007	0.0444	0.7083
MK12	**0.7223**	0.0493	**0.7193**	0.6189	0.0450	0.6187	0.6726	0.0366	0.6733
MK13	**0.7994**	0.0321	**0.7995**	0.6110	0.0574	0.6030	0.7227	0.0552	0.7226
MK14	**0.7342**	0.0422	**0.7346**	0.6117	0.0505	0.6187	0.6970	0.0308	0.6908
MK15	**0.7970**	0.0271	**0.7925**	0.6983	0.0521	0.7178	0.7705	0.0402	0.7675
LI01	**0.9323**	0.0207	**0.9352**	0.9042	0.0270	0.8971	0.8518	0.0290	0.8564
LI02	**0.8041**	0.0423	**0.8023**	0.7562	0.0416	0.7539	0.7079	0.0448	0.7047
LI03	**0.7914**	0.0374	**0.7844**	0.7564	0.0470	0.7578	0.6896	0.0350	0.6924
LI04	**0.8026**	0.0301	**0.7945**	0.7579	0.0494	0.7659	0.6372	0.0424	0.6406
LI05	**0.7998**	0.0305	**0.8009**	0.7431	0.0379	0.7385	0.6897	0.0316	0.6983
LI06	**0.7898**	0.0258	**0.7898**	0.7526	0.0379	0.7477	0.6660	0.0387	0.6627
LI07	**0.8035**	0.0280	**0.8073**	0.7537	0.0348	0.7449	0.6950	0.0328	0.6944
LI08	**0.8177**	0.0347	**0.8206**	0.7557	0.0346	0.7479	0.7229	0.0339	0.7292
LI09	**0.8333**	0.0437	**0.8307**	0.7536	0.0380	0.7498	0.7067	0.0332	0.7007
LI10	**0.8548**	0.0299	**0.8619**	0.8215	0.0308	0.8162	0.7360	0.0323	0.7361

Table 12
Wilcoxon Signed-Rank Test results of HEDA-DEV and its two variants on IGD.

Instance	HEDA-DE - HEDA-DEV				HEDA-VNS - HEDA-DEV			
	R^+	R^-	Z	P-Value	R^+	R^-	Z	P-Value
MK01	210	0	−3.920	0.000089	174	36	−2.576	0.010
MK02	210	0	−3.920	0.000089	206	4	−3.771	0.000163
MK03	208	2	−3.845	0.000120	167	43	−2.315	0.021
MK04	196	14	−3.397	0.001	180	30	−2.800	0.005
MK05	209	1	−3.883	0.000103	160	50	−2.053	0.040
MK06	210	0	−3.920	0.000089	181	29	−2.837	0.005
MK07	210	0	−3.921	0.000088	189.5	20.5	−3.155	0.002
MK08	207	3	−3.808	0.000140	183	27	−2.912	0.004
MK09	208.5	1.5	−3.864	0.000111	190	20	−3.173	0.002
MK10	210	0	−3.920	0.000089	201	9	−3.584	0.000338
MK11	209	1	−3.883	0.000103	162	48	−2.128	0.033
MK12	198	12	−3.472	0.001	180	30	−2.800	0.005
MK13	210	0	−3.920	0.000089	186.5	23.5	−3.043	0.002
MK14	210	0	−3.920	0.000089	173	37	−2.539	0.011
MK15	210	0	−3.920	0.000089	186	24	−3.024	0.002
LI01	188	22	−3.099	0.002	210	0	−3.920	0.000089
LI02	179	31	−2.763	0.006	210	0	−3.920	0.000089
LI03	198	12	−3.472	0.001	210	0	−3.920	0.000089
LI04	194.5	15.5	−3.342	0.001	210	0	−3.920	0.000089
LI05	205	5	−3.733	0.000189	208	2	−3.845	0.000120
LI06	182	28	−2.875	0.004	210	0	−3.920	0.000089
LI07	192	18	−3.248	0.001	210	0	−3.920	0.000089
LI08	181.5	28.5	−2.856	0.004	198	12	−3.472	0.001
LI09	203	7	−3.659	0.000254	210	0	−3.920	0.000089
LI10	190	20	−3.174	0.002	210	0	−3.920	0.000089

degree between jobs and processing factories; the probability matrix $\boldsymbol{\sigma}$ which represents the matching degree between product and assembly factory.

3.3.1. Initialization and sampling method

Initialize the probability matrices $\boldsymbol{\rho}$ and $\boldsymbol{\sigma}$ with a uniform distribution, as shown in Equations (15) and (16). The element $\rho_{i,j}(g)$ in probability matrix $\boldsymbol{\rho}$ denotes the probability that job i is allocated to processing factory j in the g-th iteration. Similarly, the element $\sigma_{i,j}(g)$ in probability matrix $\boldsymbol{\sigma}$ indicates the probability that product i is allocated to assembly factory j in the g-th iteration.

$$\rho_{i,j}(0)=\frac{1}{f},\forall i=1,2,\ldots,n;\forall j=1,2,\ldots,f \tag{15}$$

$$\sigma_{i,j}(0)=\frac{1}{A},\forall i=1,2,\ldots,q;\forall j=1,2,\ldots,A \tag{16}$$

In the subsequent iteration process, the algorithm generates new individuals by sampling probability matrices $\boldsymbol{\rho}$ and $\boldsymbol{\sigma}$. Specifically, the sampling process is similar to roulette selection. Taking probability matrix $\boldsymbol{\rho}$ as an example, each row of the matrix is calculated by Equation (17).

Table 13
Wilcoxon Signed-Rank Test results of HEDA-DEV and its two variants on HV.

Instance	HEDA-DE - HEDA-DEV				HEDA-VNS - HEDA-DEV			
	R^+	R^-	Z	P-Value	R^+	R^-	Z	P-Value
MK01	0	210	−3.920	0.000088	27	183	−2.912	0.004
MK02	1	209	−3.883	0.000103	4	206	−3.771	0.000163
MK03	0	210	−3.920	0.000088	33	177	−2.688	0.007
MK04	7	203	−3.659	0.000254	21	189	−3.136	0.002
MK05	1	209	−3.883	0.000103	51	159	−2.016	0.044
MK06	0	210	−3.920	0.000088	46	164	−2.203	0.028
MK07	0	210	−3.920	0.000088	8	202	−3.621	0.000293
MK08	0	210	−3.920	0.000088	8	202	−3.622	0.000293
MK09	0	210	−3.920	0.000088	31	159	−2.575	0.010
MK10	0	210	−3.920	0.000088	14	196	−3.397	0.001
MK11	2	208	−3.845	0.000120	35	175	−2.613	0.009
MK12	11	199	−3.509	0.000449	23	187	−3.061	0.002
MK13	0	210	−3.920	0.000088	11	199	−3.509	0.000449
MK14	1	209	−3.883	0.000103	32	178	−2.725	0.006
MK15	0	210	−3.920	0.000088	45	165	−2.240	0.025
LI01	31	179	−2.763	0.006	0	210	−3.920	0.000088
LI02	30	180	−2.800	0.005	1	209	−3.883	0.000103
LI03	36	174	−2.576	0.010	0	210	−3.920	0.000088
LI04	27	183	−2.912	0.004	0	210	−3.920	0.000088
LI05	8	202	−3.622	0.000293	0	210	−3.920	0.000088
LI06	37	173	−2.539	0.011	0	210	−3.920	0.000088
LI07	16	194	−3.323	0.001	0	210	−3.920	0.000088
LI08	8	202	−3.621	0.000293	0	210	−3.920	0.000088
LI09	6	204	−3.696	0.000219	0	210	−3.920	0.000088
LI10	26	184	−2.949	0.003	0	210	−3.920	0.000088

Table 14
The comparative results of the total cost.

Instance	HEDA-DEV		EDA		DE		ABC		NSGA-II		MOEA/D	
	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min
MK01	**1565**	**1656**	1595	1705	1726	1912	1740	1910	1647	1826	1745	1881
MK02	**1657**	**1761**	1717	1816	1808	1960	1877	1987	1823	1960	1783	1968
MK03	**8545**	**9160**	8841	9600	9799	10807	10087	10704	9721	10610	10063	10859
MK04	**3432**	**3714**	3656	3930	4096	4524	4281	4532	4008	4478	4295	4549
MK05	**5062**	**5380**	5125	5410	5802	6342	5887	6161	5642	6135	6008	6306
MK06	**4544**	**4775**	4719	4944	5413	5861	5246	5589	5259	5678	5455	5812
MK07	**5660**	**6136**	5682	6338	7030	7720	7321	7703	6529	7519	7126	7718
MK08	**10906**	**11490**	11164	11773	12221	13362	12629	13287	12051	13047	12672	13225
MK09	**18274**	**18985**	18635	19329	20666	22310	21175	22170	20556	22087	21277	22561
MK10	**17769**	**18544**	18075	18865	19674	21286	20476	21462	19615	20975	20219	21449
MK11	**23615**	**25163**	24711	25614	26094	28156	27189	28768	26112	27655	26613	28414
MK12	**26029**	**27054**	26246	27433	29490	31363	30455	31334	29363	31123	30160	31423
MK13	**26125**	**27614**	26912	28017	29820	31697	30540	31489	29468	31268	30330	32221
MK14	**39750**	**41267**	40080	41686	44519	46819	45479	46833	44195	46175	44802	46593
MK15	**37897**	**39202**	38666	39712	42746	44981	42698	45051	42293	44235	43726	45166
LI01	**129253**	**131887**	130343	132653	145533	149932	147830	150724	144994	149772	147042	150981
LI02	**130704**	**133778**	132324	135867	144436	149053	147293	150931	145105	148786	146216	149817
LI03	**132300**	**136487**	135225	138991	146714	151664	148849	152951	147104	151370	148474	153425
LI04	**418149**	**424560**	425263	432530	451831	460210	457689	462935	451486	460405	456603	461832
LI05	**428949**	**437448**	436237	444672	457185	463371	460892	467020	457082	463985	458303	464934
LI06	**436497**	**443019**	443604	450539	464856	472108	467537	473140	465410	472158	467805	472121
LI07	**732729**	**746962**	744023	755533	768937	779981	776441	782081	771305	781542	774760	782913
LI08	**711484**	**725092**	725329	735628	758592	770117	760842	771917	762080	770404	765418	773842
LI09	**717989**	**729453**	732178	742349	760185	774294	767691	775586	764878	774893	768437	775829
LI10	**735664**	**748522**	745933	755639	766341	775793	770927	776891	767570	777334	770204	777673

$$prob\_\rho_{i,j}(g)=\frac{\sum_{s=1}^{j}\rho_{i,s}(g)}{\sum_{s=1}^{f}\rho_{i,s}(g)},\forall i=1,2,\ldots,n;\forall j=1,2,\ldots,f \tag{17}$$

where $prob\_\rho_{i,j}(g)$ indicates the selection probability of the job i assigned to the processing factory j in the g-th iteration.

Then, a random number δ between 0 and 1 is generated to compare with $prob\_\rho_{i,j}(g)$. If $\delta \le prob\_\rho_{i,1}(g)$, job i is assigned to processing factory "1". Otherwise, if $prob\_\rho_{i,j-1}(g) \le \delta \le prob\_\rho_{i,j}(g)$, job i is assigned to processing factory j.

3.3.2. Updating mechanism of the probability model

To make the probability model properly represent the evolution trend of the population, the η superior individuals in the population are selected in each iteration to update the probability model. The superior individuals are selected as follows.

First, a non-dominant sorting of the population is conducted to determine the level of each individual. Rank (1) is Pareto solutions. If the number of Pareto solutions is larger or equal to η, then the first η individuals in Pareto solutions are selected. If the number of Pareto solutions is smaller than η, the individuals in rank (2) or lower level are supplemented in turn.

Table 15
The comparative results of the total tardiness.

Instance	HEDA-DEV		EDA		DE		ABC		NSGA-II		MOEA/D	
	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min
MK01	**96**	**123**	104	135	124	168	130	155	123	174	136	179
MK02	**49**	**69**	63	80	82	126	87	108	79	112	97	128
MK03	**36**	**163**	110	217	233	473	187	416	187	391	276	510
MK04	**10**	**24**	18	33	33	127	45	95	31	100	36	151
MK05	**50**	**82**	62	104	89	180	111	178	87	170	106	225
MK06	**40**	**69**	57	86	106	191	113	163	88	183	124	215
MK07	**11**	**62**	28	92	130	379	131	273	133	293	164	298
MK08	**56**	**145**	99	195	179	477	200	406	174	485	230	475
MK09	**207**	**371**	292	445	407	1022	629	974	402	864	500	1442
MK10	**270**	**449**	397	540	473	1163	578	1250	595	1191	677	1321
MK11	**182**	**429**	328	670	402	2850	860	4052	541	2910	888	3182
MK12	**491**	**760**	606	944	968	1886	930	1896	892	1746	1116	2654
MK13	**329**	**550**	454	698	702	1928	603	2315	632	2161	891	2309
MK14	**1140**	**1520**	1247	1769	1555	2853	1666	2501	1622	2885	1683	3476
MK15	**571**	**869**	648	1005	935	1712	995	1501	936	1750	1117	1840
LI01	**0**	**403**	99	623	743	5650	752	4124	550	5388	1086	7534
LI02	**1460**	**2533**	1598	2820	2778	8745	3194	9171	2785	11353	3137	7820
LI03	**173**	**1335**	272	1517	1043	10387	2158	15197	794	13596	1880	20264
LI04	**7418**	**11034**	8117	11979	12097	39783	14204	36974	12301	38318	18637	42117
LI05	**696**	**1983**	813	2657	3158	44737	4213	51489	2877	41680	1631	57077
LI06	**2488**	**5746**	3271	6263	6454	31942	7434	35725	6586	36250	8498	38648
LI07	**5632**	**10062**	6593	12293	12590	60600	16151	65677	13172	58298	17118	65448
LI08	**25557**	**33360**	27816	35611	34921	149782	46969	88853	42128	126411	49525	108653
LI09	**3520**	**8474**	5030	9192	10151	103650	33519	99938	18280	112400	28824	96820
LI10	**12415**	**18268**	14316	20558	27202	94858	30723	85551	24546	85155	29104	108930

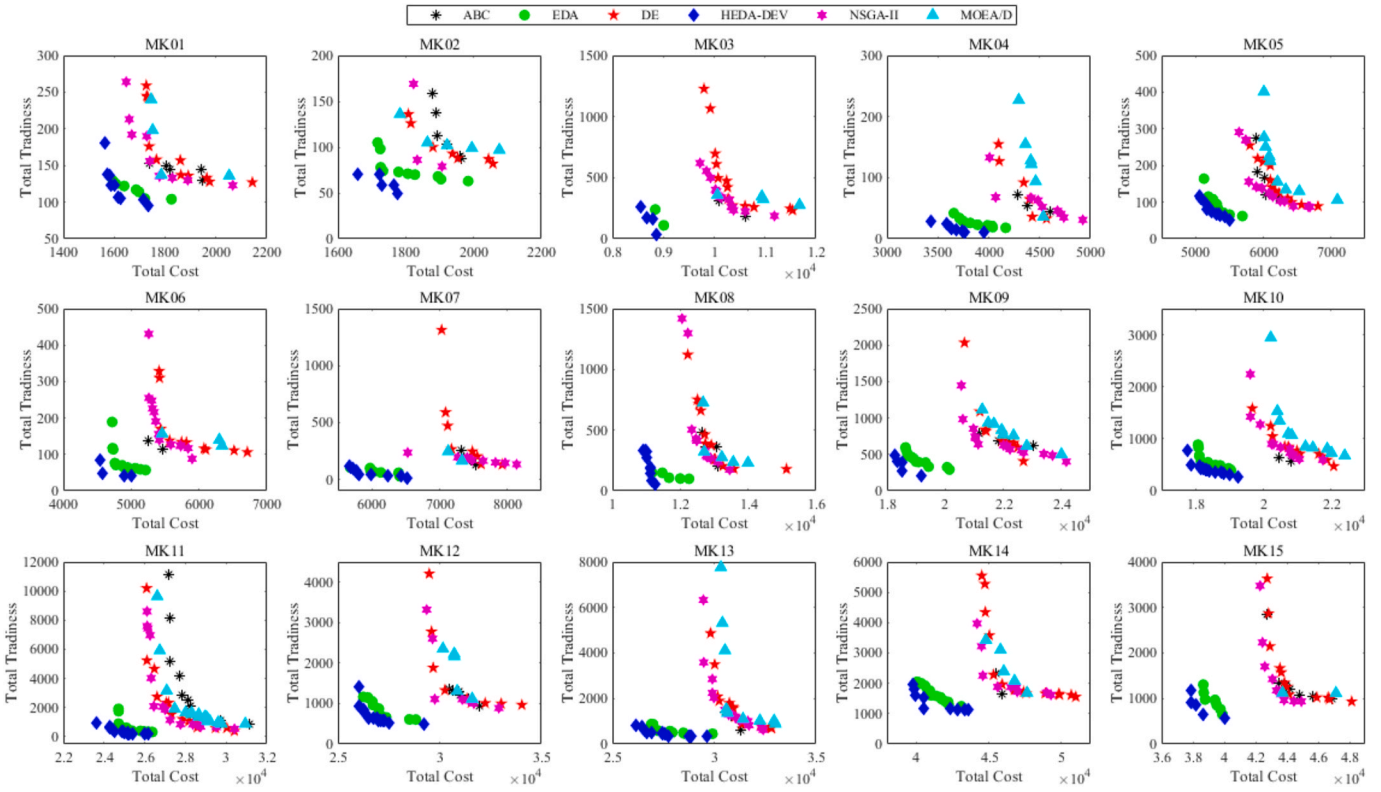

Fig. 10. Pareto solution set of six algorithms on Test Set 1.

To be determined by the superior individuals, the probability model is updated by means of incremental learning, as in Equations (18)–(20):

$$\eta = \mu \cdot Popsize \tag{18}$$

$$\rho_{i,j}(g+1) = (1-\alpha)\rho_{i,j}(g) + \frac{\alpha}{\eta}\sum_{z=1}^{\eta} I_{i,j}^{z}(g), \forall i,j \tag{19}$$

$$\sigma_{i,j}(g+1) = (1-\beta)\sigma_{i,j}(g) + \frac{\beta}{\eta}\sum_{z=1}^{\eta} L_{i,j}^{z}(g), \forall i,j \tag{20}$$

In the above formulas, $\mu \in (0,1)$, *Popsize* is the population size, $\alpha, \beta \in (0,1)$ are the learning rate. $I_{i,j}^{z}(g)$ and $L_{i,j}^{z}(g)$ are respectively the indicative variables corresponding to the z-th elite individual in the g-th iteration of the algorithm, which are defined as following Equations (21)

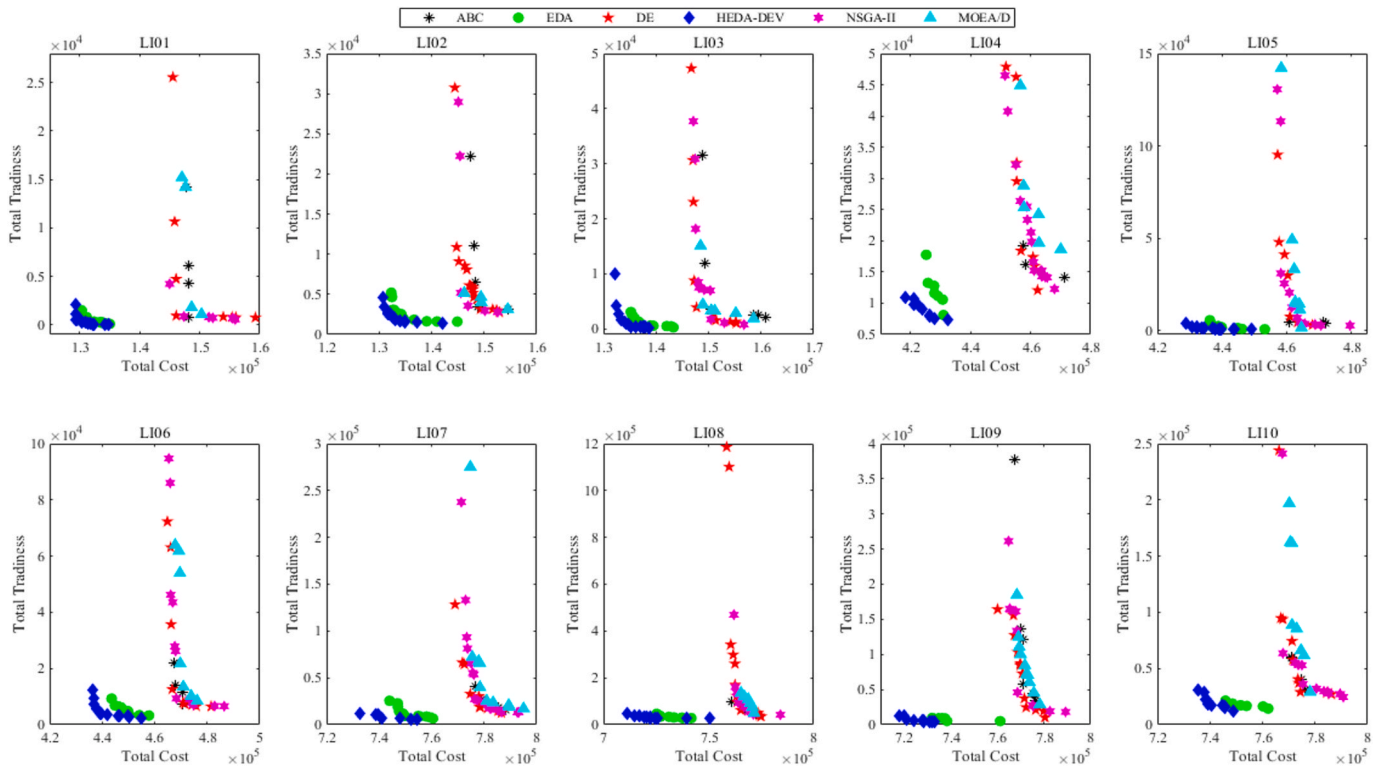

Fig. 11. Pareto solution set of six algorithms on Test Set 2.

Table 16
The IGD results of six algorithms on each instance.

Instance	HEDA-DEV		EDA		DE		ABC		NSGA-II		MOEA/D	
	Avg	Med	Avg	Med	Avg	Med	Avg	Med	Avg	Med	Avg	Med
MK01	**0.0489**	**0.0459**	0.0924	0.0850	0.3361	0.3392	0.3627	0.3704	0.2863	0.2843	0.3932	0.4107
MK02	**0.0473**	**0.0467**	0.1115	0.1136	0.3628	0.3709	0.4089	0.4110	0.3113	0.3231	0.4495	0.4440
MK03	**0.0603**	**0.0614**	0.1251	0.1249	0.4458	0.4451	0.4653	0.4710	0.3915	0.3803	0.5109	0.4995
MK04	**0.0304**	**0.0319**	0.0888	0.0955	0.4182	0.4335	0.4395	0.4535	0.3689	0.3884	0.5067	0.5017
MK05	**0.0287**	**0.0285**	0.0650	0.0684	0.4256	0.4257	0.4011	0.4020	0.3338	0.3289	0.4858	0.4776
MK06	**0.0455**	**0.0452**	0.0888	0.0863	0.4121	0.4140	0.3494	0.3428	0.3608	0.3623	0.4513	0.4486
MK07	**0.0286**	**0.0258**	0.0684	0.0720	0.4499	0.4554	0.4487	0.4395	0.3805	0.3890	0.4559	0.4714
MK08	**0.0363**	**0.0377**	0.0847	0.0820	0.4116	0.4122	0.4426	0.4546	0.3744	0.3813	0.4545	0.4530
MK09	**0.0273**	**0.0246**	0.0661	0.0665	0.4464	0.4407	0.4560	0.4604	0.3787	0.3739	0.5136	0.5098
MK10	**0.0218**	**0.0222**	0.0548	0.0530	0.3771	0.3728	0.4092	0.4086	0.3560	0.3640	0.4587	0.4545
MK11	**0.0202**	**0.0214**	0.0557	0.0571	0.3618	0.3734	0.4468	0.4390	0.3030	0.3007	0.4146	0.4230
MK12	**0.0286**	**0.0286**	0.0624	0.0622	0.4389	0.4433	0.4722	0.4657	0.4070	0.4107	0.5122	0.4979
MK13	**0.0318**	**0.0255**	0.0528	0.0523	0.3839	0.3880	0.4057	0.4095	0.3653	0.3690	0.4304	0.4145
MK14	**0.0328**	**0.0273**	0.0582	0.0579	0.3948	0.3988	0.4144	0.4056	0.3424	0.3504	0.4418	0.4423
MK15	**0.0397**	**0.0380**	0.0866	0.0848	0.4881	0.4832	0.5123	0.5054	0.4346	0.4352	0.5496	0.5586
LI01	**0.0133**	**0.0130**	0.0294	0.0294	0.5506	0.5586	0.6089	0.6127	0.5628	0.5684	0.6444	0.6428
LI02	**0.0272**	**0.0264**	0.0470	0.0443	0.4763	0.4788	0.5764	0.5755	0.5080	0.5176	0.5358	0.5398
LI03	**0.0156**	**0.0150**	0.0460	0.0454	0.4188	0.4306	0.4926	0.5048	0.4218	0.4245	0.4891	0.4783
LI04	**0.0201**	**0.0200**	0.0844	0.0885	0.6065	0.6142	0.6556	0.6570	0.5967	0.6035	0.6619	0.6730
LI05	**0.0199**	**0.0192**	0.0607	0.0617	0.4808	0.4774	0.5372	0.5441	0.4701	0.4723	0.5256	0.5301
LI06	**0.0255**	**0.0226**	0.0886	0.0884	0.5178	0.5203	0.5540	0.5704	0.5259	0.5271	0.5671	0.5629
LI07	**0.0299**	**0.0282**	0.0987	0.0986	0.4915	0.4878	0.5426	0.5498	0.5050	0.5000	0.5498	0.5496
LI08	**0.0296**	**0.0270**	0.0939	0.0861	0.5145	0.5175	0.5449	0.5496	0.5190	0.5192	0.5822	0.5793
LI09	**0.0153**	**0.0133**	0.0979	0.0960	0.5410	0.5466	0.5878	0.5846	0.5455	0.5488	0.5834	0.5795
LI10	**0.0197**	**0.0185**	0.1135	0.1040	0.5216	0.5278	0.5674	0.5707	0.5260	0.5394	0.5888	0.5874

and (22):

$$I_{i,j}^{z}(\mathrm{g}) = \begin{cases} 1, \text{if job } j \text{ is assigned to profeccing factory } i \\ 0, \text{otherwise} \end{cases} \tag{21}$$

$$L_{i,j}^{z}(\mathrm{g}) = \begin{cases} 1, \text{if product } j \text{ is assigned to assembly factory } i \\ 0, \text{otherwise} \end{cases} \tag{22}$$

3.4. Population generation method

After sampling the probability model, we can get the factory assignment scheme for jobs and products, and then we can obtain the detailed operation scheduling sequence in the factory through the factory internal scheduling (FIS).

Table 17
The HV results of six algorithms on each instance.

Instance	HEDA-DEV		EDA		DE		ABC		NSGA-II		MOEA/D	
	Avg	Med	Avg	Med	Avg	Med	Avg	Med	Avg	Med	Avg	Med
MK01	**0.9227**	**0.9243**	0.8491	0.8493	0.5697	0.5655	0.5014	0.4955	0.6414	0.6405	0.4760	0.4572
MK02	**0.9102**	**0.8958**	0.7854	0.7804	0.5123	0.5068	0.4313	0.4294	0.5454	0.5354	0.3977	0.4080
MK03	**0.9048**	**0.9006**	0.7969	0.7962	0.4721	0.4758	0.4352	0.4206	0.5236	0.5177	0.3843	0.3925
MK04	**0.9204**	**0.9191**	0.7981	0.7939	0.4556	0.4365	0.3922	0.3915	0.5088	0.4985	0.3318	0.3437
MK05	**0.9408**	**0.9405**	0.8788	0.8736	0.5119	0.5005	0.4810	0.4920	0.5827	0.5910	0.4071	0.4144
MK06	**0.9505**	**0.9482**	0.8549	0.8521	0.4908	0.4923	0.5205	0.5279	0.5530	0.5542	0.4229	0.4189
MK07	**0.9531**	**0.9552**	0.8741	0.8675	0.4785	0.4740	0.4274	0.4392	0.5464	0.5406	0.4364	0.4280
MK08	**0.9493**	**0.9493**	0.8656	0.8631	0.5350	0.5315	0.4704	0.4588	0.5758	0.5799	0.4599	0.4835
MK09	**0.9587**	**0.9560**	0.9007	0.8989	0.5110	0.5081	0.4609	0.4560	0.5728	0.5664	0.4192	0.4383
MK10	**0.9449**	**0.9405**	0.8827	0.8775	0.5391	0.5413	0.4536	0.4468	0.5679	0.5675	0.4470	0.4422
MK11	**0.9168**	**0.9030**	0.8339	0.8322	0.5633	0.5640	0.4158	0.4135	0.5986	0.6010	0.4698	0.4769
MK12	**0.9691**	**0.9684**	0.9023	0.8986	0.4802	0.4785	0.4110	0.4108	0.5135	0.5153	0.3785	0.3933
MK13	**0.9381**	**0.9333**	0.8730	0.8717	0.4905	0.4835	0.4242	0.4200	0.5250	0.5287	0.4179	0.4333
MK14	**0.9668**	**0.9667**	0.9044	0.9041	0.4902	0.4894	0.4240	0.4265	0.5326	0.5281	0.4157	0.4060
MK15	**0.9498**	**0.9527**	0.8771	0.8796	0.4548	0.4532	0.4087	0.4207	0.5037	0.4983	0.3687	0.3607
LI01	**0.9859**	**0.9854**	0.9439	0.9453	0.3986	0.3995	0.3312	0.3277	0.3938	0.3920	0.3010	0.3115
LI02	**0.9673**	**0.9666**	0.9138	0.9160	0.4369	0.4390	0.3186	0.3153	0.4190	0.4181	0.3534	0.3437
LI03	**0.9740**	**0.9736**	0.8925	0.8948	0.5163	0.5139	0.4212	0.4165	0.5069	0.5095	0.4196	0.4313
LI04	**0.9684**	**0.9684**	0.8177	0.8097	0.3095	0.3063	0.2299	0.2246	0.3056	0.3004	0.2280	0.2307
LI05	**0.9601**	**0.9573**	0.8239	0.8243	0.3955	0.3945	0.3125	0.3120	0.3922	0.3920	0.3210	0.3202
LI06	**0.9726**	**0.9679**	0.8267	0.8263	0.3789	0.3768	0.3189	0.3103	0.3780	0.3744	0.3190	0.3181
LI07	**0.9136**	**0.9069**	0.7690	0.7658	0.3731	0.3700	0.2992	0.3022	0.3570	0.3561	0.2968	0.2966
LI08	**0.9311**	**0.9274**	0.7893	0.7952	0.3346	0.3348	0.2916	0.2863	0.3310	0.3315	0.2574	0.2611
LI09	**0.9623**	**0.9616**	0.8007	0.8024	0.3710	0.3645	0.3103	0.3133	0.3659	0.3621	0.3157	0.3273
LI10	**0.9439**	**0.9427**	0.7847	0.7964	0.3887	0.3848	0.3246	0.3224	0.3802	0.3826	0.3112	0.3171

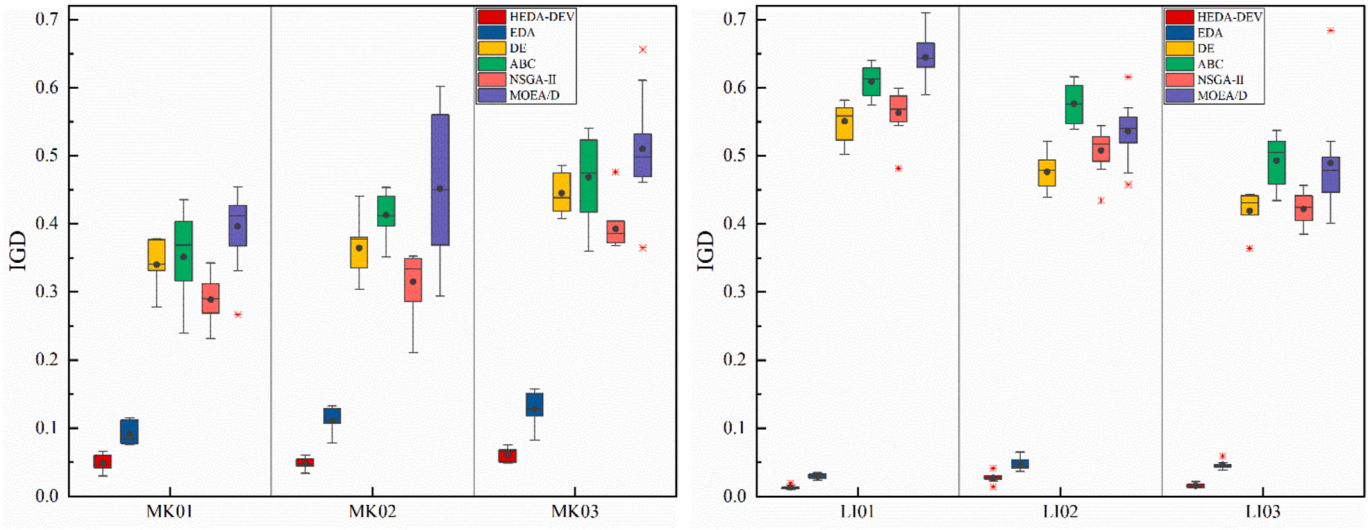

Fig. 12. The box plots of IGD obtained by the six algorithms.

The example in Section 3.2 is used again, i.e., 9 jobs, 3 processing factories, and 2 assembly factories. An example of FIS is given in Fig. 4. First, all the operations are randomly arranged into a sequence *vector*, and the occurrence sequence of the number represents the operation of the job. Then through sampling probability matrix, we can get the assignment scheme of the job-processing factory. The jobs "2, 3, 9" are allocated to factory F_1, and the operations of jobs "2, 3, 9" will be taken out from the *vector* in sequence. Then we can get the operation scheduling sequence *v1-O* in factory F_1 by putting these numbers together. After assigning machines, the complete operation-machine scheduling sequence can be obtained.

3.5. Dynamic mutation strategy

At present, various mutation strategies have been presented, including DE/rand/1 with an outstanding global search capability, and DE/best/1 with an outstanding local search capability (Wu et al., 2016). According to the characteristics of EDA, we design a similarity coefficient based on the probability matrices ρ and σ to judge the stage of the algorithm and then choose the mutation strategy suitable for different stages. In addition, due to the features of the coding scheme, the crossover and mutation only work on the processing part and do not involve the assembly part. The variable neighborhood search contains neighborhood structures specifically for the assembly part, as detailed in Section 3.6.

3.5.1. Similarity coefficient

Because the probability matrix of EDA describes the distribution of the populations, we propose a method to compare the similarity of the two generations by calculating the similarity coefficient of the probability matrix between the two generations. Then we infer the stage of the algorithm to choose different mutation operations. The similarity coef-

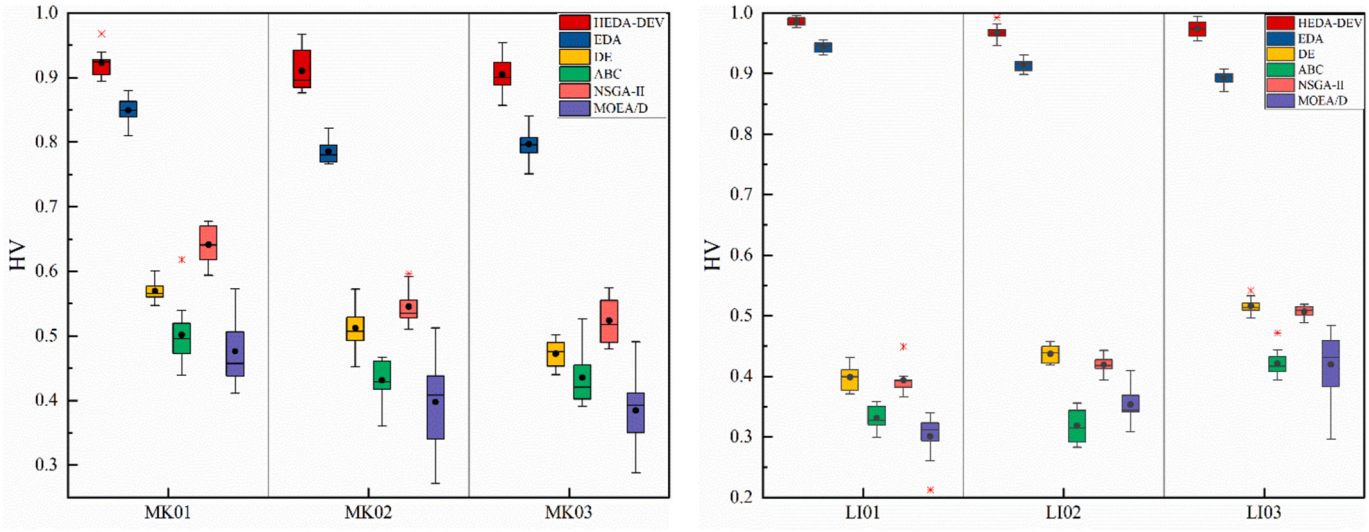

Fig. 13. The box plots of HV obtained by the six algorithms.

Table 18
The significant differences of IGD obtained by Wilcoxon Signed-Rank Test.

Instance	EDA VS HEDA-DEV		DE VS HEDA-DEV		ABC VS HEDA-DEV		NSGA-II VS HEDA-DEV		MOEA/D VS HEDA-DEV	
	Z	P-Value	Z	P-Value	Z	P-Value	Z	P-Value	Z	P-Value
MK01	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK02	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK03	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK04	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK05	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK06	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK07	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK08	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK09	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK10	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK11	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK12	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK13	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK14	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK15	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
LI01	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
LI02	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
LI03	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
LI04	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
LI05	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
LI06	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
LI07	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
LI08	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
LI09	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
LI10	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005

ficient *similar* is calculated as follows:

$$similar_{\rho} = \left(\sum_{i=1}^{n} \sum_{j=0}^{f} \left| \rho_{i,j}(g) - \rho_{i,j}(g-1) \right| \right) \Big/ n \tag{23}$$

$$similar_{\sigma} = \left(\sum_{i=1}^{q} \sum_{j=0}^{A} \left| \sigma_{i,j}(g) - \sigma_{i,j}(g-1) \right| \right) \Big/ q \tag{24}$$

$$similar = (similar_{\rho} + similar_{\sigma}) / 2 \tag{25}$$

In the later stage of the algorithm, there should be a strong local search ability to increase the accuracy and convergence speed. As the convergence slows down in the later stage, the probability model tends to be stable gradually and the value of *similar* will become smaller and smaller. Therefore, the mutation strategy is chosen as below:

$$\varepsilon = \begin{cases} DE/rand/1, & \text{if rand } (0,1) < similar \\ DE/best/1, & \text{otherwise} \end{cases} \tag{26}$$

3.5.2. Mutation operator

Step 1. After the non-dominated sorting, an elite individual "X_a" and a poor individual "X_b" are taken out from the front 50% and back 50% of the population respectively, and then take a different individual "X_c" randomly.

Step 2. By subtracting the sequence of "X_a" from that of "X_b", we can get "(X_a - X_b)", and then we can get $V_{ij}^{'}$ with "X_c" according to Equation (27).

Step 3. The final variant can be obtained by eliminating the negative number in $V_{ij}^{'}$. This operation process is shown in Equation (28).

Table 19
The significant differences of HV obtained by Wilcoxon Signed-Rank Test.

Instance	EDA VS HEDA-DEV		DE VS HEDA-DEV		ABC VS HEDA-DEV		NSGA-II VS HEDA-DEV		MOEA/D VS HEDA-DEV	
	Z	P-Value	Z	P-Value	Z	P-Value	Z	P-Value	Z	P-Value
MK01	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK02	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK03	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK04	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK05	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK06	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK07	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK08	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK09	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK10	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK11	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK12	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK13	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK14	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
MK15	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
LI01	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
LI02	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
LI03	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
LI04	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
LI05	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
LI06	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
LI07	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
LI08	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
LI09	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
LI10	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005

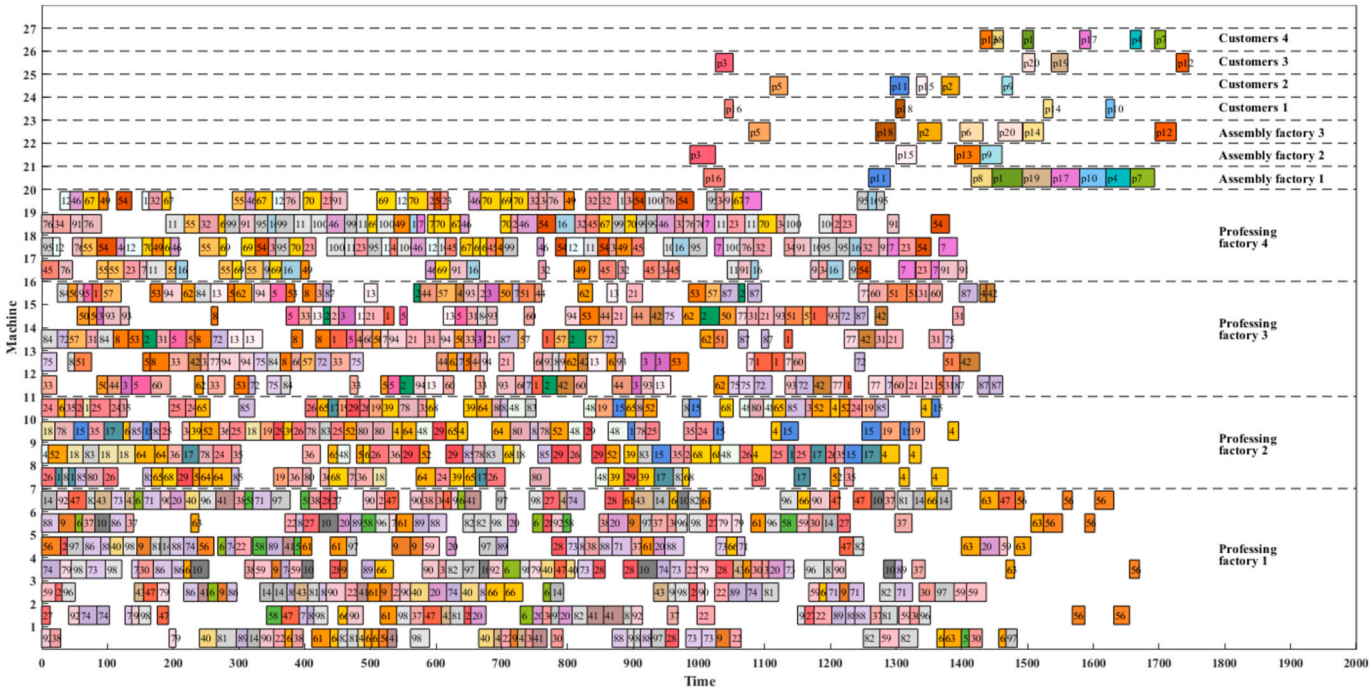

Fig. 14. Gantt Chart of LI03 instance.

$$V_{ij}^{'} = \begin{cases} \left(\left(X_{aj} - X_{bj}\right) + X_{cj}\right)\%n + 1, & rand < F \\ X_{cj} + 1, & rand \geq F \end{cases} \tag{27}$$

$$V_{ij} = \begin{cases} V_{ij}^{'}, & V_{ij}^{'} \geq 0 \\ 0, & V_{ij}^{'} < 0 \end{cases} \tag{28}$$

The above steps are the operation process of DE/rand/1. The process of DE/best/1 is the same as that of DE/rand/1, except that "X_c" is no longer randomly selected but Pareto solution of population. Among Equations (24) and (25), subscript j is the j-th dimensional element of the individual, *rand* is a random number, and F is the mutation probability. In Fig. 5, we illustrate a small instance with nine jobs for ease of understanding, assuming that the mutation probability F is 0.5.

3.6. Crossover

In HEDA-DEV, a new individual can be obtained by exchanging information between mutated individuals and original individuals. The

Table 20
Brief information on products and key components.

No.	Product	key components
1	Stacker reclaimer	(1) Stacking arm, (2) Running mechanism (3) Bucket wheel, (4) Tail carriage (5) Frame (6) Turntable
2	Rotary kiln	(7) Kiln shell, (8) Feed pipe, (9) Discharge pipe, (10) Cooling pipe, (11) Hot air pipe, (12) Supporting drum, (13) Sealing plate, (14) Drive unit, (15) Large gear unit, (16) Gear cover
3	Cement mill	(17) Feed chute, (18) Feed screw cylinder, (19) Liner, (20) Slide bearing, (21) Drive gear, (22) Cylinder, (23) Mill head, (24) Lifting plate, (25) Discharge screw cylinder, (26) Screen, (27) Discharge cover
4	Roller crusher	(28) Thermal resistance, (29) Blackened type 1 hexagonal nut, (30) Square beveled washers for blackened channel steel, (31) Blackened hexagonal head bolt, (32) Proximity switch assembly, (33) Torque arm combination 4 rollers
5	Packaging machine	(34) Connecting tube, (35) Collecting box, (36) Connecting chute, (37) Supporting frame, (38) Top platform, (39) Protection system, (40) Hopper, (41) Dust cover

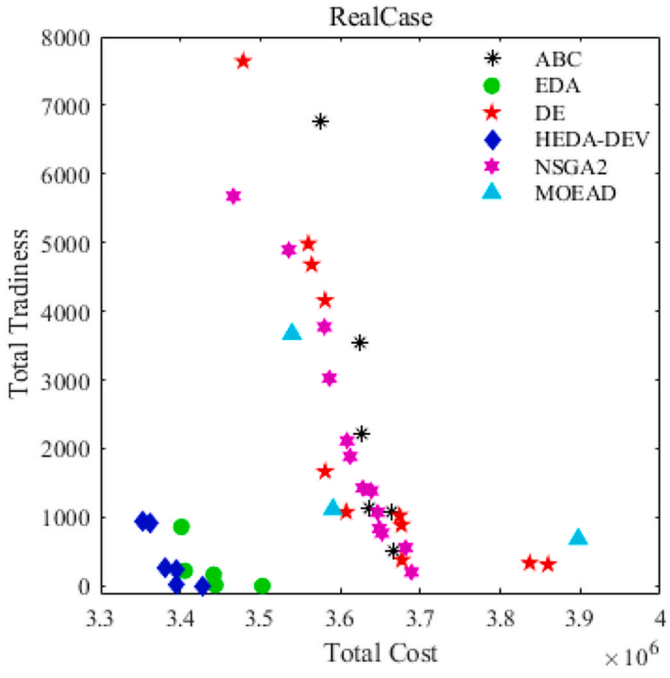

Fig. 15. Pareto solution set of six algorithms on the real case.

example in Section 3.2 is used again, as shown in Fig. 6. Assuming that the crossover probability *CR* is 0.6, V_i is the mutation individual obtained from the above mutation operation, and U_i' can be obtained through the selection of *CR*. Then we carry out the de-duplication operation on U_i', only the first occurrence gene will be reserved (shown in green color), and the duplicate gene will be deleted. Then randomly select individual X_i from the original population. Delete the green part of X_i to get X_i'. Finally, the remaining genes of X_i' are sequentially inserted into U_i'' to generate the final test sequence U_i.

3.7. Multi-neighbor structures and search strategy

In this paper, four targeted neighborhood operators and a random operator are designed, which can work together according to the cooperative search strategy.

3.7.1. Neighborhood structures

(1) Reduce maximum tardiness

NS1: Select product *b* with the largest delay time and locate job *i* that arrives at the assembly factory the latest among all jobs of product *b*. Give job *i* priority scheduling within the factory where job *i* is processed, thereby speeding up the start of the assembly for product *b* and possibly reducing the delay time. To explain it more clearly, an example is presented in Fig. 7. Table 2 shows the details of the example. Jobs "1, 2, 3" are assigned to this processing factory and *v* is the process scheduling scheme. Assume that job "1″ should be processed firstly. First, determine the position of the last operation of job "1". This position (*index*) is 10 (blue part in figure). Then we need to cut this gene to the fifth position in the operation sequence (*index*/2), as well as the machine part. When we execute this neighborhood structure, the completion time for job "1" decreases from 67 to 49.

NS2: Select product *b* with the largest delay time and locate job *i* that arrives at the assembly factory the latest among all jobs of product b, then find processing factory *a* which has the earliest completion time. If job *i* is processed in factory *a*, turn to **NS1** search; otherwise, transfer job *i* to factory *a* and schedule randomly. As shown in Fig. 8, it is still assumed that job "1" needs to be transferred and the completion time of processing factory "2" is the shortest. First, remove all gene "1" and their corresponding machine genes (blue part in figure) from v_1. Then insert these genes randomly into v_2 and available machines are randomly selected in the machine sequence.

NS3: Select product *b* with the largest delay time and re-select the assembly factory closest to its customer.

(2) Reduce total cost

NS4: Select the processing factory with the highest processing cost and find the operation with the highest processing cost. Then traverse all the processing machines in the factory and select the machine with the least processing cost for the operation.

(3) Random Neighborhood Structure

NS5: Select a processing factory at random, and randomly select 20% of the operations processed by the factory and reselect machines for them.

3.7.2. Cooperative search strategy

To fully exploit the potential of the population and improve the efficiency of the search, the object set *PSet* of neighborhood search is set as follows: all the Pareto solutions (set *S*), 10% of the single objective optimal individuals (set S_C, set S_{TD}). If the individuals belong to the set *S*, we apply all five neighborhood operators sequentially. For individuals in set S_C, the neighborhood operator NS4 which is designed to reduce total cost and the random neighborhood operator NS5 are applied. For individuals in set S_{TD}, the neighborhood operators NS1, NS2, NS3 which are designed to reduce total tardiness and the random neighborhood operator NS5 are applied. By using different neighborhood operators for individuals in different sets, we further optimize the objectives of individuals and find better non-dominated solutions. The pseudo-code for the specific process is as follows.

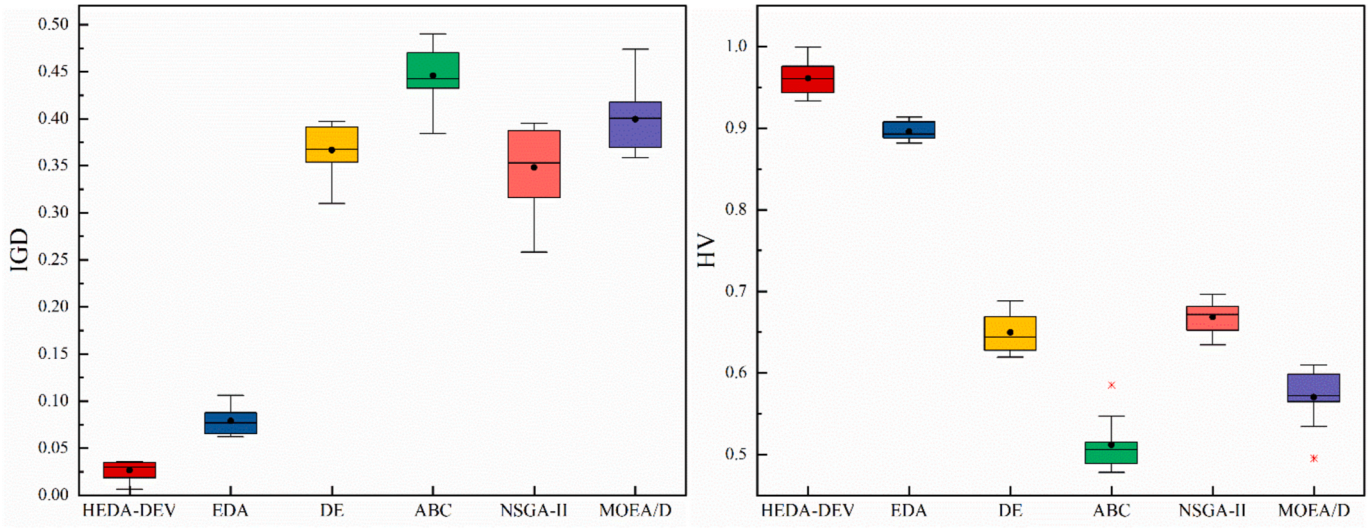

Fig. 16. The box plots of IGD and HV obtained by the six algorithms on the real case.

Table 21
The results of IGD and HV obtained by Wilcoxon Signed-Rank Test.

Indicators	EDA VS HEDA-DEV		DE VS HEDA-DEV		ABC VS HEDA-DEV		NSGA-II VS HEDA-DEV		MOEA/D VS HEDA-DEV	
	Z	P-Value	Z	P-Value	Z	P-Value	Z	P-Value	Z	P-Value
IGD	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005
HV	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005

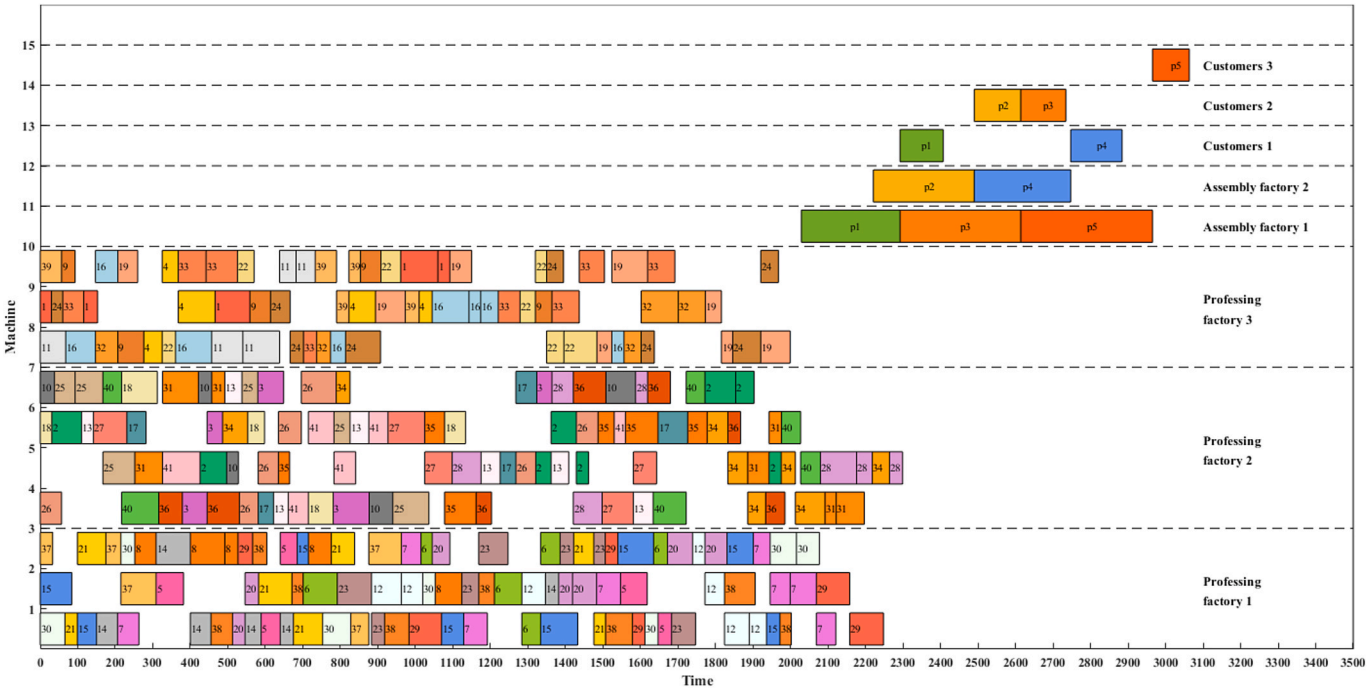

Fig. 17. Gantt Chart of the real case.

Algorithm 1. The presentation of Local Search

```
input: PSet, gen, n = PSet.size(.)
output: new PSet
for i ← 1 to n do
  for j ← 1 to gen do
    for k ← 1 to 5 do
      if PSet[i] ∈ S, then
        Generate X′ through Neighborhood structure k
        if X′ is not dominated by PSet[i], then
        PSet[i] ← X′
        end
      else if PSet[i] ∈ S_C, then
      if k = = 4 ||k = = 5, then
          Generate X′ through Neighborhood structure k
          if X′ is not dominated by PSet[i], then
          PSet[i] ← X′
          end
        end
      else
        if k = = 1 ||k = = 2 ||k = = 3 ||k = = 5, then
          Generate X′ through Neighborhood structure k
        if X′ is not dominated by PSet[i], then
          PSet[i] ← X′
          end
        end
      end
    end
  end
end
```

4. Experimental study

The computational experiments of the algorithms are demonstrated in this section. HEDA-DEV and five other comparison algorithms are carried out in JAVA(JDK1.8) on an Intel Core i5 2.3 GHz PC with 8 GB of RAM. This section is presented according to the following aspects: the determination of test data sets, parameter determination of the algorithms, validation of the mathematical model, ablation experiments, comparison of HEDA-DEV with five other algorithms, and a real case study.

4.1. Data sets and evaluating indicator

Since the proposed problem is a new one, there is no benchmark data set in the existing literature. Therefore, two test sets are constructed to prove the performance of the HEDA-DEV for DAFJSP. Test Set 1 is modified from the instance set proposed by Wu et al. (2019). In order to make the test results more convincing, we add five instances of MK11-MK15. The detailed parameters of Test Set 1 are shown in Table 3. The "10_6_3_2_3_2" represents that there are 10 jobs, 6 machines, 3 processing factories, 2 assembly factories, 3 products, and 2 customers in this instance. The values of the other parameters are generated randomly according to their value ranges. Moreover, to further validate the effectiveness of HEDA-DEV for large-scale problems, we expand the number of jobs and machines in benchmark instances proposed by Brandimarte (1993) to construct Test Set 2. The detailed parameters of Test Set 2 are listed in Table 4. Considering the characteristics of the DAFJSP, we add the following parameters: assembly time, transportation time, processing cost, assembly cost, transportation cost, and the due date of products. The values of each parameter are randomly generated from its range.

In addition, the inverted generational distance (IGD) and hypervolume (HV) are used as metrics to evaluate the performance of algorithms. IGD and HV are calculated as shown below:

$$IGD = \frac{\sum_{x \in \Omega^*} \min_{y \in \Omega_l} dis(x, y)}{|\Omega^*|} \tag{29}$$

$$HV = \cup_{i=1}^{N_s} v_i \tag{30}$$

In Equation (29), Ω^* denotes the real Pareto frontier solution set, Ω_l is the calculated Pareto frontier solution set, and $dis(x, y)$ is the Euclidean distance between x to y. In Equation (30), N_s denotes the number of non-dominated solutions obtained by the algorithm, and v_i denotes the volume/area formed by the i-th solution in the non-dominated solutions with the reference point. Based on the characteristics of the problem, the reference point is set to (1, 1).

4.2. Parameters setting

The value of the key parameters will influence the performance of the algorithm. The key parameters of the HEDA-DEV are the size of the population: *Popsize*, the learning rate of the probability matrix ρ: α, the learning rate of the probability matrix σ: β, the mutation probability: F, the crossover probability: CR and the proportion of superior individuals: μ. To obtain a set of ideal experimental parameters, the Taguchi method (Montgomery, 2005) is used for calibrating the parameters of HEDA-DEV and the other five comparison algorithms. Depending on the number of key parameters described above, the orthogonal experiment is determined as $L_{25}(5^6)$. Different combinations for the orthogonal experiments are listed in Table 5. Considering the difference between the two test sets, the parameter setting experiment is carried out for Test Set 1 and Test Set 2, respectively. The Taguchi method is performed on instance MK07 and instance LI05. All parameter combinations are performed 20 times on HEDA-DEV, and the average of the obtained IGD is used for the average response variable (ARV). Table 6 lists the orthogonal arrays and the obtained results. Table 7 shows the average responses for each parameter. The factor level trend for MK07 and LI05 is presented in Fig. 9.

The *Popsize* is the most significant parameter for both MK07 and LI05. It is also clear from Fig. 9 that the average value of IGD decreases obviously with the increase of *Popsize*. The reason is that a large *Popsize* can improve the population variety. As for μ, a small value will lead to few elite populations participating in updating the probability model, while a large value will lead to updating the probability model with some poor individuals. For the learning rate α and β, the larger the value is, the faster the convergence speed will be. However, if the value is too large, it will lead to premature convergence. As for F and CR, appropriate values can balance the population diversity and convergence rate.

According to the trend plots of the factor levels in Fig. 9, the parameter values of HEDA-DEV for the two test sets can be obtained. In the same way, we determined the parameters of the five comparison algorithms. The parameter settings of the six algorithms are listed in Table 8. The number of iterations is set to 300 for Test Set 1 and 500 for Test Set 2.

4.3. Validation of the proposed model

The software CPLEX 12.9.0 is used to verify the correctness of the proposed model. First, the nonlinear variables are linearized. Then, to implement multi-objective optimization for DAFJSP, the ε-constraint method (Mavrotas, 2009) is adopted for the transformed linear model. Finally, since the true Pareto front cannot be obtained within 4 h for solving instance MK01 (10_6_3_2_3_2), a tiny-scale instance 6_4_2_2_3_2 is generated for model validation. Table 9 shows the comparison result of CPLEX and HEDA-DEV. The result shows that the true Pareto front obtained by CPLEX verifies the correctness of the proposed model. Moreover, the Pareto front obtained by HEDA-DEV is close to the true Pareto front.

4.4. Ablation experiment

In this section, we conduct ablation experiments to evaluate the

contribution of DE-based evolutionary operators and VNS to HEDA-DEV. To validate the effectiveness of the two proposed components, two variants are designed, i.e., HEDA-VNS which removes both the dynamic mutation strategy and crossover from HEDA-DEV, and HEDA-DE which eliminates the variable neighborhood search from HEDA-DEV. These algorithms are run 20 times on all instances with the same parameter settings. The average (Avg), standard deviation (SD), and median (Med) of IGD and HV obtained by three algorithms on each instance are in Tables 10 and 11, respectively.

To further check whether statistically significant differences existed between the experimental results, the Wilcoxon Signed-Rank Test is used to analyze the statistical significance of the HEDA-DEV over the two variants. The results of the Wilcoxon Signed-Rank Test for IGD and HV are given in Tables 12 and 13, respectively. If the P-Value is less than 0.05, we can assume that the results are significantly different. From Tables 12 and 13, we can conclude that HEDA-DEV significantly outperforms the two variants, which demonstrates the effectiveness of the DE-based evolutionary operators and VNS.

4.5. Comparisons of HEDA-DEV with five other algorithms

In order to further validate the performance of the HEDA-DEV for DAFJSP, five algorithms: Estimation of Distribution Algorithm (EDA) (Wang et al., 2013b), Differential Evolution (DE) (Cao et al., 2017), Artificial Bee Colony (ABC) (Li et al., 2011), NSGA-II (Wang et al., 2020) and MOEA/D (Li and Liu, 2018) are employed as comparison. The six algorithms are run 10 times independently on all sets. The comparative results of the total cost and total tardiness are listed in Table 14 and Table 15, respectively. The total cost and total tardiness obtained by HEDA-DEV always have the smallest average value and minimum value in all instances.

Furthermore, the Pareto front of the six algorithms obtained on the two test sets are illustrated in Fig. 10 and Fig. 11, respectively. It can be seen that the quality of the solution obtained by HEDA-DEV obviously outperforms the other algorithms.

Table 16 and Table 17 list the Avg and Med of IGD and HV obtained by six algorithms, respectively. The HEDA-DEV obtains the minimum Avg and Med of IGD and the maximum Avg and Med of HV among all instances, which demonstrates the effectiveness of HEDA-DEV compared to other algorithms. Taking MK01-Mk03 and LI01-LI03 as examples, Fig. 12 and Fig. 13 show the box plots of IGD and HV for the six algorithms, respectively.

In addition, the Wilcoxon Signed-Rank Test is used to test whether the difference between HEDA-DEV and other algorithms is significant. The results are listed in Table 18 and Table 19, respectively. It can be seen that HEDA-DEV statistically outperforms the other five algorithms in all instances. Moreover, to facilitate the understanding of the problem, the Gantt chart of the LI03 instance is shown in Fig. 14. The conclusion can be drawn that the HEDA-DEV is a more effective method for solving DAFJSP.

4.6. A real case study

To further validate the effectiveness of our proposed algorithm, an interview is conducted with a decision-maker from Tianjin Cement Industry Design & Research Institute Co., Ltd, a cement equipment company located in Tianjin, China. This company primarily specializes in EPC (Engineering, Procurement, and Construction) projects and equipment design. It operates two mainframe manufacturing and assembly plants, with one located in Tianjin, China, and the other located in Tangshan, a city in Hebei Province, China. After receiving customer orders, they will be assigned to the two enterprises. To ensure the successful execution of projects and meet customer demands, these two enterprises may choose to outsource specific manufacturing tasks to external manufacturing companies. Subsequently, they will bring the manufactured components back to their own factory for the assembly process. This scenario closely aligns with the research background discussed in the article. Therefore, we have considered a real-life case study from this enterprise, which involves five different products. The brief product information is presented in Table 20. And the detailed dataset can be downloaded from https://github.com/21xiaobai/Case-data.

To verify the effectiveness of HEDA-DEV, six algorithms are run 10 times on the real case. The Pareto front of the six algorithms obtained is illustrated in Fig. 15.

It can be seen from Fig. 15 that HEDA-DEV has a better solution than the other five algorithms. Furthermore, Fig. 16 shows the box plots of IGD and HV for the six algorithms in the real case.

As can be seen from Fig. 16, HEDA-DEV yields the best IGD and HV. In addition, the Wilcoxon Signed-Rank test is used to investigate whether the differences in IGD and HV between HEDA-DEV and other algorithms are significant. The result is displayed in Table 21. All P-Values are less than 0.05 and it can be concluded that there is a significant difference between the results of IGD and HV.

To facilitate a better understanding of the problem, the Gantt chart of the real case obtained by HEDA-DEV is shown in Fig. 17. By solving the real case obtained from the survey, we further validate the effectiveness of the proposed HEDA-DEV.

5. Conclusion and future work

This paper proposes a hybrid distribution estimation optimization algorithm based on DE operator and VNS for solving the DAFJSP with the goal of minimizing the total cost and tardiness simultaneously. And the efficient performance of HEDA-DEV is verified through comparison experiments with five efficient algorithms on two test sets. The main work of this paper are as below: (1) a multi-objective optimization model for the DAFJSP is established; (2) a novel multidimensional coding method is proposed, which is consistent with the characteristics of the problem; (3) two mutation operators and a similarity coefficient based on probability matrix are designed to implement the dynamic mutation; (4) considering both the characteristics of the problem and the objective features, the multiple neighborhood structures which satisfy the cooperative search strategies are developed.

In future research, we will consider the unexpected situations in actual production, such as machine failure and transportation accident, so as to be closer to reality. In addition, we are ready to consider the integration of batch scheduling into DAFJSP.

CRediT authorship contribution statement

Baigang Du: Writing – review & editing, Supervision, Investigation, Funding acquisition. **Shuai Han:** Writing – original draft, Methodology. **Jun Guo:** Writing – review & editing, Validation, Funding acquisition. **Yibing Li:** Supervision.

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.

Data availability

Data will be made available on request.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 51705386); China Scholarship Council (No.2016 06955091).

References

Behnamian, J., Fatemi Ghomi, S.M.T., 2016. A survey of multi-factory scheduling. J. Intell. Manuf. 27 (1), 231–249.

Brandimarte, P., 1993. Routing and scheduling in a flexible job shop by tabu search. Ann. Oper. Res. 41 (3), 157–183.

Cai, J., Zhou, R., Lei, D., 2020. Dynamic shuffled frog-leaping algorithm for distributed hybrid flow shop scheduling with multiprocessor tasks. Eng. Appl. Artif. Intell. 90, 103540.

Cao, Y., Shi, H., Han, Z., 2017. Multi-objective flexible job shop scheduling problem using differential evolution algorithm. In: 2017 9th International Conference on Modelling, Identification and Control (ICMIC). IEEE, pp. 521–526.

Chang, H.C., Liu, T.K., 2017. Optimisation of distributed manufacturing flexible job shop scheduling by using hybrid genetic algorithms. J. Intell. Manuf. 28 (8), 1973–1986.

Ge, H., Sun, L., Chen, X., Liang, Y., 2016. An efficient artificial fish swarm model with estimation of distribution for flexible job shop scheduling. Int. J. Comput. Intell. Syst. 9 (5), 917–931.

Gonzalez-Neira, E.M., Ferone, D., Hatami, S., Juan, A.A., 2017. A biased-randomized simheuristic for the distributed assembly permutation flowshop problem with stochastic processing times. Simulat. Model. Pract. Theor. 79, 23–36.

Guo, S., Luo, W., Xu, W., Wang, L., 2020. Research on distributed flexible job shop scheduling problem for large equipment manufacturing enterprises considering energy consumption. In: 2020 39th Chinese Control Conference (CCC). IEEE, pp. 1501–1506.

Hamzadayı, A., 2020. An effective benders decomposition algorithm for solving the distributed permutation flowshop scheduling problem. Comput. Oper. Res. 123, 105006.

Hao, X., Gen, M., Lin, L., Suer, G.A., 2017. Effective multiobjective EDA for bi-criteria stochastic job-shop scheduling problem. J. Intell. Manuf. 28 (3), 833–845.

Hatami, S., Ruiz, R., Andres-Romano, C., 2013. The distributed assembly permutation flowshop scheduling problem. Int. J. Prod. Res. 51 (17), 5292–5308.

Hatami, S., Ruiz, R., Andrés-Romano, C., 2015. Heuristics and metaheuristics for the distributed assembly permutation flowshop scheduling problem with sequence dependent setup times. Int. J. Prod. Econ. 169, 76–88.

He, L., Li, W., Zhang, Y., Cao, Y., 2019. A discrete multi-objective fireworks algorithm for flowshop scheduling with sequence-dependent setup times. Swarm Evol. Comput. 51, 100575.

He, L., Chiong, R., Li, W., Dhakal, S., Cao, Y., Zhang, Y., 2021. Multiobjective optimization of energy-efficient job-shop scheduling with dynamic reference point-based fuzzy relative entropy. IEEE Trans. Ind. Inf. 18 (1), 600–610.

He, L., Cao, Y., Li, W., Cao, J., Zhong, L., 2022a. Optimization of energy-efficient open shop scheduling with an adaptive multi-objective differential evolution algorithm. Appl. Soft Comput. 118, 108459.

He, L., Chiong, R., Li, W., 2022b. Energy-efficient open-shop scheduling with multiple automated guided vehicles and deteriorating jobs. Journal of Industrial Information Integration 30, 100387.

Hsu, C.Y., Kao, B.R., Lai, K.R., 2016. Agent-based fuzzy constraint-directed negotiation mechanism for distributed job shop scheduling. Eng. Appl. Artif. Intell. 53, 140–154.

Huang, J.P., Pan, Q.K., Miao, Z.H., Gao, L., 2021. Effective constructive heuristics and discrete bee colony optimization for distributed flowshop with setup times. Eng. Appl. Artif. Intell. 97, 104016.

Lei, D., Yuan, Y., Cai, J., 2021. An improved artificial bee colony for multi-objective distributed unrelated parallel machine scheduling. Int. J. Prod. Res. 59 (17), 5259–5271.

Li, J.Q., Pan, Q.K., Gao, K.Z., 2011. Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems. Int. J. Adv. Des. Manuf. Technol. 55, 1159–1169.

Li, W., He, L., Cao, Y., 2021. Many-objective evolutionary algorithm with reference point-based fuzzy correlation entropy for energy-efficient job shop scheduling with limited workers. IEEE Trans. Cybern. 52 (10), 10721–10734.

Li, X., Liu, Y., 2018. Approach of solving dual resource constrained multi-objective flexible job shop scheduling problem based on MOEA/D. International Journal of Online Engineering 14 (7), 75–89.

Liu, S., Pei, J., Cheng, H., Liu, X., Pardalos, P.M., 2019. Two-stage hybrid flow shop scheduling on parallel batching machines considering a job-dependent deteriorating effect and non-identical job sizes. Appl. Soft Comput. 84, 105701.

Lin, C.S., Li, P.Y., Wei, J.M., Wu, M.C., 2020. Integration of process planning and scheduling for distributed flexible job shops. Comput. Oper. Res. 124, 105053.

Mavrotas, G., 2009. Effective implementation of the ε-constraint method in multi-objective mathematical programming problems. Appl. Math. Comput. 213 (2), 455–465.

Mönch, L., Shen, L., 2021. Parallel machine scheduling with the total weighted delivery time performance measure in distributed manufacturing. Comput. Oper. Res. 127, 105126.

Montgomery, D.C., 2005. Design and analysis of experiments. John Wiley & Sons, Arizona.

Naderi, B., Azab, A., 2014. Modeling and heuristics for scheduling of distributed job shops. Expert Syst. Appl. 41 (17), 7754–7763.

Pan, Q.K., Gao, L., Xin-Yu, L., Jose, F.M., 2019. Effective constructive heuristics and meta-heuristics for the distributed assembly permutation flowshop scheduling problem. Appl. Soft Comput. 81, 105492.

Rifai, A.P., Nguyen, H.T., Dawal, S.Z.M., 2016. Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling. Appl. Soft Comput. 40, 42–57.

Şahman, M.A., 2021. A discrete spotted hyena optimizer for solving distributed job shop scheduling problems. Appl. Soft Comput. 106, 107349.

Shao, Z., Pi, D., Shao, W., 2020. Hybrid enhanced discrete fruit fly optimization algorithm for scheduling blocking flow-shop in distributed environment. Expert Syst. Appl. 145, 113147.

Storn, R., Price, K., 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. 11 (4), 341–359.

Sun, X., Guo, S., Guo, J., Du, B., 2019. A hybrid multi-objective evolutionary algorithm with heuristic adjustment strategies and variable neighbor-hood search for flexible job-shop scheduling problem considering flexible rest time. IEEE Access 7, 157003–157018.

Sun, L., Lin, L., Li, H., Gen, M., 2019. Large scale flexible scheduling optimization by a distributed evolutionary algorithm. Comput. Ind. Eng. 128, 894–904.

Wang, S.Y., Wang, L., Liu, M., Xu, Y., 2013a. An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem. Int. J. Prod. Econ. 145 (1), 387–396.

Wang, S.Y., Wang, L., Liu, M., Xu, Y., 2015. An order-based estimation of distribution algorithm for stochastic hybrid flow-shop scheduling problem. Int. J. Comput. Integrated Manuf. 28 (3), 307–320.

Wang, Y., Shi, L., Zhang, C., Fu, L., Jin, X., 2020. NSGA-II algorithm and application for multi-objective flexible workshop scheduling. J. Algorithm Comput. Technol. 14, 1748302620942467.

Wu, G., Mallipeddi, R., Suganthan, P.N., Wang, R., Chen, H., 2016. Differential evolution with multi-population based ensemble of mutation strategies. Inf. Sci. 329, 329–345.

Wang, S., Wang, L., Xu, Y., Liu, M., 2013b. An effective estimation of distribution algorithm for the flexible job-shop scheduling problem with fuzzy processing time. Int. J. Prod. Res. 51 (12), 3778–3793.

Wu, M.C., Lin, C.S., Lin, C.H., Chen, C.F., 2017. Effects of different chromosome representations in developing genetic algorithms to solve DFJS scheduling problems. Comput. Oper. Res. 80, 101–112.

Wu, X., Liu, X., Zhao, N., 2019. An improved differential evolution algorithm for solving a distributed assembly flexible job shop scheduling problem. Memetic Computing 11 (4), 335–355.

Yang, S., Xu, Z., 2021. The distributed assembly permutation flowshop scheduling problem with flexible assembly and batch delivery. Int. J. Prod. Res. 59 (13), 4053–4071.

Ying, K.C., Lin, S.W., 2018. Minimizing makespan for the distributed hybrid flowshop scheduling problem with multiprocessor tasks. Expert Syst. Appl. 92, 132–141.

Zhang, G., Xing, K., 2018. Memetic social spider optimization algorithm for scheduling two-stage assembly flowshop in a distributed environment. Comput. Ind. Eng. 125, 423–433.

Zhang, G., Xing, K., Cao, F., 2018. Scheduling distributed flowshops with flexible assembly and set-up time to minimise makespan. Int. J. Prod. Res. 56 (9), 3226–3244.

Zhang, Z.Q., Hu, R., Qian, B., Jin, H.P., Wang, L., Yang, J.B., 2022. A matrix cube-based estimation of distribution algorithm for the energy-efficient distributed assembly permutation flow-shop scheduling problem. Expert Syst. Appl. 194, 116484.