Expert Systems With Applications 194 (2022) 116484

Contents lists available at ScienceDirect

Expert Systems With Applications

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

A matrix cube-based estimation of distribution algorithm for the energy-efficient distributed assembly permutation flow-shop scheduling problem

Zi-Qi Zhang [a,b], Rong Hu [a,c,*], Bin Qian [a,b,c], Huai-Ping Jin [a], Ling Wang [d], Jian-Bo Yang [e]

[a] School of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500, China
[b] School of Mechanical and Electrical Engineering, Kunming University of Science and Technology, Kunming 650500, China
[c] Yunnan Key Laboratory of Artificial Intelligence, Kunming University of Science and Technology, Kunming 650500, China
[d] Department of Automation, Tsinghua University, Beijing 100084, China
[e] Alliance Manchester Business School, The University of Manchester, Manchester M15 6PB, United Kingdom

ARTICLE INFO

Keywords:
Energy-efficient scheduling
Estimation of distribution algorithm
Distributed flowshop scheduling
Assembly line
Low-carbon manufacturing

ABSTRACT

In this paper, a matrix-cube-based estimation of distribution algorithm (MCEDA) is proposed to solve the energy-efficient distributed assembly permutation flow-shop scheduling problem (EE_DAPFSP) that minimizes both the maximum completion time (C_{max}) and the total carbon emission (TCE) simultaneously. Firstly, a high-quality and diverse initial population is constructed via a hybrid initialization method. Secondly, a matrix-cube-based probabilistic model and its update mechanism are designed to appropriately accumulate the valuable pattern information from superior solutions. Thirdly, a suitable sampling strategy is developed to sample the probabilistic model to generate a new population per generation, so as to guide the search direction toward promising regions in solution space. Fourthly, a problem-dependent neighborhood search based on critical path is provided to perform an in-depth local search around the promising regions found by the global search. Fifthly, two types of speed adjustment strategies based on problem properties are also embedded to further improve the quality of the obtained solutions. Sixthly, the influence of the parameters is investigated based on the multi-factor analysis of variance of Design-of-Experiments. Finally, extensive experiments and comprehensive comparisons with several recent state-of-the-art multi-objective algorithms are carried out based on the well-known benchmark instances, and the statistical results demonstrate the efficiency and effectiveness of the proposed MCEDA in addressing the EE_DAPFSP.

1. Introduction

With growing worldwide concern about the global warming and climate change, it is imperative that all of the developed and developing countries around the world should enact laws and take energy-efficient measures or technologies to relief global environment and energy crisis (May et al., 2015). Low-carbon or energy-efficient manufacturing is the key measure and inexorable choice to realize the integration of environmental sustainability and economic development. Meanwhile, along with the deepening of economic globalization, modern production pattern of many enterprises has a tendency to change from the traditional centralized manufacturing mode to the *trans*-regional decentralized manufacturing mode. Under these backgrounds, the research on energy-efficient and distributed production scheduling problems has great practical and engineering significance.

Among the distributed scheduling problems, the distributed assembly permutation flow-shop scheduling problem (DAPFSP) is widely encountered in advanced manufacturing systems and modern supply chains. In fact, many real-life scheduling problems in manufacturing enterprises can be modeled as the DAPFSP. For example, in some large Chinese enterprises manufacturing automobile engines (e.g., Wei chai Power Co., Ltd. and Yu chai Group), all parts of each engine, such as cylinder block, cylinder head, and crankshaft, are first allocated to different factories or flow shops for processing, and then these parts are assembled into the final automobile engine in an assembly shop. Because the DAPFSP has been proved to be NP-hard with strong sense (Hatami et al., 2013), the relationship between its inherent geometric structure and the optimal solution is still an open problem, and there is

* Corresponding author.
E-mail address: ronghu@vip.163.com (R. Hu).

https://doi.org/10.1016/j.eswa.2021.116484
Received 20 July 2021; Received in revised form 28 December 2021; Accepted 28 December 2021
Available online 10 January 2022

no algorithm that can obtain the optimal solution in polynomial time. The mathematical programming algorithms need to traverse or partially traverse the DAPFSP's solution space, which makes them limited due to the long running time in solving medium and large-scale problems. To tackle this issue, the following metaheuristics have been presented in recent years to obtain satisfactory solutions for the DAPFSP and its variant under different scales within several seconds or tens of seconds.

As for DAPFSP, a fast variable neighborhood descent (VND) method (Hatami et al., 2013), an estimation of distribution algorithm-based memetic algorithm (EDAMA) (Wang & Wang, 2016), an effective hybrid biogeography-based optimization (HBBO) (Lin & Zhang, 2016), a backtracking search hyper-heuristic (BS-HH) (Lin et al., 2017), three improved discrete invasive weed optimization algorithms (DIWOs) (Sang et al., 2019) and a matrix cube-based estimation of distribution algorithm (MCEDA) (Zhang et al., 2021) have been proposed. As for the variant of the DAPFSP, Pan et al. (2019) considered a series of identical factories in which each factory consists of a flow shop for job processing and an assembly line for product processing, and then presented three effective constructive heuristics and an enhance iterated greedy algorithm (IG). However, the above studies of DAPFSP only consider the efficiency-oriented criteria, and no existing studies involve energy conservation. Therefore, this paper aims to solve the energy-efficient DAPFSP (EE_DAPFSP) with the criteria of minimizing the maximum completion time (C_{max}) and the total carbon emission (TCE) at the same time.

The considered EE_DAPFSP is more complex and general than the DAPFSP, and the latter reduces to the former. This means that the EE_DAPFSP is also a NP-hard problem in strong sense. Obviously, it is a challenge to design an effective algorithm to address this problem. Among the existing metaheuristics, the estimation of distribution algorithm (EDA) is a special one. Unlike the crossover and mutation operators in most traditional metaheuristics, EDA builds one or more probabilistic models to learn the valuable information of the structure patterns of superior solutions, and generates the next offspring population by sampling these models. Such a new population generation mechanism can avoid the destruction of the building blocks (the partial structure patterns) in superior individuals or solutions to a certain extent (Larrañaga & Lozano, 2001). Due to its good exploration ability, inherent parallelism and quick convergence, EDA has been applied to deal with different kinds of scheduling problem, e.g., the permutation flow-shop scheduling problem (PFSP) (Jarboui et al., 2009), the multi-objective PFSP (Tiwari et al., 2014), the lot-streaming flow-shop scheduling problem (Pan & Ruiz, 2012), the flexible job-shop scheduling problem (Wang et al., 2012), and the DAPFSP (Wang & Wang, 2016; Zhang et al., 2021).

In the above EDAs, the two-dimensional probability model or matrix is used to store the information of the blocks and the order of jobs from each superior solution or individual. Here one block consists of any two consecutive jobs in a solution. Obviously, the structure of matrix determines that only the matrix elements and the subscripts of these elements can be used to store information. For the two-dimensional matrix, each element is used to save the occurrence frequency or probability that the job C appears immediately after the job R, and its subscript [R, C] is only enough to save the corresponding block's pattern. There is no extra space to record the position of this block. This causes the sampling procedure may misplace the blocks in new individuals. As a result, the search direction cannot be reasonably guided, and the actual performance of these existing EDAs is limited (refer to Subsection 3.2 for more details). To overcome this defect, a novel EDA with a matrix-cube-based or three-dimensional probability model, namely MCEDA, is designed for the EE_DAPFSP. The test results on the instances with different scales demonstrate that MCEDA can obtain better solution than state-of-the-art algorithms under the same running time.

The main features of MCEDA lie in four aspects: the high-quality initial population generated by a hybrid initialization strategy, the global search guided by a three-dimensional probabilistic model, the deep local search driven by a multi-neighborhood search, and the solution quality enhanced via two speed adjustment strategies. In terms of the initial population, a hybrid initialization strategy combining an effective constructive heuristic method and a randomization method is devised to generate high-quality initial population. This strategy can make the algorithm's search start from some regions close to the promising regions. In terms of the global search, a three-dimensional probabilistic model with an update mechanism is designed to reasonably reserve the valuable information of superior individuals, and a special sampling strategy is devised to guide the search to the promising regions in solution space. Since the three-dimensional structure is utilized to accurately record the job blocks with their exact positions, the global search can be effectively guided to the truly promising regions. In terms of the local search, an efficient neighborhood search adopting four critical path-based neighborhood structures is developed to execute deep search from the promising regions obtained by the global search. In terms of the solution quality, the problem properties are analyzed and two types of the problem-specific speed adjustment strategies are proposed to further improve the quality of the obtained non-dominated solutions. The novelties of this paper are summarized in Table 1.

Table 1
Difference between this work and the previous literature.

Difference	The previous literature	This work
Problem formulation	The existing formulations of the DAPFSP only take account of traditional efficiency-dependent criteria (Hatami et al., 2013; Lin & Zhang, 2016; Wang & Wang, 2016; Lin et al., 2017; Pan et al., 2019; Zhang et al., 2021).	The sequence model of the EE_DAPFSP is established by minimizing both efficiency-dependent and energy-saving criteria. This is the first time that the TCE (*i.e.*, an energy-saving criterion) has been treated as a separate criterion in the DAPFSPs.
Global search framework	The traditional metaheuristics for the DAPFSP only employ conventional genetic operators (*i.e.*, the selection, crossover and mutation) or common neighborhood operators (*i.e.*, insert, interchange and swap) to generate offspring to execute exploration (Lin & Zhang, 2016; Lin et al., 2017; Pan et al., 2019; Sang et al., 2019). Furthermore, most of existing EDAs for scheduling problem use the two-dimensional probability model to save the information of superior individuals (Jarboui et al., 2009; Pan & Ruiz, 2012; Tiwari et al., 2014; Wang & Wang, 2016).	A novel EDA with a matrix-cube-based or three-dimensional probability model is utilized to reasonably reserve the valuable information of superior individuals and effectively guide the search direction.
Speed adjustment strategy	Most literatures only consider designing energy- saving speed adjustment strategies to reduce energy-saving criteria without reducing efficiency-dependent criteria, so as to enhance the quality of current non-dominated solutions (Ding et al., 2016; Chen et al., 2019; Abedi et al., 2020; Jiang & Zhang, 2019; Wang & Wang, 2020).	In addition to designing an energy-saving speed adjustment strategy to enhance the quality of each current non-dominated solution, a reduced-time speed adjustment strategy is also designed to generate possible non-dominated solutions, so as to further increase the diversity and number of non-dominated solutions.

The remainder of this paper is organized as follows. Section 2 describes and formulates the EE_DAPFSP. Section 3 introduces the proposed MCEDA in detail. Section 4 carries out computational comparisons and statistical analyses. Finally, some conclusions and future works are provided in Section 5.

Table 2
The notations used in the permutation-based model of the EE_DAPFSP.

Indices	
i	The index for jobs where $i = 1,2,\ldots,n$.
j	The index for machines where $j = 1,2,\ldots,m$.
h	The index for products where $h = 1,2,\ldots,S$.
f	The index for factories where $f = 1,2,\ldots,F$.
k	The index for velocities where $k = 1,2,\ldots,d$.
Parameters	
n	The total number of jobs.
m	The total number of machines.
F	The total number of factories.
S	The total number of products.
d	The total number of velocities.
J	The set of jobs, *i.e.*, $J = \{J_1, J_2, \ldots, J_n\}$.
M	The set of machines, *i.e.*, $M = \{M_1, M_2, \ldots, M_m\}$ where $\lvert M_j\rvert \geqslant 2$.
P	The set of products, *i.e.*, $P = \{P_1, P_2, \ldots, P_S\}$.
O	The set of operations, *i.e.*, $O = \{O_{i,1}, O_{i,2}, \ldots, O_{i,m}\}$.
V	The set of velocities, *i.e.*, $V = \{v_1, v_2, \ldots, v_d\}$.
N_h	The number of jobs belongs to product P_h.
$p_{i,j}$	The processing time of operation $O_{i,j}$ on machine M_j.
p_h^A	The assembling time of product h on machine M_A.
Variables	
n_f	The total number of jobs assigned to factory f, where $\sum_{f=1}^{F} n_f = n$.
ω_h	The total number of jobs of product P_h, where $\sum_{h=1}^{S} \omega_h = n$.
$\boldsymbol{\pi}$	The total sequence of jobs, *i.e.*, $\boldsymbol{\pi} = [\pi_1, \pi_2, \ldots, \pi_n]$.
$\boldsymbol{\pi}_f$	The sub-sequence of jobs in factory f, *i.e.*, $\boldsymbol{\pi}^f = [\pi_1^f, \pi_2^f, \ldots, \pi_{n_f}^f]$.
$\boldsymbol{\pi}_p$	The sequence of assembled products.
$\boldsymbol{\lambda}$	The assembly sequence of products, *i.e.*, $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_S]$.
$\widehat{p}_{i,j}^k$	The actual processing time of $O_{i,j}$ on M_j at the speed v_k where $\widehat{p}_{i,j}^k = p_{i,j}/v_k$.
$C_{i,j}$	The completion time of $O_{i,j}$ on M_j.
S_h^A	The earliest possible assembly time of product h on the assembly line.
C_h^A	The completion time of product h on the assembly line.
E_{jk}	The energy consumption per unit time of machine M_j running at speed v_k.
SE_j	The energy consumption per unit time when machine M_j is in standby state.
ε	The coefficient between energy consumption and carbon emission, where $\varepsilon = 0.7559$. ε refers to the carbon emission per unit of consumed energy (kilogram CO_2 equivalent/kiloWatthour).
$C_{max}(\boldsymbol{\pi}, \boldsymbol{V})$	The makespan of a feasible solution $(\boldsymbol{\pi}, \boldsymbol{V})$.
$\mathrm{TCE}(\boldsymbol{\pi}, \boldsymbol{V})$	The total carbon emission of a feasible solution $(\boldsymbol{\pi}, \boldsymbol{V})$.
$(\boldsymbol{\Pi}, \boldsymbol{\Sigma})$	A set of feasible scheduling schemes for the problem considered.

2. Problem statement

2.1. Energy-efficient distributed assembly permutation flow-shop scheduling problem

The EE_DAPFSP can be briefly described as follows. There are S products which consist of a set of n jobs. Each of the n jobs from the set $J = \{J_1, J_2, \ldots, J_n\}$ is allocated to one of the F factories and is to be processed sequentially through m machines $M = \{M_1, M_2, \ldots, M_m\}$, and then these n jobs are to be assembled into S products. Each product consists of a series of specific jobs and each job belongs to one certain product. The production process mainly consists of two stages, namely the processing stage and the assembly stage. The processing stage consists of F factories, and each factory is regarded as a flow shop composed of the same number of machines, *i.e.*, m heterogeneous machines with different functions. Each machine has d discrete and adjustable processing speeds, *i.e.*, $V = \{v_1, v_2, \ldots, v_d\}$. Each job $J_i \in J$ has a predetermined processing time $p_{i,j}$ on every machine $M_j \in M$, and a series of m operations $[O_{i,1}, O_{i,2}, \ldots, O_{i,m}]$ of J_i can be completed in any factory, where the actual processing time of $O_{i,j}$ on M_j at speed $v_k \in V$ is $\widehat{p}_{i,j}^k = p_{i,j}/v_k$. Then, the corresponding energy consumption is produced whether the machine is in the processing state or in the standby state. Once all of the jobs of the specific product have been processed in factories, the assembly process can be started on the assembly machine. The EE_DAPFSP considered contains three subproblems, namely, appropriately assign jobs to factories, suitable select the processing speed of machines, and reasonably determine the processing order and the assembly order of jobs and products. The notations description for the EE_DAPFSP are provided in Table 2, and the illustration of the EE_DAPFSP is shown in Fig. 1.

In addition, all assumptions of FSP also meet herein. (i) No release time is considered. Each job and machine are available and independent, and each job can be processed immediately in time; (ii) No machine breakdown or setup time is considered. Preemption and interruption are not allowed in the processing; (iii) No transportation time is considered. Each product can be assembled once its jobs have been processed; (iv) Each job cannot be changed the factory and all operations of each job should be processed in the same factory, once the job assignment has been determined. At any time, each job can only be processed on at most one machine, and each machine is allowed to process no more than one job; (v) The processing time is deterministic, and the processing speed of machines remain unchanged during the processing process. The EE_DAPFSP aim to determine the allocation of jobs in the factories, the processing order of jobs on machines and the assembly order of each product, and the processing speed of all the machines. The production efficiency criterion needs to minimize the makespan, which is defined as the completion time of the last product on the assembly machine. The energy consumption criterion needs to minimize the total carbon

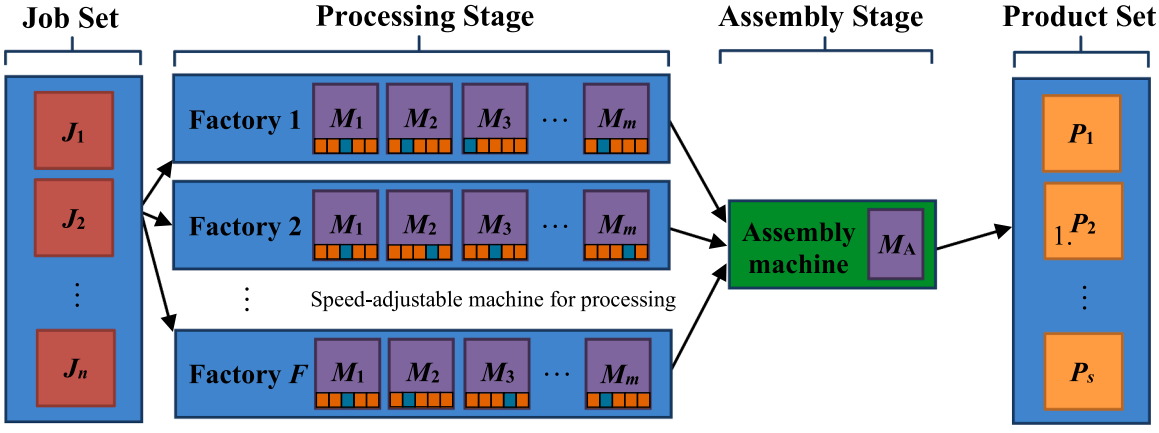


Fig. 1. Illustration of the EE_DAPFSP.

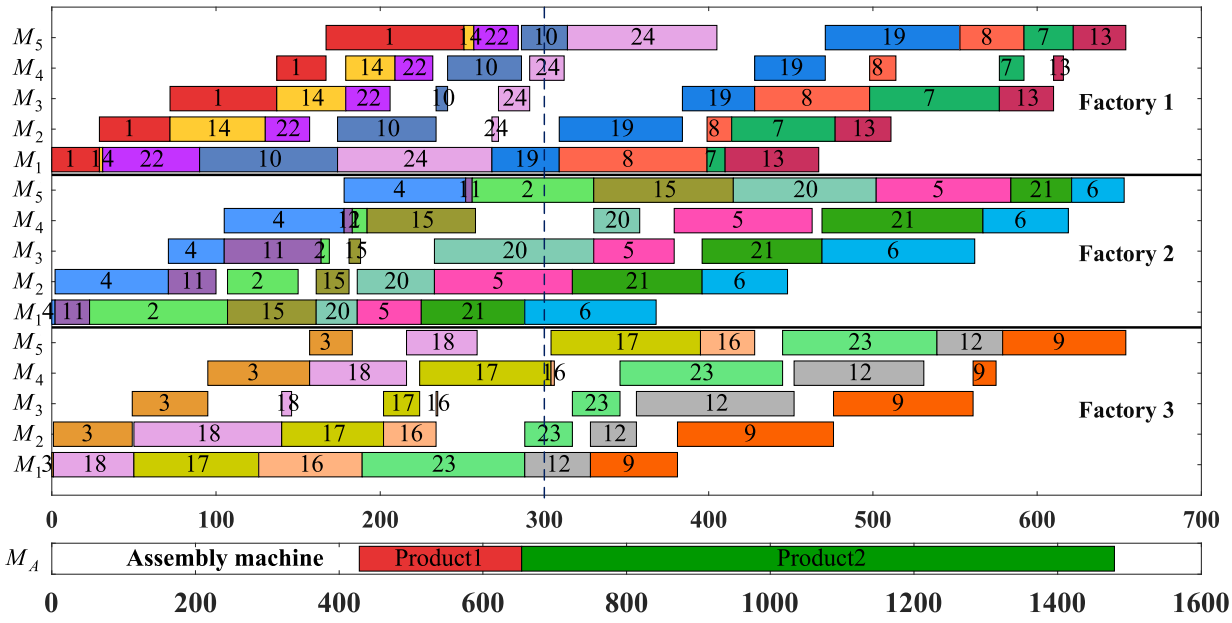


Fig. 2. The Gantt chart of a feasible solution of the EE_DAPFSP.

emission throughout the manufacturing period. The goal of the EE_DAPFSP is to find a group of optimal scheduling schemes with minimizing of both makespan and total carbon emission. According to the above description, the permutation-based model of the EE_DAPFSP can be given as follows.

$$C_{\pi_1^f,1} = \widehat{p}^k_{\pi_1^f,1}, f = 1,2,...,F; k = 1,2,...,d. \tag{1}$$

$$C_{\pi_i^f,1} = C_{\pi_{i-1}^f,1} + \widehat{p}^k_{\pi_i^f,1}, i = 2,3,...,n_f; f = 1,2,...,F; k = 1,2,...,d. \tag{2}$$

$$C_{\pi_1^f,j} = C_{\pi_1^f,j-1} + \widehat{p}^k_{\pi_1^f,j}, j = 2,3,...,m; f = 1,2,...,F; k = 1,2,...,d. \tag{3}$$

$$C_{\pi_i^f,j} = max\{C_{\pi_{i-1}^f,j}, C_{\pi_i^f,j-1}\} + \widehat{p}^k_{\pi_i^f,j}, \quad i = 2,3,...,n_f; j = 2,3,...,m; f = 1,2,...,F; k = 1,2,...,d. \tag{4}$$

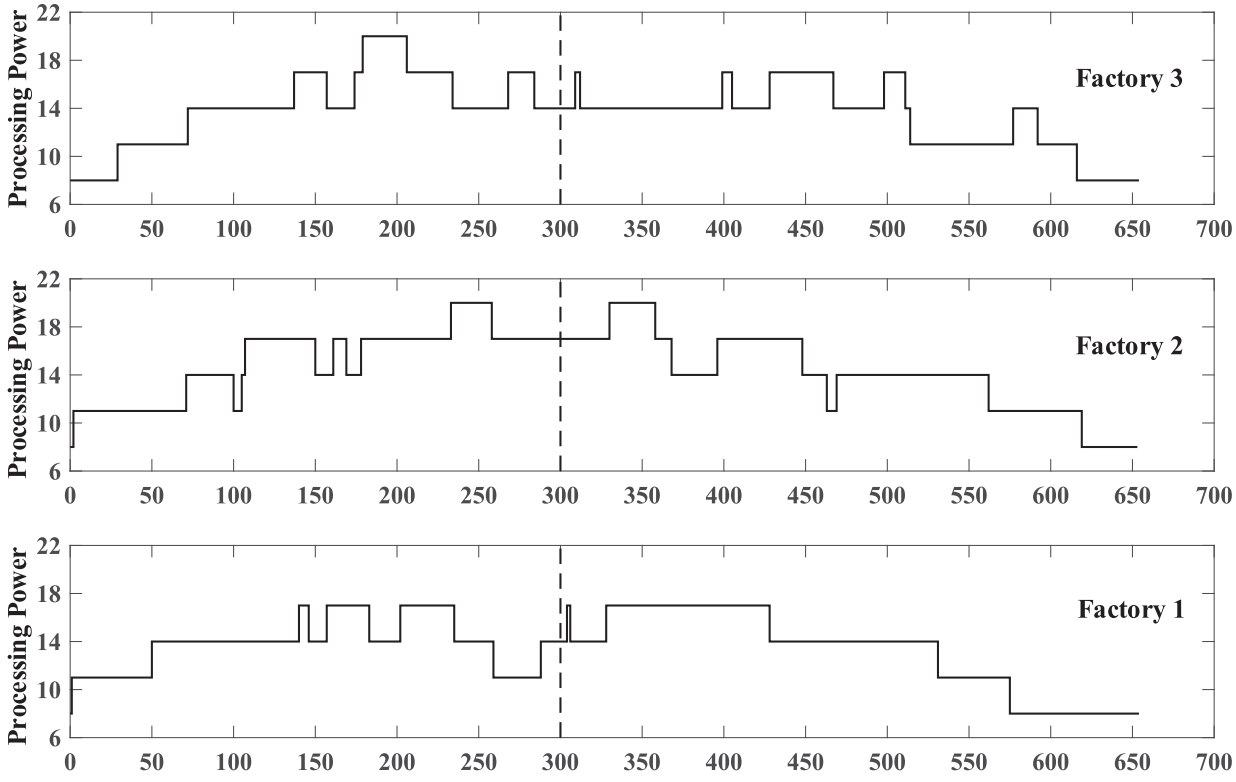


Fig. 3. The real-time power consumption curve of the EE_DAPFSP.

$$S^A_{\lambda_h} = \max_{\pi^f_i \in \lambda_h} C_{\pi^f_i, m}, i = 1, 2, ..., n_f;\ f = 1, 2, ..., F; h = 1, 2, ..., S. \tag{5}$$

$$C^A_{\lambda_1} = S^A_{\lambda_1} + p^A_{\lambda_1}. \tag{6}$$

$$C^A_{\lambda_h} = \max\{ C^A_{\lambda_{h-1}}, S^A_{\lambda_h}\} + p^A_{\lambda_h}, h = 2, 3, ..., S. \tag{7}$$

$$C_{max}(\boldsymbol{\pi}, \boldsymbol{V}) = C^A_{\lambda_S}. \tag{8}$$

$$\mathrm{TCE}(\boldsymbol{\pi}, \boldsymbol{V}) = \varepsilon \int_0^{C_{max}} \left(\sum_{j \in M} E_{jk} x^t_{jk} + \sum_{j \in M} SE_j y^t_j \right) dt, k \in \{ 1, 2, ..., d\} . \tag{9}$$

$$(\boldsymbol{\pi}^*, \boldsymbol{V}^*) = arg \min_{\boldsymbol{\pi} \in \boldsymbol{\Pi}, \boldsymbol{V} \in \boldsymbol{\Sigma}} \{C_{max}(\boldsymbol{\pi}, \boldsymbol{V}), \mathrm{TCE}(\boldsymbol{\pi}, \boldsymbol{V})\}. \tag{10}$$

Eqs. (8) and (9) are the calculation formulas of the maximum completion time (C_{max}) and the total carbon emission (TCE) respectively. The formula of the TCE in Eq. (9) is divided into two parts, the first part is the total carbon emission when the machines are in processing state, and the second part is the total carbon emission when the machines are in standby state. The x^t_{jk} and y^t_j in Eq. (9) are binary variables, indicating that if machine M_j is running at speed v_k at the time t, then $x^t_{jk} = 1$, and $x^t_{jk} = 0$ otherwise. In addition, if machine M_j is in the standby state at time t, then $y^t_j = 1$, and $y^t_j = 0$ otherwise. Eq. (10) is the final goal of the considered EE_DAPFSP. That is, the criteria are to find each optimal non-dominated solution $(\pi^*, \boldsymbol{V}^*)$ in the scheduling scheme set $(\boldsymbol{\Pi}, \boldsymbol{\Sigma})$, so that a suitable balance between the maximum completion time (C_{max}) and the total carbon emission (TCE) is achieved.

For ease of understanding, Fig. 2 illustrates the Gantt chart of a feasible solution of the EE_DAPFSP with 24 jobs, 5 machines, three factories and two products. The first product P_1 contains 13 jobs, (*i.e.*, $J_1 J_4, J_{10}, J_{11}, J_{14} J_{18}, J_{22}, J_{24}$), while the second product P_2 consists of 11 jobs, (*i.e.*, $J_5 J_9, J_{12}, J_{13}, J_{19} J_{21}, J_{23}$). The sub-sequences of all the jobs assigned to the three factories are $\boldsymbol{\pi_1} = [3, 18, 17, 16, 23, 12, 9]$, $\boldsymbol{\pi_2} = [4, 11, 2, 15, 20, 5, 21, 6]$ and $\pi_3 = [1, 14, 22, 10, 24, 19, 8, 7, 13]$, respectively. Fig. 3 shows the three curves of the real-time power consumption corresponding to the Gantt chart of the feasible solution provided in Fig. 2. It can be seen from Fig. 2 that at time 300, M_1, M_2, and M_4 in factory 1 are all in the processing state, while M_3 and M_5 are in the standby state. Suppose that M_1, M_2, and M_4 are running at speeds $v_{k1} \in V$, $v_{k2} \in V$ and $v_{k4} \in V$ at this time, and thus the instantaneous processing power of these three machines are E_{1k1}, E_{2k2} and E_{4k4}, respectively. In addition, the power consumption of M_3 and M_5 in the standby state are SE_3 and SE_5, and then the overall power consumption at time 300 in factory 1 can be calculated by summing up the power of the five machines, (*i.e.*, $E_{1k1} + E_{2k2} + SE_3 + E_{4k4} + SE_5$), as shown in Fig. 3. From Fig. 3, at time 300, M_1, M_2, M_3 and M_5 in factory 2 are in the processing state and only M_4 is in the standby state, while M_1, M_4 and M_5 in factory 3 are in the processing state and M_2 and M_3 are in the standby state. Similarly, the real-time power consumption of factory 2 and factory 3 can be calculated at this moment. Therefore, the real-time power consumption of each factory at different time points can be calculated respectively. Note that the total energy consumption of Eq. (9) can be represented as the sum of the area enclosed by the real-time power curve corresponding to each factory and the time axis, and thus the total carbon emission (TCE) can be computed via the coefficient ε between energy consumption and carbon emission.

It should be noted that compared to the DAPFSP studied by Hatami et al. (2013), where the objective is to minimize the makespan and the processing time is regarded as constant, the EE_DAPFSP considered not only adds a green scheduling objective of minimizing the total carbon emission, but also sets all of the processing times to be adjustable (depending on the actual running speed of the machine). Therefore, motivated by the energy saving consideration, this paper investigated the modeling and solving of the energy-efficient DAPFSP, which is a more complex (with more decision variables) multi-objective optimization problem than the typical DAPFSP. The main difficulty of multi-objective optimization lies in the large and complex feasible solution space, that is, the variable processing speed will further enhance the difficulty of solving the considered problem, and obtaining high-quality solutions often has a high-level requirement of both the understanding of the structural features of the problem and the search efficiency of the algorithm. Moreover, since the EE_DAPFSP is an NP-hard problem, it is not theoretically guaranteed to find the optimal scheduling scheme in polynomial time. Therefore, for medium and large-scale instances, the scheduling goal is to obtain the near-optimal scheduling schemes within an acceptable time, and the metaheuristics are an effective way to solve the EE_DAPFSP.

2.2. Multi-objective optimization problem

The multi-objective optimization usually requires a satisfactory trade-off between two or more conflicting objectives, so that the multiple objective functions involved are optimized simultaneously. The EE_DAPFSP involved in this study needs to consider both production efficiency and energy consumption, and there is an obvious conflict relationship between these two criteria. The optimal or near-optimal scheduling schemes for the EE_DAPFSP is a set of feasible solutions, instead of a single solution. These solution sets must be determined by the dominance relationship via some specific multi-objective optimization techniques. Generally, the MOPs can be described as shown in Eq. (11).

$$\begin{aligned} &\mathrm{Min}\ \boldsymbol{F}(\boldsymbol{x}) = \{f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_m(\boldsymbol{x})\} \\ &s.t.\ g_l(\boldsymbol{x}) \geqslant 0; h_k(\boldsymbol{x}) = 0, l = 1, 2, ..., L; k = 1, 2, ..., K. \\ &x^{lb}_i \leqslant x_i \leqslant x^{ub}_i, i = 1, 2, ..., n. \end{aligned} \tag{11}$$

where $\boldsymbol{x} = [x_1, x_2, ..., x_n]^T \in \Omega$ is a decision vector composed of n decision variables, and $f_i(\boldsymbol{x})$ is the ith sub-objective function. The mapping function $\boldsymbol{F}(\boldsymbol{x})$ contains m sub-objective functions and $\boldsymbol{F}(\boldsymbol{x}) : \Omega \rightarrow \mathbb{R}^m$. Ω is the decision space and $\mathbb{R}^m$ is the m-dimensional objective space. x^{lb}_i and x^{ub}_i are the upper bound and lower bound for the ith dimensional variable x_i, respectively. Then, some basic concepts commonly used in MOPs are introduced below (Minella et al., 2008).

- **Pareto dominance**: For any two solutions $\boldsymbol{x} \in \Omega$ and $\boldsymbol{y} \in \Omega$, if and only if $\forall i \in \{1,2,...,m\}, f_i(\boldsymbol{x}) \leqslant f_i(\boldsymbol{y})$ and $\exists i' \in \{1,2,...,m\}, f_{i'}(\boldsymbol{x}) < f_{i'}(\boldsymbol{y})$, then the solution $\boldsymbol{x}$ dominates solution $\boldsymbol{y}$, denoted as $\boldsymbol{x} \prec \boldsymbol{y}$.
- **Pareto archive**: For any solution $\boldsymbol{x} \in \Omega$, $\boldsymbol{x}$ is Pareto optimal if and only if $\neg \exists \boldsymbol{y} \in \Omega$: $\boldsymbol{y} \prec \boldsymbol{x}$. The solution set composed of all Pareto optimal solutions is called Pareto optimal set or Pareto archive, denoted as Ω^*.

- **Pareto front**: The Pareto optimal front is aggregated by all of the solutions in Pareto optimal set Ω*, that is $PF = \{ \boldsymbol{F}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_m(\boldsymbol{x})]^{\mathrm{T}} | \boldsymbol{x} \in \Omega^* \}$.
- **Non-dominated solution**: In the feasible solution subset $A \subseteqq \Omega$, for each solution $\boldsymbol{x} \in A$, if and only if $\neg\exists \boldsymbol{y} \in A$: $\boldsymbol{x} \prec \boldsymbol{y}$ or $\boldsymbol{x} \succ \boldsymbol{y}$.
- **Non-dominated set**: For a feasible solution set $\Omega'' \subseteqq \Omega$, if there exists another non-dominated subset Ω' such that $\Omega' \subseteqq \Omega''$, $\forall \boldsymbol{a} \subseteqq \Omega'$ and $\boldsymbol{b} \subseteqq \Omega''$, $\neg\exists \boldsymbol{b} \prec \boldsymbol{a}$ and $\boldsymbol{a} \neq \boldsymbol{b}$, then Ω' is a non-dominated solution set.
- **Non-dominated sorting**: It is used to sort all obtained non-dominated solutions and divide them into different levels according to the Pareto dominance relationship (Deb et al., 2002).
- **Crowded distance**: It is used to measure the degree of dispersion among the non-dominated solutions on the same level (Ding et al., 2016; Deb et al., 2002). In general, the larger the crowding distance of a set of non-dominated solutions is, the better the dispersion of the non-dominated solution set is. The crowding distance calculation method is given in Algorithm 1.

In many practical production processes, decision makers usually choose their preferred scheduling solution from the obtained Pareto optimal front based on a reasonable trade-off between preferences or priorities of objectives. Therefore, the aim of MOPs is to obtain a variety of non-dominated solutions with good proximity and diversity with respect to the true Pareto optimal front.

Algorithm 1: Crowded distance calculation method

Input: t non-dominated solutions where $\mathbf{C} = (C^1_{\max}, C^2_{\max}, ..., C^t_{\max})^T$ and $\mathbf{E} = (\mathrm{TCE}^1, \mathrm{TCE}^2, ..., \mathrm{TCE}^t)^T$.

1: Sort $\mathbf{C}$ in ascending order, i.e., $(C^{I(1)}_{\max}, C^{I(2)}_{\max}, ..., C^{I(t)}_{\max})^T$, where $\boldsymbol{I} = (I(1), I(2), ..., I(t))^T$ and the vector I is the index of the ascending order.

2: Initialization: $C_{dis}(I(1)) := +\infty$ and $C_{dis}(I(t)) := +\infty$.

3: **for** $i = 2$ *to* $t-1$ **do**

4: $\quad C_{\max\_}C_{dis}(I(i)) \leftarrow |C^{I(i+1)}_{\max} - C^{I(i-1)}_{\max}| / (\max\{\mathbf{C}\} - \min\{\mathbf{C}\})$.

5: $\quad \mathrm{TCE}\_C_{dis}(I(i)) \leftarrow |\mathrm{TCE}^{I(i+1)} - \mathrm{TCE}^{I(i-1)}| / (\max\{\mathbf{E}\} - \min\{\mathbf{E}\})$.

6: $\quad C_{dis}(I(i)) \leftarrow C_{\max\_}C_{dis}(I(i)) + \mathrm{TCE}\_C_{dis}(I(i))$.

7: **end for**

Output: Crowded distance for each feasible solution $C_{dis}(i), i = 1, 2, ..., t$.

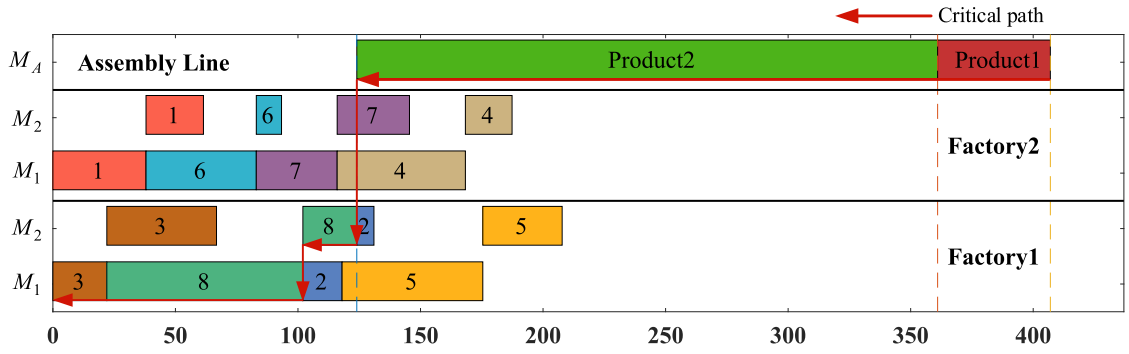
2.3. Problem property analysis

The EE_DAPFSP is a typical complex PFSP with adjustable machine speed and there is a close coupling relationship between the production and assembly stages. In EE_DAPFSP, the machine speed directly determines the processing operation, and the processing stage affects the assembly stage, that is, to determine the start time of each product in the assembly stage, the completion time of all parts to which the product belongs must be taken into account in the processing stage. Thus, the achievement of objectives and the handling of constraints must not only take into account the optimal production efficiency, but also meet the production requirements of low-carbon manufacturing. Conflicting constraints complicate the feasible solution space of the considered problem, resulting in a great challenge that the algorithm being more difficult to obtain a satisfactory solution within an acceptable time. Therefore, when designing EDA-based algorithms to address the EE_DAPFSP, it is not only necessary to establish effective probabilistic models to quickly guide directions to truly promising regions, but also need to analyze some problem-dependent properties and then utilize them to narrow the search scope by avoiding invalid searches.

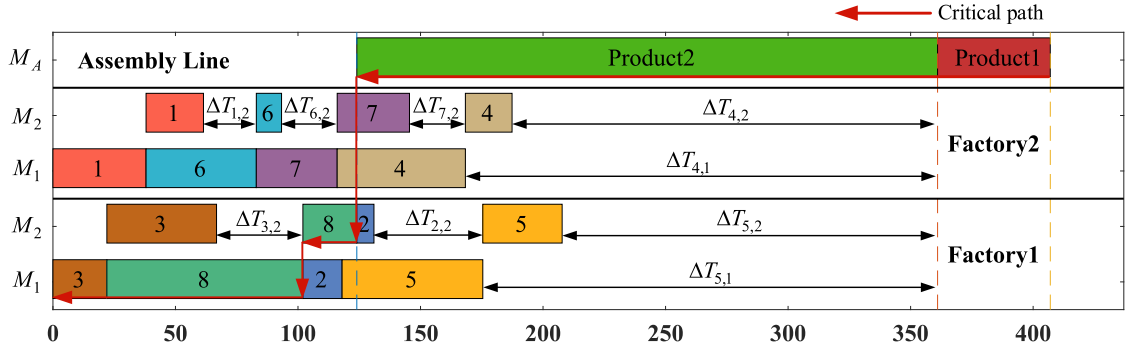
Ding et al. (2016) proposed two assumptions about the relationship between job processing speed and machine energy consumption based on the problem properties of the low-carbon PFSP. That is, when job J_i is processed at a higher speed on a machine M_j, the processing time of the job would be shorter and the total energy consumption would be increased. In other words, if $\forall v_{k1} > v_{k2}$ ($v_{k1}, v_{k2} \in V$), we have $\hat{p}^{k1}_{ij} < \hat{p}^{k2}_{ij}$, then $\hat{p}^{k1}_{ij} \cdot E_{jk1} > \hat{p}^{k2}_{ij} \cdot E_{jk2}$. Obviously, there is an inevitable contradiction relation between the two objectives C_{max} and TCE in the EE_DAPFSP, and there is no optimal scheduling solution can be found in the absolute sense. For any two feasible scheduling solutions $(\pi', \boldsymbol{V}')$ and $(\pi'', \boldsymbol{V}'')$, when $C_{max}(\pi', \boldsymbol{V}') \geqslant C_{max}(\pi'', \boldsymbol{V}'')$, $\mathrm{TCE}(\pi', \boldsymbol{V}') \geqslant \mathrm{TCE}(\pi'', \boldsymbol{V}'')$ and $(C_{max}(\pi', \boldsymbol{V}'), \mathrm{TCE}(\pi', \boldsymbol{V}')) \neq (C_{max}(\pi'', \boldsymbol{V}''), \mathrm{TCE}(\pi'', \boldsymbol{V}''))$, it can be concluded that the solution $(\pi'', \boldsymbol{V}'')$ dominates the solution $(\pi', \boldsymbol{V}')$ (denoted as $(\pi', \boldsymbol{V}') \prec (\pi'', \boldsymbol{V}'')$). According to the above assumptions and the definition of multi-objective domination, two properties of the EE_DAPFSP can be put forward as follows.

Property 1. *Let us assume that the machine speed is fixed in the processing stage. That is, the processing speed matrix remains constant. Under this assumption, for any two feasible scheduling solutions* $(\pi', \boldsymbol{V}')$ *and* $(\pi'', \boldsymbol{V}'')$, *if* $C_{max}(\pi', \boldsymbol{V}') > C_{max}(\pi'', \boldsymbol{V}'')$ *and* $\boldsymbol{V}' = \boldsymbol{V}''$, *then it has* $(\pi', \boldsymbol{V}') \prec (\pi'', \boldsymbol{V}'')$.

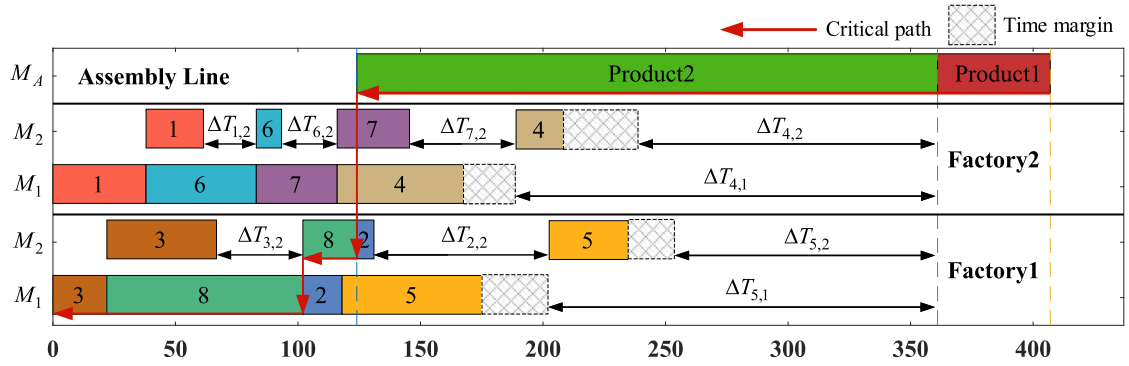
Proof: Assume that the two feasible solutions of the EE_DAPFSP are $(\pi', \boldsymbol{V}')$ and $(\pi'', \boldsymbol{V}'')$, respectively. According to Eq. (9) in Subsection 2.1, the total carbon emission of these two solutions $(\pi', \boldsymbol{V}')$ and $(\pi'', \boldsymbol{V}'')$ can be represented as $\mathrm{TCE}(\pi', \boldsymbol{V}') = \varepsilon(\mathrm{E}(\pi', \boldsymbol{V}') + \mathrm{SE}(\pi', \boldsymbol{V}'))$ and $\mathrm{TCE}(\pi'', \boldsymbol{V}'') = \varepsilon(\mathrm{E}(\pi'', \boldsymbol{V}'') + \mathrm{SE}(\pi'', \boldsymbol{V}''))$, respectively. $\mathrm{E}(\cdot)$ and $\mathrm{SE}(\cdot)$ indicate the total



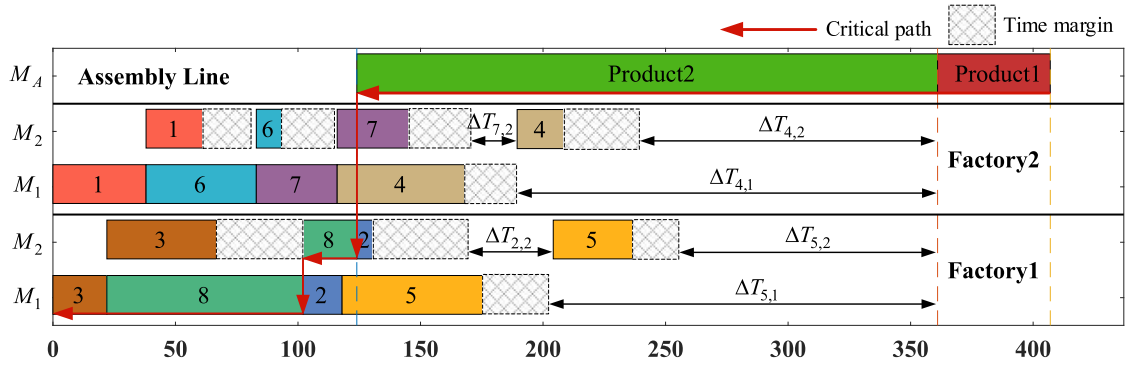
(*a*) Identify critical path for a feasible solution.



(*b*) Calculate time margin between adjacent jobs on the non-critical path.



(*c*) Perform speed-down operations and update time margin from back to front.



(*d*) Perform speed-down operations on all jobs in non-critical path.

Fig. 4. An illustrative example of the energy-saving speed adjustment strategy for EE_DAPFSP.

energy consumption of machines in running state and the energy consumption of machines in standby state. Since $V^{'} = V''$, the two solutions $(\pi^{'}, V^{'})$ and (π'', V'') have exactly the same processing processes of jobs in the processing stage. That is, the processing time and energy consumption of the processing operations are also the same, and obviously $\mathrm{E}(\pi^{'}, V^{'}) = \mathrm{E}(\pi'', V'')$ holds. In addition, each product λ_h ($h = 1,...,S$) in the product order $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, ..., \lambda_S]$ corresponding to π in the solution (π, V) must wait until it's all parts are completed before starting the product assembly process.

According to Eq. (5) in Subsection 2.1, denote $\underset{\pi_i^f \in \lambda_h}{maxC}_{\pi_i^f, m}$ be the start assembly time of product λ_h ($i = 1, ..., n_f$, $f = 1, ..., F$ and $h = 1, ..., S$), $\widehat{\boldsymbol{\pi}}_{\boldsymbol{h}} = [\widehat{\pi}_{h,1}, \widehat{\pi}_{h,2}, ..., \widehat{\pi}_{h,\omega_h}]$ be the job order belonging to product λ_h. Let $\widehat{Pt}^f_{\pi_{h,l},j}$ and $\widehat{St}^f_{\pi_{h,l},j}$ respectively represent the total duration of the processing state and the total duration of the standby state of the job $\widehat{\pi}_{h,l}$ ($l = 1, ..., \omega_h$) contained in product λ_h in the product order $\boldsymbol{\lambda}$ on the machine M_j in factory f, and then the start assembly time of product λ_h can be represented as $\underset{\pi_i^f \in \lambda_h}{maxC}_{\pi_i^f, m} = \max_{f,l}\{\widehat{Pt}^f_{\pi_{h,l},m} + \widehat{St}^f_{\pi_{h,l},m}\}$. The makespan $C_{max}(\pi, V)$ of the feasible solution (π, V) is determined by the start assembly time of each product. Let $\boldsymbol{\lambda}^{'}$ and $\boldsymbol{\lambda}''$ be the product orders corresponding to $\pi^{'}$ and π'' in the solutions $(\pi^{'}, V^{'})$ and (π'', V''). Since $\pi^{'}$ and π'' in $(\pi^{'}, V^{'})$ and (π'', V'') correspond to the same product, and the job order $\widehat{\boldsymbol{\pi}}^{'}_{\boldsymbol{h}}$ in λ_h is the same as that in the job order $\widehat{\boldsymbol{\pi}}''_{\boldsymbol{h}}$ and $\mathrm{E}(\pi^{'}, V^{'}) = \mathrm{E}(\pi'', V'')$. It is clear that $\widehat{Pt}^f_{\pi^{'}_{h,l},j} = \widehat{Pt}^f_{\pi''_{h,l},j}, \forall j \in M$ and $C_{max}(\pi^{'}, V^{'}) > C_{max}(\pi'', V'')$. It can be deduced that if $\exists \lambda_{h^*} \in \boldsymbol{\lambda}$, it has $\widehat{St}^f_{\pi^{'}_{h^*,l'},j} > \widehat{St}^f_{\pi''_{h^*,l'},j}$, $\exists j \in M$. Thus, for the standby state, the total energy consumption can be given as follows:

$$SE(\pi^{'}, V^{'}) = \sum_{f}\sum_{j\in M} St^f_{\pi^{'}_{n_f},j} \times SE_j > \sum_{f}\sum_{j\in M} St^f_{\pi''_{n_f},j} \times SE_j = SE(\pi'', V'') \tag{12}$$

Since there is $\mathrm{TCE}(\pi^{'}, V^{'}) = \varepsilon(\mathrm{E}(\pi^{'}, V^{'}) + \mathrm{SE}(\pi^{'}, V^{'}))$, $\mathrm{TCE}(\pi'', V'') = \varepsilon(\mathrm{E}(\pi'', V'') + \mathrm{SE}(\pi'', V''))$ and $\mathrm{E}(\pi^{'}, V^{'}) = \mathrm{E}(\pi'', V'')$ is met, it has $\mathrm{TCE}(\pi^{'}, V^{'}) > \mathrm{TCE}(\pi'', V'')$ and $C_{max}(\pi^{'}, V^{'}) > C_{max}(\pi'', V'')$. Thus, we can get $(\pi^{'}, V^{'}) \prec (\pi'', V'')$. Property 1 is proved.

Property 2. *For any two feasible solutions $(\pi^{'}, V^{'})$ and (π'', V''), if they have the same maximum completion time, then the solution with the slower processing speed dominates the other solution. That is, if $C_{max}(\pi^{'}, V^{'}) = C_{max}(\pi'', V'')$ when $V^{'}_{i,j} \geqslant V''_{i,j}$ ($\forall i = 1, ..., n; j = 1, ..., m$) and $V^{'} \neq V''$, then it has $(\pi^{'}, V^{'}) \prec (\pi'', V'')$.*

Proof: Since $V^{'}_{i,j} \geqslant V''_{i,j}$ ($\forall i = 1, ..., n; j = 1, ..., m$), $C_{max}(\pi^{'}, V^{'}) = C_{max}(\pi'', V'')$ and $V^{'} \neq V''$, it is clear that the production process of the solution $(\pi^{'}, V^{'})$ is faster than that of the solution (π'', V''). For the job orders $\pi^{'}$ and π'' in any two solutions $(\pi^{'}, V^{'})$ and (π'', V''), each product in the product orders $\boldsymbol{\lambda}^{'}$ and $\boldsymbol{\lambda}''$ corresponding to the job orders $\pi^{'}$ and π'' must wait for the completion of all the parts belonging to the product before it can be assembled. In addition, the job orders $\widehat{\boldsymbol{\pi}}^{'}_{\boldsymbol{h}}$ and $\widehat{\boldsymbol{\pi}}''_{\boldsymbol{h}}$ corresponding to the same product λ_h in the job orders $\pi^{'}$ and π'' are the same. Since $\widehat{Pt}^f_{\pi^{'}_{h,l},j} \leqslant \widehat{Pt}^f_{\pi''_{h,l},j}, \forall j \in M$ and $C_{max}(\pi^{'}, V^{'}) = C_{max}(\pi'', V'')$, then it has $\widehat{St}^f_{\pi^{'}_{h,l},j} \geqslant \widehat{St}^f_{\pi''_{h,l},j}$, $\forall j \in M$. Since $V^{'} \neq V''$, then $\exists j^{'} \in M$ makes $\widehat{Pt}^f_{\pi^{'}_{h,l},j^{'}} < \widehat{Pt}^f_{\pi''_{h,l},j^{'}}$, that is, it has $\widehat{St}^f_{\pi^{'}_{h,l},j^{'}} > \widehat{St}^f_{\pi''_{h,l},j^{'}}$. Therefore, the total energy consumption in Eq. (12) for the standby state is satisfied. According to $\mathrm{TCE}(\pi^{'}, V^{'}) = \varepsilon(\mathrm{E}(\pi^{'}, V^{'}) + \mathrm{SE}(\pi^{'}, V^{'}))$ and $\mathrm{TCE}(\pi'', V'') = \varepsilon(\mathrm{E}(\pi'', V'') + \mathrm{SE}(\pi'', V''))$ in Property 1, the total energy consumption in the processing state satisfies $\mathrm{E}(\pi^{'}, V^{'}) > \mathrm{E}(\pi'', V'')$, so it has $\mathrm{TCE}(\pi^{'}, V^{'}) > \mathrm{TCE}(\pi'', V'')$. Considering that $C_{max}(\pi^{'}, V^{'}) = C_{max}(\pi'', V'')$, the dominance relationship $(\pi^{'}, V^{'}) \prec (\pi'', V'')$ between $(\pi^{'}, V^{'})$ and (π'', V'') is met. Property 2 is proved.

According to above properties, a reasonable compromise between the two objectives of C_{max} and TCE can be achieved, that is, the total carbon emissions can be reduced by appropriately adjusting the speed of the machines in each factory, while the maximum completion time remains unchanged.

It should be clear that for all types of flow shop scheduling problems, the critical path directly determines the maximum completion time of any feasible solution (Wang & Wang, 2016). In this section, according to Property 2, we present an energy-saving speed adjustment strategy that can adjust the processing speed of some jobs on non-critical paths while keeping the processing speed of jobs on the critical path unchanged. The proposed energy-saving speed adjustment strategy can ensure that the maximum completion time of the scheduling solution remains unchanged, while effectively reducing the total carbon emissions, and improving the algorithm's ability to obtain high-quality solutions with low-energy consumption. The energy-saving speed adjustment strategy is provided as follows.

Step 1: Identify a critical path of the feasible solution (π, V). The critical path directly determines the value of the production efficiency criterion (*i.e.*, the maximum completion time C_{max}). If more than one critical path exists for the same C_{max}, then one of the critical paths is randomly selected.

Step 2: Determine whether the job on the non-critical path meets the speed-down operation conditions: there is a certain amount of time margin between the completion time of the current job and the start time of the next job on the machine, and the processing speed of the job is not in the lowest gear. If the speed-down operation conditions are not met, then no speed reduction is required, otherwise continue to the next step.

Step 3: Reduce the processing speed of the job by one level. Since all processing speeds considered are a series of discrete values, the speed-down operation needs to be performed to ensure that the increment of the job processing time is within the time margin and does not affect the critical path identified in Step 1. If the critical path is affected, the speed-down operation is not performed, otherwise skip to Step 2 and continue to perform speed reduction on the remaining jobs until all jobs on the non-critical path have been executed the speed-down operation.

For ease of understanding, Fig. 4 provides a diagram of the energy-saving speed adjustment strategy for EE_DAPFSP when $n = 8, m = 2, F = 2, S = 2$. Firstly, a critical path for the feasible scheduling solution is determined, as shown in Fig. 4(a). It can be seen that the critical factory is Factory 1, and the corresponding critical operations are $O_{3,1}$, $O_{8,1}, O_{8,2}$. Then, the time margins for a series of non-critical operations $O_{3,2}, O_{2,2}, O_{5,2}, O_{5,1}$ and $O_{1,2}, O_{6,2}, O_{7,2}, O_{4,1}, O_{4,2}$ in factory 1 and factory

2 are calculated as shown in Fig. 4(*b*). Finally, the speed-down operations are performed and the time margins are updated for the jobs on the non-critical path from back to front, as illustrated in Fig. 4(*c*). Fig. 4(*d*) gives the Gantt chart corresponding to the obtained solution with low-energy consumption by using the energy-saving speed adjustment strategy. Then, the Pareto archive is updated after the speed adjustment.

Algorithm 2: Multi-objective initialization method

Input: Non-dominated solution set NS.

All products are sorted according to their corresponding assembly time in ascending order, and the product order is $\boldsymbol{\lambda}'=[\lambda'_1,\lambda'_2,...,\lambda'_n]$.

Initialize processing speed matrix $\mathbf{V}:=\mathbf{V}'$. Initialize non-dominated solution set $NS:=\varnothing$.

for $h=1$ *to* S **do**

  The jobs of product λ'_h in product order $\boldsymbol{\lambda}'$ are sorted in ascending order by the total completion time. Then the job order of product λ'_h after sorting is $\hat{\boldsymbol{\pi}}_{\boldsymbol{h}}$.

end for

The complete job order $\boldsymbol{\pi}'$ is constructed by combining each sub-sequence $\hat{\boldsymbol{\pi}}_{\boldsymbol{h}}$ according to the product order $\boldsymbol{\lambda}'=[\lambda'_1,\lambda'_2,...,\lambda'_n]$.

Evaluate $(\boldsymbol{\pi}',\mathbf{V}')$ and update $NS\leftarrow(\boldsymbol{\pi}',\mathbf{V}')$.

for $h=1$ *to* S **do**

  Select the job order $\hat{\boldsymbol{\pi}}_{\boldsymbol{h}}$ of product λ'_h, where ω_h is the number of jobs in $\hat{\boldsymbol{\pi}}_{\boldsymbol{h}}$. $NS_l:=\varnothing$.

  for $l=1$ *to* ω_h **do**

    Remove the lth job from $\hat{\boldsymbol{\pi}}_{\boldsymbol{h}}$ and reinsert it into the ω_h+1 positions of $\hat{\boldsymbol{\pi}}_{\boldsymbol{h}}$ in turn.

    Evaluate the solution to obtain $C_{\max}$ and TCE. The non-dominated solution set obtained after insertion operation is NS_l, and update NS: $NS\leftarrow NS \quad NS_l$.

  end for

end for

while $|NS|>10\%\times ps$ **do**

  Calculate the crowded distance of solutions in NS according to Algorithm 1.

  Remove solutions with the smallest crowded distance from NS until the condition is met.

end while

Output: Non-dominated solution set NS.

3. MCEDA for EE_DAPFSP

In this section, a multidimensional distribution estimation algorithm (MCEDA) is presented to solve the EE_DAPFSP. Firstly, the solution representation and population initialization are provided. Secondly, the matrix-cube-based multidimensional probabilistic model and its updating mechanism and sampling strategy are proposed, respectively. Then, the critical path based local search and the speed adjustment strategies are designed. Finally, the overall process of MCEDA is described and the computational complexity of MCEDA is briefly analyzed.

3.1. Solution representation and population initialization

The solution representation is of great importance to the meta-heuristics. The PFSP usually uses a coding sequence of jobs $\boldsymbol{\pi}=[\pi(1),\pi(2),...,\pi(n)]$ that can directly determine the processing priority of n jobs to represent a feasible solution of the problem (Ding et al., 2016; Lu et al., 2017). For the job order $\boldsymbol{\pi}$ of the EE_DAPFSP, in order to obtain a feasible scheduling scheme, the factory allocation rule NR_2 proposed by Hatami et al. (2013) is employed to decode the job order $\boldsymbol{\pi}$ and assign each job to each factory in turn. Once the job order $\boldsymbol{\pi}$ is determined and the corresponding processing speeds for all operations are assigned, and then a feasible solution is obtained. Therefore, the feasible solution representation of the EE_DAPFSP studied in this paper needs to consider not only the processing sequence $\boldsymbol{\pi}$ for jobs but also the processing speed

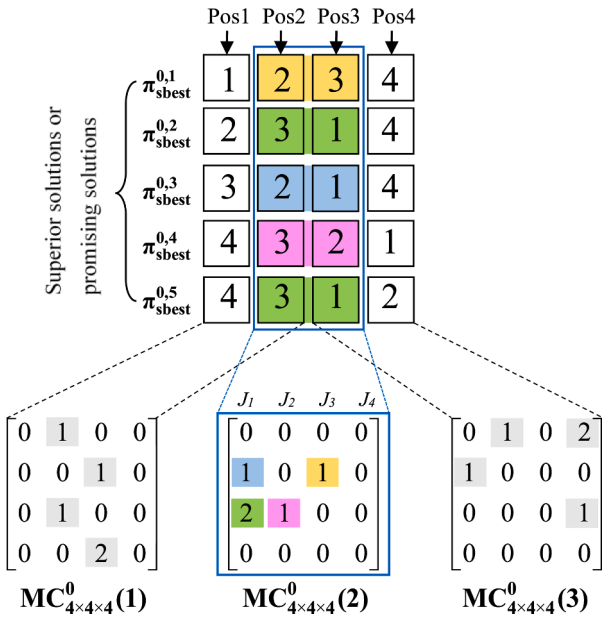


Fig. 5. An illustrative example of the accumulation process of job blocks by $MC^0_{4\times4\times4}$.

matrix V for machines, so a feasible solution can be denoted as (π, V). Thus, the production efficiency and the energy consumption criteria are denoted as $C_{max}(\pi, V)$ and $TCE(\pi, V)$, respectively.

It needs to be noted that the size of the set $\mathbf{\Pi}$ of feasible coding sequences is $n!$, the size of the set of feasible speed matrices $\mathbf{\Sigma}$ is d^{nm}, and the solution spaces of $\mathbf{\Pi}$ and $\mathbf{\Sigma}$ are independent of each other. Compared with the solution space size $n!$ of the traditional PFSP, the feasible solution space size of the EE_DAPFSP is as large as $S! \times \prod_{h=1}^{S} |N_h|! \times d^{nm}$. Obviously, the significant expansion of the feasible search space for the considered problem requires much more computational efforts to find the global optimal solutions or at least the near-optimal solutions, and it is need to develop some high-performing algorithms for solving the EE_DAPFSP. To be specific, in order to decode a feasible solution and obtain a scheduling scheme, firstly, each job in the job order π is sequentially assigned to machines of different factories by means of the NR2 rule. That is, the completion time of each job in each factory is determined respectively according to the corresponding processing speed in V. Secondly, each job is allocated to the factory that can complete all processing processes of the job at the earliest time, and then the sub-sequence of jobs $[\pi_1, \pi_2, ..., \pi_F]$ in all factories is obtained. Then, the assembly order of the products is determined according to the processing completion time of the corresponding jobs for each product. Finally, the maximum completion time $C_{max}(\pi, V)$ and the total carbon emission $TCE(\pi, V)$ of the feasible solution (π, V) can be calculated according to Eq. (8) and Eq. (9) in Section 2.

In order to take into account both the quality and the diversity of the feasible solutions in the initial population, we employ a hybrid initialization strategy that combines the effective constructive heuristic method and the randomization method to generate the initial population. To be specific, 10% of the feasible solutions in the initial population are produced by using the multi-objective initialization method given in Algorithm 2, and the remaining 90% of the other solutions are generated randomly. For the multi-objective optimization problems, the non-dominated set NS is usually used to record and reserve all of the obtained non-dominated solutions. The expression $NS \leftarrow NS \cup NS_l$ in Algorithm 2 represents the set of non-dominated solutions picked from the union of the two sets NS and NS_l. The notation $|NS|$ refers to the number of elements in the non-dominated set NS. Note that the initialization of the processing speed matrix V for each non-dominated solution (π, V) also has a certain impact on the performance of the algorithm. If a higher initial processing speed is set, it would be more favorable to optimize the maximum completion time $C_{max}(\pi, V)$; while if a lower initial processing speed is set, it would be more inclined to optimize the total energy consumption $TCE(\pi, V)$. Therefore, in order to achieve a reasonable trade-off between these two criteria, different speed levels should be adopted to initialize the processing speed matrix V in the initialization process of the population, and then all non-dominated solutions obtained at different initial processing speeds are merged. Then, the top 10% of high-quality solutions are selected as a part of the initial population via Algorithm 1, while the remaining 90% of the solutions in the initial population are generated by using randomization method. Meanwhile, to ensure the fairness of the computational comparisons, the proposed population initialization method is used for both the presented MCEDA and the compared algorithms in the subsequent experimental sections.

3.2. Multi-dimensional probabilistic model

Since most of EDAs were proposed based on the two-dimensional probabilistic models, these two-dimensional probabilistic models cannot learn the promising patterns adequately. In this subsection, a multi-dimensional probabilistic model is designed to reasonably learn and accumulate the structural characteristics and promising patterns of the superior solutions, *i.e.*, the order relation information of jobs and the position information of job blocks. which can effectively guide the search direction toward the potential regions in the solution space. Then, the framework of the multi-dimensional probabilistic model is provided by introducing the block structure and the matrix cube, the updating mechanism and the sampling strategy, respectively.

3.2.1. Block structure and matrix cube

For the feasible scheduling solution (π, V) of the EE_DAPFSP, the job block is first defined as the two consecutive adjacent jobs in the job order π. Obviously, π can be composed of all the job blocks that appear at different positions in the job order. For n different jobs, there is a total of $(n-1)^2$ job blocks. The same job blocks appearing at different positions in the job order π of each feasible solution (π, V) are defined as the similar blocks. For example, for two job orders $\pi' = [3,2,1,4]$ and $\pi'' = [4,3,2,1]$, there are a total of four job blocks, *i.e.*, $[3,2]$, $[2,1]$, $[1,4]$ and $[4,3]$. Since the two job blocks $[3,2]$ and $[2,1]$ appear both in π' and π'', then the job blocks $[3,2]$ and $[2,1]$ are similar blocks. In order to investigate the distribution characteristics of the total job blocks in the high-quality solutions of the considered problem, in this subsection, a data structure of three-dimensional matrix cube is designed to reasonably record and reserve the total order relation information of the jobs and the distribution information of the job blocks. Moreover, the matrix cube can also appropriately learn and accumulate the valuable structural characteristics of superior solutions in a statistical way, and then can be used to construct a more effective probabilistic model. Without loss of generality, let $Pop(G)$ be the population at Gth generation, where $G = 0, 1, ...$

, $MaxG$ and ps is the size of $\boldsymbol{Pop}(\boldsymbol{G})$. Let $\boldsymbol{SPop}(\boldsymbol{G}) = \{\boldsymbol{\pi}_{\boldsymbol{sbest}}^{\boldsymbol{G,1}}, \boldsymbol{\pi}_{\boldsymbol{sbest}}^{\boldsymbol{G,2}}, ..., \boldsymbol{\pi}_{\boldsymbol{sbest}}^{\boldsymbol{G,sps}}\}$ be the promising solutions or the superior subpopulation extracted from $\boldsymbol{Pop}(\boldsymbol{G})$, where sps is the size of $\boldsymbol{SPop}(\boldsymbol{G})$. It is clear that $\boldsymbol{\pi}_{\boldsymbol{sbest}}^{\boldsymbol{G,k}}$ is the kth individual in $\boldsymbol{SPop}(\boldsymbol{G})$, which can be denoted as $\boldsymbol{\pi}_{\boldsymbol{sbest}}^{\boldsymbol{G,k}} = [\pi_{sbest,1}^{G,k}, \pi_{sbest,2}^{G,k}, ..., \pi_{sbest,n}^{G,k}]$ ($k = 1, ..., sps$). Let $\boldsymbol{MC}_{\boldsymbol{n\times n\times n}}^{\boldsymbol{G}}$ denote the matrix cube at Gth generation, where $MC_{n\times n\times n}^{G}(x,y,z)$ ($x = 1, ..., n-1; y, z = 1, ..., n$) with the subscript (x,y,z) represents the element in $\boldsymbol{MC}_{\boldsymbol{n\times n\times n}}^{\boldsymbol{G}}$. $\boldsymbol{MC}_{\boldsymbol{n\times n\times n}}^{\boldsymbol{G}}$ is used to record and reserve the information of the job relations and the distribution of similar blocks of high-quality subpopulation in Gth generation. $\boldsymbol{MC}_{\boldsymbol{n\times n\times n}}^{\boldsymbol{G}}$ is described as follows.

$$I_{MC_{n\times n\times n}^{G,k}(x,y,z)} = \begin{cases} 1, & y = \pi_{sbest,x}^{G,k} \text{ and } z = \pi_{sbest,x+1}^{G,k} \\ 0, & else \end{cases}, \tag{13}$$

$$x = 1, 2, ..., n-1; y, z = 1, 2, ..., n;\ k = 1, 2, \ \ ..., sps.$$

$$MC_{n\times n\times n}^{G}(x,y,z) = \sum_{k=1}^{sps} I_{MC_{n\times n\times n}^{G,k}(x,y,z)}, x = 1, 2, ..., n-1; y, z = 1, 2, ..., n. \tag{14}$$

$$\boldsymbol{MC}_{\boldsymbol{n\times n\times n}}^{\boldsymbol{G}}(\boldsymbol{x},\boldsymbol{y}) = [MC_{n\times n\times n}^{G}(x,y,1), MC_{n\times n\times n}^{G}(x,y,2), ..., MC_{n\times n\times n}^{G}(x,y,n)]_{n\times n},$$

$$x = 1, 2, ..., n-1; y = 1, 2, ..., n. \tag{15}$$

$$\boldsymbol{MC}_{\boldsymbol{n\times n\times n}}^{\boldsymbol{G}}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{MC}_{\boldsymbol{n\times n\times n}}^{\boldsymbol{G}}(\boldsymbol{x},\boldsymbol{1}) \\ \vdots \\ \boldsymbol{MC}_{\boldsymbol{n\times n\times n}}^{\boldsymbol{G}}(\boldsymbol{x},\boldsymbol{n}) \end{bmatrix} = \begin{bmatrix} MC_{n\times n\times n}^{G}(x,1,1) & \cdots & MC_{n\times n\times n}^{G}(x,1,n) \\ \vdots & \ddots & \vdots \\ MC_{n\times n\times n}^{G}(x,n,1) & \cdots & MC_{n\times n\times n}^{G}(x,n,n) \end{bmatrix}_{n\times n}. \tag{16}$$

where $I_{MC_{n\times n\times n}^{G,k}(x,y,z)}$ in Eq. (13) is an indicator function, which is used to record the information of job blocks of the kth solution $\boldsymbol{\pi}_{\boldsymbol{sbest}}^{\boldsymbol{G,k}}$ in $\boldsymbol{SPop}(\boldsymbol{G})$, that is, the number of occurrences of job blocks $[\pi_{sbest,x}^{G,k}, \pi_{sbest,x+1}^{G,k}]$ at the xth position in the $\boldsymbol{\pi}_{\boldsymbol{sbest}}^{\boldsymbol{G,k}}$ (*i.e.*, $y = \pi_{sbest,x}^{G,k}, z = \pi_{sbest,x+1}^{G,k}$). The element $MC_{n\times n\times n}^{G}(x,y,z)$ in Eq. (14) is used to accumulate the number of job blocks, *i.e.*, to count the distribution of job blocks among all individuals in $\boldsymbol{SPop}(\boldsymbol{G})$. Eqs. (15) and (16) give the specific hierarchical structure of the matrix cube, where the two-dimensional submatrix $\boldsymbol{MC}_{\boldsymbol{n\times n\times n}}^{\boldsymbol{G}}(\boldsymbol{x})$ is used to store the total frequency information of the job block $[\pi_{sbest,x}^{G,k}, \pi_{sbest,x+1}^{G,k}]$ at the xth position of all individuals in the $\boldsymbol{SPop}(\boldsymbol{G})$. Obviously, by using the matrix cube $\boldsymbol{MC}_{\boldsymbol{n\times n\times n}}^{\boldsymbol{G}}$ composed of a series of the position relation based two-dimensional matrices $\boldsymbol{MC}_{\boldsymbol{n\times n\times n}}^{\boldsymbol{G}}(\boldsymbol{1}), \boldsymbol{MC}_{\boldsymbol{n\times n\times n}}^{\boldsymbol{G}}(\boldsymbol{2}), ..., \boldsymbol{MC}_{\boldsymbol{n\times n\times n}}^{\boldsymbol{G}}(\boldsymbol{n})$ in Eq. (16), the proposed multi-dimensional probabilistic model can accurately and effectively learn and preserve all the information of both the ordinal relation of jobs and the distribution of job blocks of high-quality individuals in an intuitive way. Considering five high-quality individuals with $sps = 5$, *i.e.*, $\boldsymbol{\pi}_{\boldsymbol{sbest}}^{\boldsymbol{1,1}} = [1,2,3,4]$, $\boldsymbol{\pi}_{\boldsymbol{sbest}}^{\boldsymbol{1,2}} = [2,3,1,4]$, $\boldsymbol{\pi}_{\boldsymbol{sbest}}^{\boldsymbol{1,3}} = [3,2,1,4]$, $\boldsymbol{\pi}_{\boldsymbol{sbest}}^{\boldsymbol{1,4}} = [4,3,2,1]$ and $\boldsymbol{\pi}_{\boldsymbol{sbest}}^{\boldsymbol{1,5}} = [4,3,1,2]$, an illustration of the accumulation process of job blocks from these five high-quality individuals is given below, as shown in Fig. 5. Firstly, for the first position ($x = 1$) of all individuals from $\boldsymbol{\pi}_{\boldsymbol{sbest}}^{\boldsymbol{1,1}}$ to $\boldsymbol{\pi}_{\boldsymbol{sbest}}^{\boldsymbol{1,5}}$, the existing job blocks $[1,2]$ (*i.e.*, $y = 1, z = 2$), $[2,3]$ (*i.e.*, $y = 2, z = 3$), $[3,2]$ (*i.e.*, $y = 3, z = 2$), and $[4,3]$ (*i.e.*, $y = 4, z = 3$) can be recorded and reserved by $\boldsymbol{MC}_{\boldsymbol{4\times4\times4}}^{\boldsymbol{G}}(\boldsymbol{1})$. Since the job block $[4,3]$ appears twice while the other job blocks appear only once, it can be concluded that $MC_{4\times4\times4}^{G}(1,1,2) = 1$, $MC_{4\times4\times4}^{G}(1,2,3) = 1$, $MC_{4\times4\times4}^{G}(1,3,2) = 1$, and $MC_{4\times4\times4}^{G}(1,4,3) = 2$. Secondly, for the second position ($x = 2$) of all individuals, the existing job blocks $[2,1]$ (*i.e.*, $y = 2, z = 1$), $[2,3]$ (*i.e.*, $y = 2, z = 3$), $[3,1]$ (*i.e.*, $y = 3, z = 1$), and $[3,2]$ (*i.e.*, $y = 3, z = 2$) can be recorded and reserved by $\boldsymbol{MC}_{\boldsymbol{4\times4\times4}}^{\boldsymbol{G}}(\boldsymbol{2})$. Meanwhile, we have $MC_{4\times4\times4}^{G}(2,2,1) = 1$, $MC_{4\times4\times4}^{G}(2,2,3) = 1$, $MC_{4\times4\times4}^{G}(2,3,1) = 2$, and $MC_{4\times4\times4}^{G}(2,3,2) = 1$, respectively. Finally, for the third position ($x = 3$) of all individuals, the existing job blocks $[1,2]$ (*i.e.*, $y = 1, z = 2$), $[1,4]$ (*i.e.*, $y = 1, z = 4$), $[2,1]$ (*i.e.*, $y = 2, z = 1$), and $[3,4]$ (*i.e.*, $y = 3, z = 4$) can be recorded and reserved by $\boldsymbol{MC}_{\boldsymbol{4\times4\times4}}^{\boldsymbol{G}}(\boldsymbol{3})$, that is, $MC_{4\times4\times4}^{G}(3,1,2) = 1$, $MC_{4\times4\times4}^{G}(3,1,4) = 2$, $MC_{4\times4\times4}^{G}(3,2,1) = 1$, and $MC_{4\times4\times4}^{G}(3,3,4) = 1$. It should be noted that since the useful information of the job block in the last position ($x = 4$) is already contained in the previous position ($x = 3$), all elements in $\boldsymbol{MC}_{\boldsymbol{4\times4\times4}}^{\boldsymbol{G}}(\boldsymbol{4})$ are set to zero directly.

It can be seen from Fig. 5 that all of the job blocks or similar blocks of each excellent individual located at different positions can be fully learned and preserved in $\boldsymbol{MC}_{\boldsymbol{4\times4\times4}}^{\boldsymbol{0}}$. For the two-dimensional probabilistic model based EDA commonly used in the literature (Pan & Ruiz, 2012; Jarboui et al., 2009; Wang & Wang, 2016), the structural information of similar blocks $[2,3]$, $[4,3]$, and $[1,4]$ existed in $\boldsymbol{\pi}_{\boldsymbol{sbest}}^{\boldsymbol{1,1}}$ to $\boldsymbol{\pi}_{\boldsymbol{sbest}}^{\boldsymbol{1,5}}$ is only stored in the same subscript $(2,3)$, $(4,3)$, and $(1,4)$ by using the two-dimensional matrices, which may not be able to accurately distinguish the specific location of each job block in the elite solutions and inevitably lead to confusion about the location of the promising blocks. As a result, the two-dimensional probabilistic model based EDA is unable to effectively determine the proper positions to place these promising similar blocks when sampling the two-dimensional probabilistic model to generate new individuals. However, for the designed matrix cube $\boldsymbol{MC}_{\boldsymbol{n\times n\times n}}^{\boldsymbol{G}}$, these valuable similar blocks can be respectively record and retained in different layers of $\boldsymbol{MC}_{\boldsymbol{n\times n\times n}}^{\boldsymbol{G}}$ depending on their specific positions. It is clear that the similar block $[2,3]$ in $\boldsymbol{\pi}_{\boldsymbol{sbest}}^{\boldsymbol{1,1}} = [1,2,3,4]$ is recorded in $\boldsymbol{MC}_{\boldsymbol{4\times4\times4}}^{\boldsymbol{G}}(\boldsymbol{2})$(*i.e.*, the second layer of $\boldsymbol{MC}_{\boldsymbol{n\times n\times n}}^{\boldsymbol{G}}$), while the similar block $[2,3]$ in $\boldsymbol{\pi}_{\boldsymbol{sbest}}^{\boldsymbol{1,2}} = [2,3,1,4]$ is reserved in $\boldsymbol{MC}_{\boldsymbol{4\times4\times4}}^{\boldsymbol{G}}(\boldsymbol{1})$ (*i.e.*, the first layer of $\boldsymbol{MC}_{\boldsymbol{n\times n\times n}}^{\boldsymbol{G}}$), respectively. That is, the promising patterns of all job blocks or similar blocks are properly learned by employing the proposed matrix cube, which can be used to bulid more effective probabilistic model, while also avoiding the destruction or improper fusion of promising patterns.

3.2.2. Updating mechanism

The probabilistic models are crucial to the EDAs, and the effectiveness and reasonableness of the designed probabilistic model directly affects the performance of the algorithm (Zhang et al., 2021). Different from the two-dimensional probabilistic models, in this subsection, a novel matrix-cube-based multidimensional probabilistic is presented to learn and accumulate the valuable information of the relation of jobs and the distribution of similar blocks in high-quality subpopulation. For descriptive convenience, define $\boldsymbol{PM}_{\boldsymbol{n\times n\times n}}^{\boldsymbol{G}}$ as a multidimensional probabilistic model based on $\boldsymbol{MC}_{\boldsymbol{n\times n\times n}}^{\boldsymbol{G}}$, where $PM_{n\times n\times n}^{G}(x,y,z)$ ($x = 1, ..., n-1; y, z = 1, ..., n$) is the element of $\boldsymbol{PM}_{\boldsymbol{n\times n\times n}}^{\boldsymbol{G}}$. Then, the formal definition of the probability distribution of the job blocks at the xth position in the job order $\boldsymbol{\pi}$ of the selected superior solutions is shown in Eq. (17).

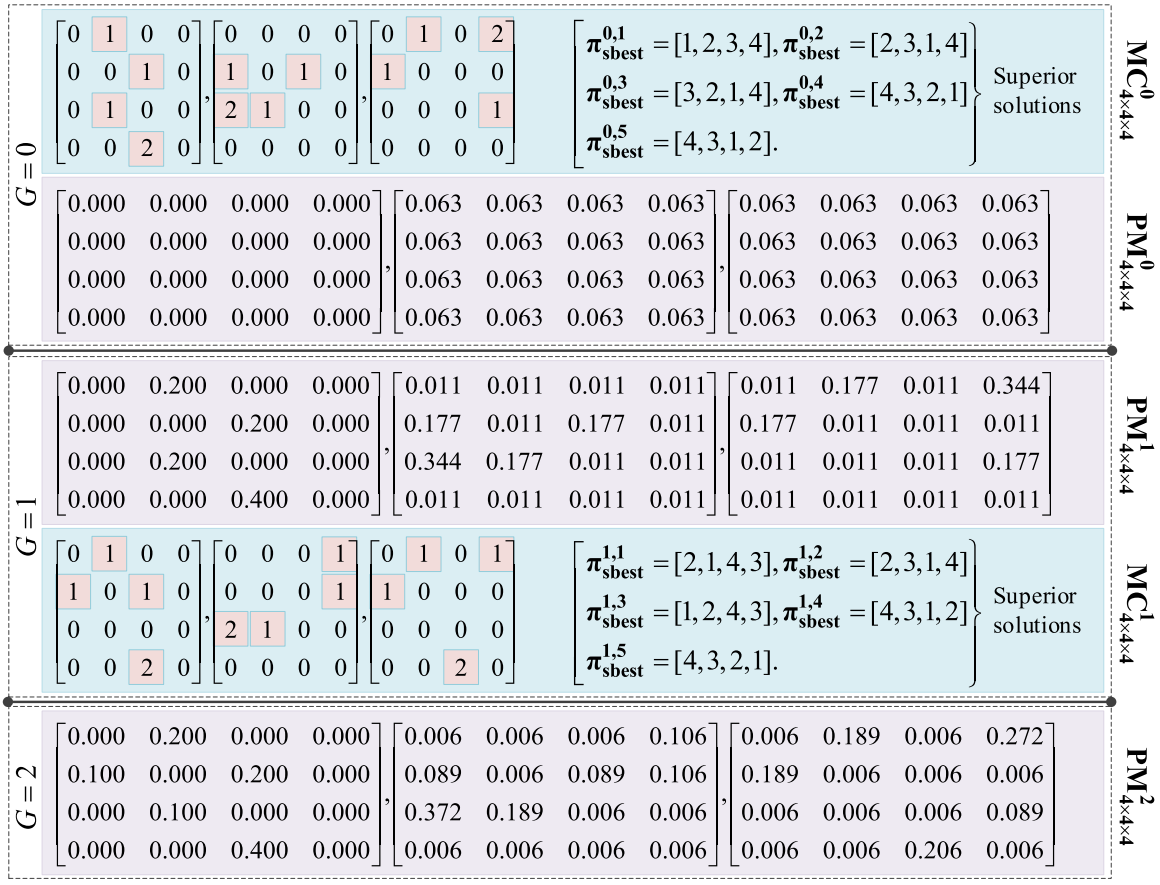


Fig. 6. An illustrative example of the update process of the proposed probabilistic model.

$$\boldsymbol{PM}^{G}_{n\times n\times n}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{PM}^{G}_{n\times n\times n}(\boldsymbol{x},\boldsymbol{1}) \\ \vdots \\ \boldsymbol{PM}^{G}_{n\times n\times n}(\boldsymbol{x},\boldsymbol{n}) \end{bmatrix} = \begin{bmatrix} PM^{G}_{n\times n\times n}(x,1,1) & \cdots & PM^{G}_{n\times n\times n}(x,1,n) \\ \vdots & \ddots & \vdots \\ PM^{G}_{n\times n\times n}(x,n,1) & \cdots & PM^{G}_{n\times n\times n}(x,n,n) \end{bmatrix}_{n\times n}. \tag{17}$$

where the probability value in $\boldsymbol{PM}^{G}_{n\times n\times n}(\boldsymbol{x})$ indicates the probability of occurrence of the job blocks $[\pi^{G,k}_{sbest,x}, \pi^{G,k}_{sbest,x+1}]$ at the xth position in the $\boldsymbol{\pi}^{G,k}_{sbest}$ (*i.e.*, $y = \pi^{G,k}_{sbest,x}, z = \pi^{G,k}_{sbest,x+1}$). In order to update the probabilistic model, let $N^{G}_{MC}(x)$ be the total number of job blocks that have appeared at the xth position in the superior subpopulation $\boldsymbol{SPop}(\boldsymbol{G})$, *i.e.*, $N^{G}_{MC}(x) = \sum_{y=1}^{n}\sum_{z=1}^{n} MC^{G}_{n\times n\times n}(x,y,z)$, $N^{G}_{PM}(x)$ be the sum of all the probabilities of different job blocks appearing at the xth position in $\boldsymbol{SPop}(\boldsymbol{G})$, *i.e.*, $N^{G}_{PM}(x) = \sum_{y=1}^{n}\sum_{z=1}^{n} PM^{G}_{n\times n\times n}(x,y,z)$. The specific update steps of the proposed probabilistic model $\boldsymbol{PM}^{G}_{n\times n\times n}$ are as follows.

Step 1: When $G = 0$, initialize the probabilistic model $\boldsymbol{PM}^{0}_{n\times n\times n}$ according to Eq. (18).

$$PM^{0}_{n\times n\times n}(x,y,z) = \begin{cases} 0, & x = 1; y,z = 1,2,\ldots,n \\ 1/n^2, & x = 2,3,\ldots,n-1; y,z = 1,2,\ldots,n \end{cases}. \tag{18}$$

Step 2: When $G = 1$, calculate the matrix cube $\boldsymbol{MC}^{0}_{n\times n\times n}$ according to Eqs. (13–16) and update the probabilistic model $\boldsymbol{PM}^{1}_{n\times n\times n}$ according to Eq. (19).

$$PM^{1}_{n\times n\times n}(x,y,z) = \begin{cases} MC^{0}_{n\times n\times n}(x,y,z)/N^{0}_{MC}(x), & x = 1; \\ \dfrac{[PM^{0}_{n\times n\times n}(x,y,z) + MC^{0}_{n\times n\times n}(x,y,z)]}{[N^{0}_{PM}(x) + N^{0}_{MC}(x)]}, & x \geqslant 2; \end{cases} \quad \forall y,z = 1,2,\ldots,n. \tag{19}$$

Step 3: When $G > 1$, $\boldsymbol{MC}^{G-1}_{n\times n\times n}$ is calculated and the probabilistic model $\boldsymbol{PM}^{G}_{n\times n\times n}$ is updated iteratively according to Eq. (20), where r is the adjustable learning rate.

$$PM^{G}_{n\times n\times n}(x,y,z) = (1-r)\times PM^{G-1}_{n\times n\times n}(x,y,z) + r\times MC^{G-1}_{n\times n\times xn}(x,y,z)/N^{G-1}_{MC}(x), \quad x = 1,2,\ldots,n-1; y,z = 1,2,\ldots,n. \tag{20}$$

Step 4: Set $G = G + 1$. If $G < MaxG$, then go to Step 3.

Note that all probability values in the in the first layer of $\boldsymbol{PM}^{0}_{n\times n\times n}$ are set to 0 while others are set to $1/n^2$ in Eq. (18), which can learn the initial structural features of superior sub-population and increase the guidance toward potential regions at the initial phase. Moreover, the probabilistic model $\boldsymbol{PM}^{G}_{n\times n\times n}$ in step 3 can keep learning the promising patterns extracted from the superior subpopulation and progressively accumulate the useful information of similar blocks by using an

adjustable learning rate to make a trade-off between historical and current information. At the end of the update process, the normalization is necessary for the probabilistic model $\boldsymbol{PM}^{G}_{n\times n\times n}$. In order to illustrate the proposed probabilistic model $\boldsymbol{PM}^{G}_{n\times n\times n}$ clearly, the update process of $\boldsymbol{PM}^{G}_{n\times n\times n}$ is given in Fig. 6, where the learning rate r is set to 0.5.

3.2.3. Sampling strategy

The probabilistic model is able to implicitly estimate the distribution characteristics of the superior solutions in the solution space by converting characteristic information into corresponding probability values. In order to appropriately apply these stored promising patterns from the high-quality solutions by means of $\boldsymbol{MC}^{G}_{n\times n\times n}$, it is necessary to design an effective sampling strategy for the proposed multi-dimensional probabilistic model $\boldsymbol{PM}^{G}_{n\times n\times n}$. Let $\boldsymbol{\pi}^{G,k}=[\pi_1^{G,k},\pi_2^{G,k},\ldots,\pi_n^{G,k}]$ denote the kth individual in $\boldsymbol{Pop}(\boldsymbol{G})$, and $\boldsymbol{R}^{G}_{PM}=[R^{G}_{PM}(1),R^{G}_{PM}(2),\ldots,R^{G}_{PM}(n)]$ denote the temporary row vector. Define $SelectJob(\boldsymbol{\pi}^{G,k},i)$ $(i>1)$ as the job selection function, which is used to determine one candidate job J_s at the ith position of $\boldsymbol{\pi}^{G,k}$. Since the probability information that the job block $[\pi_{i-1}^{G,k},\pi_i^{G,k}]$ is selected to locate in the $(i-1)$th position in $\boldsymbol{\pi}^{G,k}$ is stored in the $\boldsymbol{PM}^{G}_{n\times n\times n}(\boldsymbol{i-1})$, where $i>1$, the job selection function $SelectJob(\boldsymbol{\pi}^{G,k},i)$ is used to sample only by means of the $(i-1)$th layer of the probabilistic model $\boldsymbol{PM}^{G}_{n\times n\times n}$, and then the procedure of $SelectJob(\boldsymbol{\pi}^{G,k},i)$ is described in Algorithm 3. Note that the job selection function $SelectJob(\boldsymbol{\pi}^{G,k},i)$ depends on the job $\pi_{i-1}^{G,k}$ at the $(i-1)$th position when selecting the job $\pi_i^{G,k}$ at the ith position in $\boldsymbol{\pi}^{G,k}$. Because the job $\pi_0^{G,k}$ does not exist, $SelectJob(\boldsymbol{\pi}^{G,k},i)$ cannot be adopted to determine the first job $\pi_1^{G,k}$ of $\boldsymbol{\pi}^{G,k}$. In order to guide the search direction toward promising regions found by the superior solutions, a specially designed sampling strategy is used to determine the first job of $\boldsymbol{\pi}^{G,k}$, namely first position limited sampling strategy (FPLSS), which is described in Algorithm 4. It is clear that the $\boldsymbol{PM}^{G}_{n\times n\times n}(\boldsymbol{1})$ holds the total probability information of various job blocks that appeared in the first position of all the selected superior solutions in $\boldsymbol{SPop}(\boldsymbol{G})$ before the Gth generation. In Algorithm 4, lines 1–4 are used to calculate the cumulative probability of each row vector in $\boldsymbol{PM}^{G}_{n\times n\times n}(\boldsymbol{1})$, and lines 5–13 are used to generate the first job $\pi_1^{G,k}$ of $\boldsymbol{\pi}^{G,k}$ by means of the roulette wheel selection, which is helpful in controlling the search direction reasonably. The procedure of new population generation is provided in Algorithm 5.

Algorithm 3: $SelectJob(\boldsymbol{PM}^{G}_{n\times n\times n}(\boldsymbol{i-1}),\boldsymbol{\pi}^{G,k},i)$

Input: $\mathbf{PM}^{G}_{n\times n\times n}(\mathbf{i-1})$, $\boldsymbol{\pi}^{G,k}$ and i.

1: Produce a random value p_r where $p_r\in\left[0,\sum_{h=1}^{n}PM^{G}_{n\times n\times n}(i-1,\pi_{i-1}^{G,k},h)\right)$.
2: **if** $p_r\in\left[0,PM^{G}_{n\times n\times n}(i-1,\pi_{i-1}^{G,k},1)\right)$ **then**
3: $J_s\leftarrow 1$.
4: **else**
5: **for** $t=1$ *to* $n-1$ **do** *//Roulette wheel selection.*
6: **if** $p_r\in\left[\sum_{h=1}^{t}PM^{G}_{n\times n\times n}(i-1,\pi_{i-1}^{G,k},h),\sum_{h=1}^{t+1}PM^{G}_{n\times n\times n}(i-1,\pi_{i-1}^{G,k},h)\right)$ **then**
7: $J_s\leftarrow t+1$, *break.*
8: **end if**
9: **end for**
10: **end if**
11: **for** $t=i$ *to* $n-1$ **do** *//Avoid repeated selection of jobs.*
12: **for** $j=1$ *to* n **do**
13: $PM^{G}_{n\times n\times n}(t,j,J_s)=0$.
14: **end for**
15: **end for**

Output: the candidate job J_s.

It should be pointed out that line 6 in Algorithm 5 is used to build the promising job block (*i.e.*, $[\pi_{i-1}^{G,k},\pi_i^{G,k}]$) at positions $i-1$ and i by using the roulette wheel selection rule on the row vector $\boldsymbol{PM}^{G}_{n\times n\times n}(\boldsymbol{i-1},\boldsymbol{\pi}_{\boldsymbol{i-1}}^{G,k})$. Meanwhile, the promising job blocks at different positions can be linked together via order relation information in lines 2–9 to produce a new individual, which is a key step of MCEDA's global exploration. The larger the probability value corresponding to the job block, the more

likely it is to be selected during the sampling process. Since the large values and their subscripts in $\boldsymbol{PM}^{\boldsymbol{G}}_{\boldsymbol{n\times n\times n}}$ are determined by the excellent individuals, new generated individuals can inherit more promising patterns or blocks at suitable positions. Therefore, MCEDA can better guide the search to promising regions in solution space. In Algorithm 3, the computational complexity in lines 2–10 is $O(n)$ and the complexity in lines 11–15 is $O(n^2)$. In Algorithm 4, the computational complexity in line 2, lines 5–13 and lines 14–18 are $O(n^2)$, $O(n)$, and $O(n^2)$, respectively. Thus, the complexity of Algorithm 3 and Algorithm 4 are $O(n^2)$.

In Algorithm 5, the computational complexity in line 4, line 6, and lines 1–11 are $O(n^2)$, $O(n^2)$ and $O(ps\times n^2)$, respectively. Thus, the total complexity of Algorithm 5 is $O(ps\times n^2)$.

Algorithm 4: $FPBSS(\boldsymbol{PM}^{\boldsymbol{G}}_{\boldsymbol{n\times n\times n}}(\mathbf{1}),\boldsymbol{\pi}^{\boldsymbol{G,k}})$

Input: $\mathbf{PM}^{\mathbf{G}}_{\mathbf{n\times n\times n}}(\mathbf{1})$, $\boldsymbol{\pi}^{\mathbf{G,k}}$.
1: **for** $y=1$ *to* n **do**
2: $R^G_{PM}(y)=\sum_{z=1}^{n} PM^G_{n\times n\times n}(1,y,z)$. *//Calculate the cumulative probability.*
3: **end for**
4: Produce a random value p_r where $p_r\in\left[0,\sum_{y=1}^{n} R^G_{PM}(y)\right)$.
5: **if** $p_r\in\left[0,R^G_{PM}(1)\right)$ **then**
6: $J_s\leftarrow 1$.
7: **else**
8: **for** $t=1$ *to* $n-1$ **do** *//Roulette wheel selection.*
9: **if** $p_r\in\left[\sum_{y=1}^{t} R^G_{PM}(y),\sum_{y=1}^{t+1} R^G_{PM}(y)\right)$ **then**
10: $J_s\leftarrow t+1$, *break.*
11: **end if**
12: **end for**
13: **end if**
14: **for** $t=1$ *to* $n-1$ **do** *//Avoid repeated selection of jobs.*
15: **for** $j=1$ to n **do**
16: $PM^G_{n\times n\times n}(t,j,J_s)=0$.
17: **end for**
18: **end for**
Output: the candidate job J_s.

Algorithm 5: New Population Generation

Input: $\mathbf{PM}^{\mathbf{G}}_{\mathbf{n\times n\times n}}$, $\boldsymbol{\pi}^{\mathbf{G,k}}$, $\mathbf{Pop(G+1)}$.
1: **for** $k=1$ *to* ps **do**
2: **for** $i=1$ *to* n **do**
3: **if** $i=1$ **then**
4: $J_s\leftarrow FPBSS(\mathbf{PM}^{\mathbf{G}}_{\mathbf{n\times n\times n}}(\mathbf{1}),\boldsymbol{\pi}^{\mathbf{G,k}})$. *//Algorithm 4*
5: **else**
6: $J_s\leftarrow SelectJob(\mathbf{PM}^{\mathbf{G}}_{\mathbf{n\times n\times n}}(\mathbf{i-1}),\boldsymbol{\pi}^{\mathbf{G,k}},i)$. *//Algorithm 3*
7: **end if**
8: $\pi_i^{G,k}=J_s$.
9: **end for**
10: $\mathbf{Pop(G+1)}\leftarrow\mathbf{Pop(G+1)}$ $\boldsymbol{\pi}^{\mathbf{G,k}}$, $\boldsymbol{\pi}^{\mathbf{G,k}}\leftarrow\varnothing$.

3.3. Critical path-based local search

According to the property analysis of the considered problem in Subsection 2.3, it can be seen that the landscape of the feasible solution space of the EE_DAPFSP is complex and varied, resulting in a large number of high-quality non-dominated solutions are non-uniformly scattered in several local regions near the bottom of the feasible solution space. In order to enhance the depth search capability and achieve a satisfactory trade-off between exploration and exploitation, it is very necessary to conduct a deeper exploitation (local search) on the nearest neighbor regions of these non-dominated solutions found by the exploration (global search) of MCEDA. It is well known that the ability of the local search largely depends on the development of the neighborhood structure and the design of the neighborhood order. For the sequence model of PFSP, there are several commonly used neighborhood search operations, such as *Insert*, *Swap*, *Interchange* and *Inverse*. Notice that, for the DAPFSP, the intra-factory swap or insert operations and the inter-factory swap or insert operations are commonly used neighborhood operations (Wang & Wang, 2016; Zhang et al., 2021). Since the maximum completion time $C_{max}(\pi, \boldsymbol{V})$ of each feasible solution $(\pi, \boldsymbol{V})$ is directly determined by the critical path (Wang & Wang, 2016; Zhang et al., 2021), it is possible to shorten the maximum completion time only by adjusting all jobs on the critical path. In this subsection, four kinds of the neighborhood structures are designed based on the problem-specific critical path, and then a variable neighborhood search method based on these neighborhood structures is performed for the high-quality solutions in the non-dominated set Ω obtained in MCEDA's global search. To be specific, the swap and insert operations within the factory are performed separately for each non-dominated solution, and then the swap and insert operations between factories are performed separately. It should be noted that the job on the critical path of the feasible solution $(\pi, \boldsymbol{V})$ is the critical job, the factory where the critical job is located is the critical factory, denoted as f_c. Let n_{f_c} be the total number of jobs assigned to the critical factory f_c, and n_c be the number of critical jobs in the critical factory. To make it more intuitive, an illustration of critical path-based neighborhood operations is given in Fig. 7. Moreover, the four types of critical path-based neighborhood structures are described in detail as follows.

(1) *Intra-factory Swap*: Randomly select a critical job $\pi_u^{f_c}$ ($u \in \{1,2,\ldots,n_c\}$) and exchange the positions of the critical job $\pi_u^{f_c}$ with each non-critical job $\pi_v^{f_c}$ ($v = n_c+1,\ldots,n_{f_c}$) within the critical factory f_c.

(2) *Intra-factory Insertion*: Randomly select a critical job $\pi_u^{f_c}$ ($u \in \{1,2,\ldots,n_c\}$) and insert the critical job $\pi_u^{f_c}$ before or after the position of

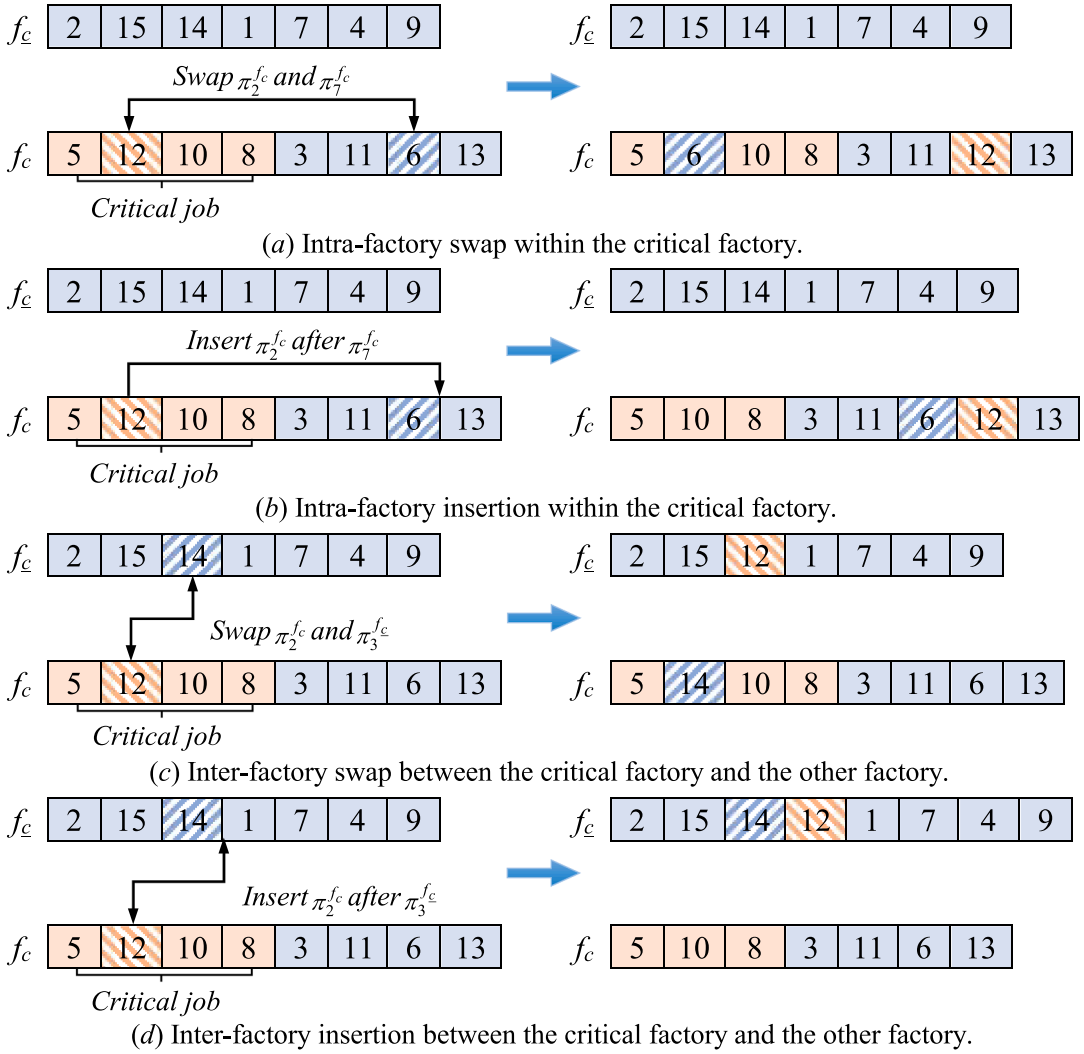


Fig. 7. An illustration of the four critical path-based neighborhood structures.

each non-critical job $\pi_v^{f_c}$ ($v = n_c + 1, \ldots, n_{f_c}$) in the critical factory f_c, respectively.

(3) *Inter-factory Swap*: Randomly select a critical job $\pi_u^{f_c}$ ($u \in \{1, 2, \ldots, n_c\}$) and a non-critical job $\pi_v^{f_{\underline{c}}}$ ($v \in \{1, 2, \ldots, n_{f_{\underline{c}}}\}$) in $F-1$ non-critical factories $f_{\underline{c}}$, and exchange the positions of the critical job $\pi_u^{f_c}$ with each non-critical job $\pi_v^{f_{\underline{c}}}$, respectively.

(4) *Inter-factory Insertion*: Randomly select a critical job $\pi_u^{f_c}$ ($u \in \{1, 2, \ldots, n_c\}$) and a non-critical job $\pi_v^{f_{\underline{c}}}$ ($v \in \{1, 2, \ldots, n_{f_{\underline{c}}}\}$) in $F-1$ non-critical factories $f_{\underline{c}}$, and insert the critical job $\pi_u^{f_c}$ before or after the position of each non-critical job $\pi_v^{f_{\underline{c}}}$, respectively.

3.4. Speed adjustment strategy

In order to achieve a satisfactory tradeoff between the maximum completion time $C_{max}(\pi, \boldsymbol{V})$ and total carbon emission $\mathrm{TCE}(\pi, \boldsymbol{V})$, the proposed algorithm not only needs to conduct the critical path based local search for the corresponding job order π of each non-dominated solution $(\pi, \boldsymbol{V})$ obtained in each generation, but also needs to regulate the corresponding speed assignment matrix $\boldsymbol{V}$ for each non-dominated solution $(\pi, \boldsymbol{V})$ appropriately. Since the processing speed of each machine in all factories is adjustable, it is necessary to further perform a series of speed adjustment strategies for each non-dominated solution $(\pi, \boldsymbol{V})$ obtained from the critical path based local search in Subsection 3.3. According to the problem property analysis in Subsection 2.3, two types of speed adjustment strategies are given as follows.

(1) The energy-saving speed adjustment strategy. As for the total carbon emission (TCE) criterion, the processing speed of each job on the critical path of feasible solution is kept unchanged, and the processing speed of the other jobs on the non-critical path should be suitably shortened to avoid excessive energy consumption and carbon emission as much as possible. This speed adjustment strategy (see Subsection 3.2) is given in Algorithm 6.

(2) The reduced-time speed adjustment strategy. As for the maximum completion time (C_{max}) criterion, the processing state of each job on the non-critical path of each feasible non-dominated solutions is kept unchanged, and the processing speed of each job on the critical path can be increased to further reduce the maximum completion time (C_{max}) as much as possible. To be specific, if there is a slightly larger time margin between the critical operation $O_{i,j}$ and the non-critical operation $O_{i-1,j}$ on any one machine M_j, the processing speed of the critical operation $O_{i,j-1}$ can be increased appropriately, so the corresponding processing time of $O_{i,j-1}$ would be shortened. Then, all related operations after $O_{i,j-1}$ can be shifted forward, and these operations can be completed as early as possible, which may directly advance the assembly completion time of the first product and all other subsequent products. As introduced in Table 1, this speed adjustment strategy can generate possible new non-dominated solutions, thereby enhancing the diversity and number of the obtained non-dominated solutions.

The procedure of the reduced-time speed adjustment strategy is shown in Algorithm 7. For example, three operations $O_{3,1}, O_{8,1}, O_{8,2}$ on the critical path in Fig. 4(a) correspond to three critical operations, and there is a certain time margin between the critical operation $O_{8,2}$ and the non-critical operation $O_{3,2}$. Then, the processing speed of the critical operation $O_{8,1}$ can be increased, so that the processing time of this critical operation is shortened accordingly. All of operations after the critical operation $O_{8,1}$ can be moved from the backward to the forward in turn, which may allow the jobs and the product P_2 to be processed and assembled as early as possible. In order to ensure that the non-dominated solutions obtained by the proposed algorithm have good dispersion and diversity, different speed adjustment strategies are further carried out to selectively optimize different criteria, *i.e.*, the makespan and the total carbon emission. For each non-dominated solution $(\pi, \boldsymbol{V})$ in Pareto archive Ω^*, the maximum completion time (C_{max}) and the total carbon emission (TCE) are first normalized separately for each non-dominated solution, that is, $\overline{C}_{max}(\pi, \boldsymbol{V}) = (C_{max}(\pi, \boldsymbol{V}) - C_{max}^{min})/(C_{max}^{max} - C_{max}^{min})$, $\overline{\mathrm{TCE}}(\pi, \boldsymbol{V}) = (\mathrm{TCE}(\pi, \boldsymbol{V}) - \mathrm{TCE}^{min})/(\mathrm{TCE}^{max} - \mathrm{TCE}^{min})$, where C_{max}^{min}, C_{max}^{max}, TCE^{min} and TCE^{max} respectively represent the smallest C_{max}, the largest C_{max}, the smallest TCE and the largest TCE of all feasible solutions in the obtained non-dominated set. Then, the computational expression $\alpha(\pi, \boldsymbol{V}) = (\overline{\mathrm{TCE}}(\pi, \boldsymbol{V}) + \varepsilon)/(\overline{C}_{max}(\pi, \boldsymbol{V}) + \varepsilon)$ is defined as the preference level of each feasible solution $(\pi, \boldsymbol{V})$, where $\varepsilon = 0.01$ and the range of α is $(0, +\infty)$. Obviously, the larger value of α implies that the total carbon emission of the feasible solution $(\pi, \boldsymbol{V})$ are higher while the maximum completion time of this solution is smaller, so it is necessary to focus on reducing the total carbon emission rather than the maximum completion time. Conversely, the smaller value of α suggests that the total carbon emission of the feasible solution $(\pi, \boldsymbol{V})$ is lower but its maximum completion time is larger, then more attentions should be paid to optimize the production efficiency criterion. According to the different values of preference level α for each feasible solution $(\pi, \boldsymbol{V})$, two kinds of speed adjustment strategies are provided as follows.

Step 1: Identify a critical path for each non-dominated solution in non-dominated solution set Ω. If there are multiple critical paths, one critical path is selected randomly.

Step 2: Perform the presented four types of critical path-based neighborhood search operations for each non-dominated solution in turn. If the obtained new solution dominates old solution, the new non-dominated solution is used to replace the old one and re-determine the critical path. Then, continue to perform the subsequent neighborhood search operations. If they are not dominated by either of them, the obtained new solution is added to the non-dominated solution set Ω, and other remaining neighborhood search operations continue to be performed on the current solution.

Step 3: Calculate $\overline{C}_{max}(\pi, \boldsymbol{V})$ and $\overline{\mathrm{TCE}}(\pi, \boldsymbol{V})$ for each solution $(\pi, \boldsymbol{V})$ in the non-dominated set Ω, and then $\alpha(\pi, \boldsymbol{V}) = (\overline{\mathrm{TCE}}(\pi, \boldsymbol{V}) + \varepsilon)/(\overline{C}_{max}(\pi, \boldsymbol{V}) + \varepsilon)$ is obtained.

Step 4: Sort all non-dominated solutions in ascending value of α, and divide the non-dominated solution set Ω into two parts, namely Ω_c and Ω_e, where $|\Omega| = |\Omega_c| + |\Omega_e|$. The obtained non-dominated solutions in Ω_c with a small value of α need to be optimized for C_{max}, while the non-dominated solutions in Ω_e have a large value of α and these solutions should to be optimized for TCE.

Step 5: Perform the reduced-time speed adjustment strategy for all non-dominated solutions in Ω_c. That is, the processing state of all jobs on the non-critical path is kept unchanged and the processing speed of some jobs on the critical path is increased, so as to reduce the maximum completion time (C_{max}) of these non-dominated solutions.

Step 6: Perform the energy-saving speed adjustment strategy for all non-dominated solutions in Ω_e. That is, the processing speed of all jobs on the critical path is kept unchanged and the processing speed of each job on the non-critical path is adjusted, so as to achieve the lowest possible total carbon emission (TCE) under the same makespan (C_{max}) and further improve the quality of each non-dominated solution.

Step 7: Update the Pareto archive Ω^*. If the number of the non-dominated solutions in the non-dominated set Ω is larger than *ps*, then all of the feasible solutions are sorted by means of their crowded distances according to Algorithm 1, so as to eliminate some solutions with the smallest crowded distance until the number of feasible solutions in the non-dominated set (population) reaches *ps*.

Algorithm 6: Energy-saving speed adjustment strategy

Input: The processing speed matrix $\mathbf{V}$, and non-dominated set Ω.

Identify a critical path for each non-dominated solution in non-dominated set Ω.
Put all of the critical jobs into a set CJ.
for $f = 1$ *to* F **do**
  for $j = m$ *downto* 1 **do**
    for $i = n_f$ *downto* 1 **do**
      if $\pi_i^f \in CJ$ **then goto** Next Job.
      if $j = m$ and $i = n_f$ **then**
        Determine the product P_h to which job i belongs.
        $T_{margin} = S_{P_h}^A - C_{i,j} - p_{i,j} / v_{i,j}$.
      else if $j = m$ **then**
        $T_{margin} = C_{i+1,j} - p_{i+1,j} / v_{i+1,j} - (C_{i,j} - p_{i,j} / v_{i,j})$.
      else if $i = n_f$ **then**
        $T_{margin} = C_{i,j+1} - p_{i,j+1} / v_{i,j+1} - (C_{i,j} - p_{i,j} / v_{i,j})$.
      else
        $T_{margin} = \min\{C_{i,j+1} - p_{i,j+1} / v_{i,j+1}, C_{i+1,j} - p_{i+1,j} / v_{i+1,j}\} - (C_{i,j} - p_{i,j} / v_{i,j})$.
      end if
      $v = v_{i,j}$.
      for $t = 4$ *downto* 1 **do**
        if $v_{i,j} > v_t$ **then**
          if $T_{margin} \geq p_{i,j} / v_t$ **then**
            $v_{i,j} = v_t$. *//Obtain the minimum allowable speed within* T_{margin}.
          end if
        end if
      end for
      if $v \neq v_{i,j}$ **then**
        $C_{i,j} = C_{i,j} - p_{i,j} / v_{i,j} - T_{margin}$.
      end if
      Next Job:
    end for
  end for
end for

Output: The processing speed matrix $\mathbf{V}$, non-dominated set Ω.

Algorithm 7: Reduced-time speed adjustment strategy

Input: The processing speed matrix $\mathbf{V}$, and non-dominated set Ω.

Identify a critical path for each non-dominated solution in non-dominated set Ω.
Put all of the critical jobs into a set CJ. Execute operations in critical factory f_c as follows.
for $j=1$ *to* m **do**
  for $i=1$ *to* n_f **do**
    if $\pi_i^{f_c} \notin CJ$ **then goto** Next Job.
    if $i=1$ and $j=1$ **then**
      $C_{i,j} = C_{i,j} - \hat{p}_{i,j}^{k} + \hat{p}_{i,j}^{5}$. *//Adjust to maximum speed.*
    else if $j=1$ **then**
      $C_{i,j} = C_{i-1,j} + \hat{p}_{i,j}^{5}$.
    else
      $C_{i,j} = \max\{C_{i-1,j}, C_{i,j-1}\} + \hat{p}_{i,j}^{5}$.
    end if
    Next Job:
  end for
end for

Output: The processing speed matrix $\mathbf{V}$, non-dominated set Ω.

Algorithm 6) and the reduced-time speed adjustment strategy (see Algorithm 7), for all of the non-dominated solutions according to different preference levels. Then, update the Pareto archive Ω*.

Step 7: Determine whether the termination condition is satisfied. If not, then the program goes to Step 2, otherwise terminate the loop and output the currently obtained Pareto archive Ω*.

The flowchart of the proposed MCEDA for the EE_DAPFSP is illustrated in Fig. 8. According to the above steps, it can be seen that the MCEDA proposed in Section 3 effectively integrates many advantages of both the EDA-specific global exploration and the problem-dependent local exploitation. During each iteration of MCEDA, each new solution is generated by sampling from the promising regions in the solution space based on the multi-dimensional probabilistic model, and then the critical-path based local search with two speed adjustment strategies are performed for high-quality solutions, respectively. Since the multi-dimensional probabilistic model is updated based on high-quality subpopulation, the total characteristic distribution information of promising patterns from superior solutions can be well learned and stored, so that the sampling process can be concentrated around more potential regions in the solution space. Because of the performance of the scheduling schedule is well stressed and balanced by taking into account the benefit of both global exploration and local exploitation, it can be expected to achieve better results for solving the EE_DAPFSP.

3.5. The framework of MCEDA

According to the design above, the specific process of the proposed MCEDA for solving the EE_DAPFSP is as follows.

Step 1: (**Initialization**) Initialize the population $\boldsymbol{Pop}(\mathbf{0})$ via Algorithm 2, the multi-dimensional probabilistic model $\boldsymbol{PM}^{\mathbf{0}}_{n\times n\times n}$ by using Eq. (18), and key parameters of MCEDA, *i.e.*, ps,φ,r.

Step 2: Evaluate each individual in $\boldsymbol{Pop}(\boldsymbol{G})$ by using Eqs. (1–9), and calculate the matrix cube $\boldsymbol{MC}^{\boldsymbol{G}}_{n\times n\times n}$ according to Eqs. (13–14).

Step 3: Determine the Pareto dominance relationship of all individuals in $\boldsymbol{Pop}(\boldsymbol{G})$. Calculate the corresponding dominance level and crowding distance of these individuals via Algorithm 1. The top $ps\times\varphi$ excellent individuals are selected to form a high-quality subpopulation $\boldsymbol{SPop}(\boldsymbol{G})$ after sorting all individuals in $\boldsymbol{Pop}(\boldsymbol{G})$. Then, update the Pareto archive Ω*.

Step 4: (**Global exploration**) Update the multi-dimensional probabilistic model $\boldsymbol{PM}^{\boldsymbol{G}}_{n\times n\times n}$ by means of the incremental learning mechanism in Subsection 3.2.2, where the superior subpopulation $\boldsymbol{SPop}(\boldsymbol{G})$ is selected in $\boldsymbol{Pop}(\boldsymbol{G})$.

Step 5: (**Global exploration**) Sample the multi-dimensional probabilistic model $\boldsymbol{PM}^{\boldsymbol{G}}_{n\times n\times n}$ to generate new population $\boldsymbol{Pop}(\boldsymbol{G}+\mathbf{1})$ by using the specific sampling strategy in Subsection 3.2.3.

Step 6: (**Local exploitation**) Perform the critical path based local search in Subsection 3.3 for each non-dominated solution, respectively, and further execute two types of speed adjustment strategies in Subsection 3.4, *i.e.*, the energy-saving speed adjustment strategy (see

4. Experimental comparisons and statistical analysis

The following subsections are devoted to evaluate the overall performance of the proposed MCEDA for the considered EE_DAPFSP. First of all, some information about experiments is provided, which includes experimental setup and performance metrics. Then, the parameters of MCEDA are calibrated. Afterwards, the advantages of each improvement strategy in MCEDA are investigated. Lastly, comprehensive comparisons are conducted and experimental results are analyzed by comparing

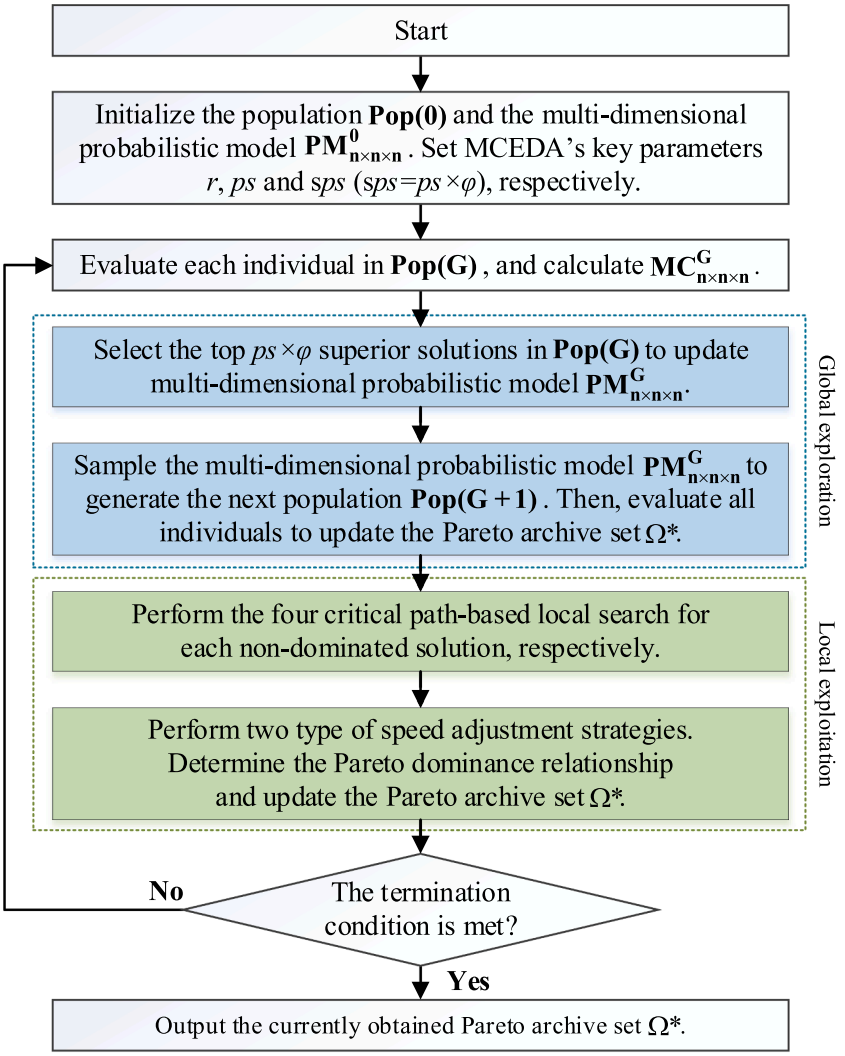


Fig. 8. The flowchart of MCEDA for the EE_DAPFSP.

MCEDA with several state-of-the-art multi-objective algorithms.

4.1. Experimental setup

In order to effectively and reasonably investigate the performance of MCEDA for addressing the considered EE_DAPFSP, in this section, extensive experiments and computational comparisons are carried out for the proposed MCEDA with several high-performing algorithms in the literature. Since the performance of the algorithm is usually greatly affected by different problem scales, we employ two well-known benchmark data sets presented by Hatami et al. (2013) and extend them by adding speed levels of machines in the processing stage to obtain two suitable testing sets for the EE_DAPFSP. The first testing set consists of 900 small-scale instances with 180 groups and each group contains 5 different instances, where $n = \{8,12,16,20,24\}$, $m = \{2,3,4,5\}$, $F = \{2,3,4\}$, and $S = \{2,3,4\}$. The second testing set is composed of 810 large-scale instances with 81 groups and each group includes 10 different instances, where $n = \{100,200,500\}$, $m = \{5,10,20\}$, $F = \{4,6,8\}$, and $S = \{30,40,50\}$. So, the total number of testing instances is 1710 and datasets are available at http://soa.iti.es. Since the running state of each machine in processing stage is variable and adjustable, the processing speed is selected from a series of discrete values $\{1, 1.3, 1.55, 1.75, 2.10\}$, that is, there are total five adjustable speed levels for each machine. The processing power consumption of machine M_j running at speed v_k is $E_{jk} = 4 \times v_k^2$ (kW) while the standby power consumption of machine M_j is $SE_j = 1$ (kW), and the speed adjustment is not considered in the assembly stage. In addition, it should be pointed out that calibrating algorithms by using the same benchmark instances that will later be adopted for computational comparisons constitute poor practices, which would result in over-fitting or biased experimental results (Pan & Ruiz, 2012). Therefore, it is of great importance to definitely distinguish between the calibrating instances and the final testing benchmark instances. For this reason, two types of new calibrating set that contain 180 instances for small-scale problems and 81 instances for large-scale problems are independently yielded based on the problem generation method provided by Hatami et al. (2013) for parameter calibration. To be specific, the small-scale calibrating instances contain complete combinations of $n = \{8,12,16,20,24\}$, $m = \{2,3,4,5\}$, $F = \{2,3,4\}$, and $S = \{2,3,4\}$ and the large-scale calibrating instances consist of $n = \{100,200,500\}$, $m = \{5,10,20\}$, $F = \{4,6,8\}$, and $S = \{30,40,50\}$. The processing times of jobs are randomly sampled from a uniform distribution in the range $[1,99]$, and the assembly times of each product P_h are generated according to a uniform distribution in the range $[1 \times N_h, 99 \times N_h]$.

In order to conduct computational comparisons fairly, the same experimental settings are adopted for all compared algorithms, including the same CPU Gigahertz frequency, programming language and termination criteria. All algorithms involved in this study are coded by Pascal language and compiled on Embarcadero RAD Studio XE8. The numerical experiments are independently executed on a PC with Inter

(R) Core(TM) i7-8700 M CPU @ 3.2 GHz processor and 16G of RAM under Microsoft Windows 7 OS. According to the analysis of the problem properties in Subsection 2.3, it is difficult to find the Pareto optimal frontier within a finite period of time since the feasible solution space of the considered problem is complex and multivariate. The performance of the proposed MCEDA are measured by comparisons with four related state-of-the-art multi-objective evolutionary algorithms (MOEAs), namely, non-dominated sorting genetic algorithm (NSGA-II) (Deb et al., 2002), modified multi-objective iterated greedy (MMOIG) algorithm (Ding et al., 2016), knowledge-based cooperative algorithm (KCA) (Wang & Wang, 2020), and multi-objective whale swarm algorithm (MOWSA) (Wang et al., 2020). The NSGA-II was first proposed by Deb et al. (2002). Due to its sample and efficient optimization performance, NSGA-II has been widely applied in a variety of multi-objective optimization fields, and it is recognized as one of the most effective algorithms for solving various multi-objective shop scheduling problems. The iterated greedy (IG) was first proposed by Ruiz and Stutzle (2008), it combined with the high efficiency of constructive heuristics and the advantages of simulated annealing approach, which is also regarded as a simple and effective algorithm for tackling different kinds of flow shop scheduling problems. The MMOIG is an effective extension of the typical IG algorithm in a multi-objective optimization perspective, where an extended NEH-Insertion is incorporated in the framework of MMOIG and several problem-dependent multi-neighborhood local searches are adopted to achieve satisfactory results in addressing the low-carbon PFSP (Ding et al., 2016). KCA is a high-performing algorithm recently proposed for handling the energy-efficient DPFSP. KCA can adopt a series of searching operators based on problem's characteristics and achieve multi-neighborhood cooperative search in the feasible solution space through control factors. MOWSA is another newly presented multi-objective algorithm for the energy-efficient DPFSP (Wang et al., 2020). In the literature, KCA and MOWSA have shown relatively good performance than other well-known MOEAs in solving the low-carbon DPFSP.

Taking into account the fairness, all tested algorithms are performed under the same termination conditions. That is, the maximum elapsed CPU time of the proposed MCEDA with NSGA-II, MMOIG, KCA and MOWSA is set to the same time commonly used in the literature, *i.e.*, $n \times m \times f \times 10$ milliseconds. In order to ensure the stability and reliability of the experimental results, computational comparisons are independently conducted 30 and 10 replications for small-scale instances and large-scale instances respectively, and all of the numerical results are collected and statistically averaged to eliminate random errors. It's worth noting that, for the largest testing instances of 500 jobs, 20 machines and 8 factories, the program needs to run about 800 s. Due to the proposed MCEDA and four compared algorithms are independently tested in 30 and 10 replications for small-scale and large-scale instances, a total number of $900 \times 5 \times 30 + 810 \times 5 \times 10 = 175500$ results for each performance metric would be yielded. As a result, for this larger dataset, most factors are statistically significant, which may allow us to draw strong conclusions.

4.2. Performance metrics

The metrics of MOEAs are significantly different from that of single-objective algorithms, so it is of great importance to comprehensively evaluate the convergence and diversity (distribution) of MOEAs (Wang & Wang, 2020). On the one hand, the high-performing MOEAs need to find more non-dominated solutions that should approximate to the true Pareto optimal front as closely as possible within an acceptable time. On the other hand, the distribution characteristics of the non-dominated solution sets obtained by MOEAs should be as widely as possible, so as to facilitate the decision maker to choose some suitable schemes by preferences. To verify the effectiveness and efficiency of MOEAs, the non-dominated solution set obtained by all algorithms are measured by some general metrics as follows.

(1) *Coverage Metric*: It can be used to measure the relatively quality between two Pareto archives obtained by two different algorithms A and B, denoted as $C(A,B)$. $C(A,B)$ gives a mapping from a pair of algorithms (A,B) into the interval $[0,1]$, which is represented as follows.

$$C(A,B) = \frac{1}{|B|}|\{\boldsymbol{b} \in B | \exists \boldsymbol{a} \in A : \boldsymbol{a} \succ \boldsymbol{b} \ or \ \boldsymbol{a} = \boldsymbol{b}\}|. \tag{21}$$

From Eq. (21), the coverage metric reflects the dominance relationship of feasible solutions from two non-dominated solution sets. If all solutions obtained by B are dominated by some solutions obtained by A, then $C(A,B) = 1$. Conversely, if all solutions obtained by B are not dominated by any solution obtained by A, then we have $C(A,B) = 0$. Since the solutions obtained by A and B do not necessarily dominate each other, it holds that $C(A,B) + C(B,A) \neq 1$.

(2) *Reference Distance* (DI_R): The reference distance metric DI_R is used to measure the distance of the elements in non-dominated solution set Ω' with respect to the reference set Ω^*, where Ω^* is composed of high-quality solutions in the Pareto archives yielded by all algorithms. Due to the NP-hard in strong sense with high complexity of the EE_DAPFSP, it is usually difficult to obtain the true Pareto optimal set, so the reference set Ω^* is composed of the Pareto archives aggregated jointly by all algorithms. The distance metric DI_R can be expressed as follows.

$$\mathrm{DI}_R(\Omega') = \frac{1}{|\Omega^*|}\sum_{y \in \Omega^*} min\{d(\boldsymbol{x},\boldsymbol{y}) | \boldsymbol{x} \in \Omega'\}. \tag{22}$$

where $d(\boldsymbol{x},\boldsymbol{y})$ is the Euclidean distance between $\boldsymbol{x} \in \Omega'$ and $\boldsymbol{y} \in \Omega^*$ in the normalized objective space (Ishibuchi et al., 2003), which can be calculated in Eq. (23).

$$d(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_{i=1}^{G}\left[(f_i(\boldsymbol{x}) - f_i(\boldsymbol{y}))/\left(f_i^{max} - f_i^{min}\right)\right]^2}. \tag{23}$$

In Eq. (23), f_i is the ith objective function, and f_i^{max} and f_i^{min} are the maximum and minimum values of the objective function f_i, respectively. Obviously, $\mathrm{DI}_R(\Omega')$ is expected to be as small as possible, indicating that Ω' is closer to reference set Ω^*, which means the algorithm has better performance.

(3) *Distribution Spacing* (*DS*): The distribution spacing metric *DS* is adopted to measure the distribution uniformity of solutions obtained by a specified algorithm. The distribution spacing metric can be calculated as follows.

$$DS = \sqrt{\frac{1}{|\Omega'|}\sum_{i=1}^{|\Omega'|}(D_i - \overline{D})^2}/\overline{D}. \tag{24}$$

where $\overline{D} = \sum_{i=1}^{|\Omega'|} D_i/|\Omega'|$. D_i denotes the minimum Euclidean distance between the ith feasible solution in the solution set Ω' and the other feasible solutions in Ω'. From Eq. (24), it is obvious that the smaller the value of *DS* is, the more evenly distributed of the solutions in Ω' are.

(4) *Non-dominance Ratio* (ρ_r): It is used to measure the proportion of the non-dominated solution set Ω' obtained by a specific algorithm in the reference set Ω^*. Obviously, the larger the ρ_r, the better the performance of the algorithm.

Furthermore, to increase the soundness of our conclusion, two state-of-the-art quality metrics, *i.e.*, Hypervolume (Zitzler & Thiele, 1999) and Unary epsilon (Zitzler et al., 2003), are also adopted to evaluate quality of the found non-dominated solution sets, which are described as

follows:

(5) *Hypervolume* (I_H): The hypervolume represents the volume of the hypercube enclosed by all solutions in a given Pareto front PF_G and the reference point $z^r = (z_1^r, z_2^r)$ in the objective space. Each objective value of each solution $x \in PF_G$ is normalized into $[0, 1]$ before calculating the I_H value. The normalized value $f_i^{'}(x)$ for the ith objective of solution x can be calculated as

$$f_i^{'}(x) = \left(f_i(x) - f_i^{min}\right) / \left(f_i^{max} - f_i^{min}\right). \tag{25}$$

where f_i^{min}, f_i^{max} and $f_i(x)$ have the same meanings with Eq.(23). Then, the hypervolume indicator I_H for the bi-objective problems can be calculated using Lebesgue measurement in Eq.(26).

$$I_H(PF_G) = \sum_{k=1}^{|PF_G|} \left(z_1^r - f_1^{'}(x_k)\right) \times \left(f_2^{'}(x_k) - f_2^{'}(x_{k-1})\right). \tag{26}$$

The accuracy of calculating I_H depends on the choice of the reference point, *i.e.*, when evaluating the same Pareto front (non-dominated solution set), selecting different reference points will result in different results. According to the references (Minella et al., 2008; Ciavotta et al., 2013), the reference point is usually selected as $(1.2, 1.2)$. The hypervolume is a Pareto-compliant evaluation metric, which means that if one Pareto front PF_G^1 is better than another Pareto front PF_G^2, then the Hypervolume metric of PF_G^1 will be greater than that of PF_G^2. The larger value of the hypervolume indicator I_H, the better convergence as well as a good coverage of the optimal Pareto front.

(6) *Unary Epsilon* (I_ε^1): It is used to measure the minimum distance between a given Pareto front PF_G and the reference (optimal) Pareto front PF_R. To avoid errors arising from dividing by zero in the calculation of I_ε^1, each objective value $f_i(x)$ should be normalized into $[1, 2]$ as follows.

$$f_i^{'}(x) = \left(f_i(x) - f_i^{min}\right) / \left(f_i^{max} - f_i^{min}\right) + 1. \tag{27}$$

According to Eq.(27), the I_ε^1 value varies between 1 and 2. If the I_ε^1 value close to 1 implies that the given Pareto front is close to the reference Pareto front, whereas the I_ε^1 value close to 2 means that it is distant. Then, the bi-objective I_ε^1 can be calculated as

$$I_\varepsilon^1(PF_G, PF_R) = \max_{y \in PF_R} \min_{x \in PF_G} \max_{1 \leqslant i \leqslant 2} \left(f_i^{'}(x) / f_i^{'}(y)\right). \tag{28}$$

Since the true Pareto front for each instance is unknown, a union set constituted by aggregated all non-dominated solutions obtained by all algorithms, is regarded as the reference (optimal) Pareto front PF_R. It is clear that a smaller I_ε^1 value means a better approximation to the reference Pareto front.

Notice that, it is unnecessary to immediately evaluate the Pareto archives obtained by each algorithm after finishing each iteration. The Pareto archives yielded by each of the algorithms are measured if and only if all of the algorithms have been run and all non-dominated solutions have been collected. Then the maximum and minimum values of each objective function can be determined, so the minimum and maximum values are fixed, which is fair for the evaluation of each non-dominated solution.

4.3. Parameter calibration

Since parameter calibration plays an important role in the development of high performing metaheuristics, the reasonable values of parameters have remarkable impact on the effectiveness and efficiency of the designed algorithms. Meanwhile, it should be point out that the statistical calibration is only a fine-tuning process and stochastic algorithms are not expected to behave entirely different after calibration. In this section, the Design-of-Experiments (DOE) methodology (Montgomery, 2008) is employed to analyze the sensitivity of main parameters and further investigate the effects of each parameter on the performance of the proposed MCEDA. There are three control parameters in MCEDA, *i.e.*, the population size (ps), the percentage of superior subpopulation (φ), and the learning rate (r). A series of potential values (levels) of parameters (factors) are firstly considered through summarizing previous relevant literature (Wang & Wang, 2016; Zhang et al., 2021), and then the suitable scope of each parameter is determined according to some preliminary experiments. The considered levels for all factors are listed in Table 3. As seen in Table 3, a total of $5 \times 5 \times 5 = 125$ different configurations for the proposed MCEDA are yielded. Thus, a full factorial experimental design is considered for all configurations, and some additional instances generated by ourselves are adopted as the test bed, which includes 180 basic instances for small-scale problems and 81 basic instances for large-scale problems. Each configuration is tested on these 261 instances where 10 independent replications are performed under the same parameter combination for each instance. As a result, there are a total of $125 \times 261 \times 10 = 326250$ treatments, which implies that a total of 326250 results (non-dominated sets) would be yielded. The hypervolume (I_H), unary epsilon (I_ε^1), and non-dominance ratio (ρ_r) indicators are regarded as three response values, respectively. The maximum elapsed CPU time of $n \times m \times f \times 10$ milliseconds is used as the termination criterion, so it requires at least 74.375 CPU days to complete all calibration experiments. Due to multi-cores in our personal

Table 3
The levels of parameters.

Parameters	Factor level				
	1	2	3	4	5
ps	10	30	60	90	120
φ	0.1	0.2	0.3	0.4	0.5
r	0.05	0.1	0.2	0.3	0.4

Table 4
The results of ANOVA over calibrating the parameters of MCEDA for small-scale instances.

Source	Hypervolume (I_H)					eUnary Epsilon (I_ε^1)					Non-dominance ratio (ρ_r)				
	Sum of squares	*Df*	Mean square	*F*-radio	*p*-value	Sum of squares	*Df*	Mean square	*F*-radio	*p*-value	Sum of squares	*Df*	Mean square	*F*-radio	*p*-value
ps	0.03916	4	0.00979	1271.58	**0.0000**	0.11476	4	0.02869	4379.97	**0.0000**	0.04759	4	0.01190	1180.86	**0.0000**
φ	0.01868	4	0.00467	606.39	**0.0000**	0.04666	4	0.01166	1780.89	**0.0000**	0.01199	4	0.00300	297.49	**0.0000**
r	0.03730	4	0.00933	1211.06	**0.0000**	0.04469	4	0.01117	1705.77	**0.0000**	0.00841	4	0.00210	208.75	**0.0000**
ps*φ	0.00012	16	0.00001	1.00	0.4683	0.00011	16	0.00001	1.04	0.4310	0.00020	16	0.00001	1.21	0.2847
ps*r	0.00006	16	0.00000	0.48	0.9480	0.00008	16	0.00000	0.73	0.7506	0.00005	16	0.00000	0.32	0.9933
φ*r	0.00019	16	0.00001	1.52	0.1205	0.00025	16	0.00002	2.41	**0.0067**	0.00009	16	0.00001	0.57	0.8977
Residual	0.00049	64	0.00001			0.00042	64	0.00001			0.00064	64	0.00001		
Total	0.09600	124				0.20696	124				0.06897	124			

Table 5
The results of ANOVA over calibrating the parameters of MCEDA for large-scale instances.

Source	Hypervolume (I_H)					Unary Epsilon (I_ϵ^1)					Non-dominance ratio (ρ_r)				
	Sum of squares	*Df*	Mean square	*F*-radio	*p*-value	Sum of squares	*Df*	Mean square	*F*-radio	*p*-value	Sum of squares	*Df*	Mean square	*F*-radio	*p*-value
ps	0.02195	4	0.00549	1662.67	**0.0000**	0.00421	4	0.00105	109.99	**0.0000**	0.18919	4	0.04730	6306.40	**0.0000**
φ	0.02188	4	0.00547	1657.21	**0.0000**	0.00760	4	0.00190	198.56	**0.0000**	0.04943	4	0.01236	1647.73	**0.0000**
r	0.02144	4	0.00536	1624.48	**0.0000**	0.00305	4	0.00076	79.71	**0.0000**	0.04838	4	0.01209	1612.53	**0.0000**
*ps**φ	0.00004	16	0.00000	0.85	0.6279	0.00021	16	0.00001	1.38	0.1812	0.00018	16	0.00001	1.47	0.1408
*ps***r*	0.00036	16	0.00002	6.76	**0.0000**	0.00004	16	0.00000	0.28	0.9966	0.00023	16	0.00001	1.93	**0.0330**
φ**r*	0.00011	16	0.00001	2.06	**0.0217**	0.00013	16	0.00001	0.86	0.6193	0.00003	16	0.00000	0.27	0.9975
Residual	0.00021	64	0.00000			0.00061	64	0.00001			0.00048	64	0.00001		
Total	0.06599	124				0.01587	124				0.28792	124			

Note: All F-ratios are based on the residual mean square error. Boldface indicates that is significant at the 0.05 level.

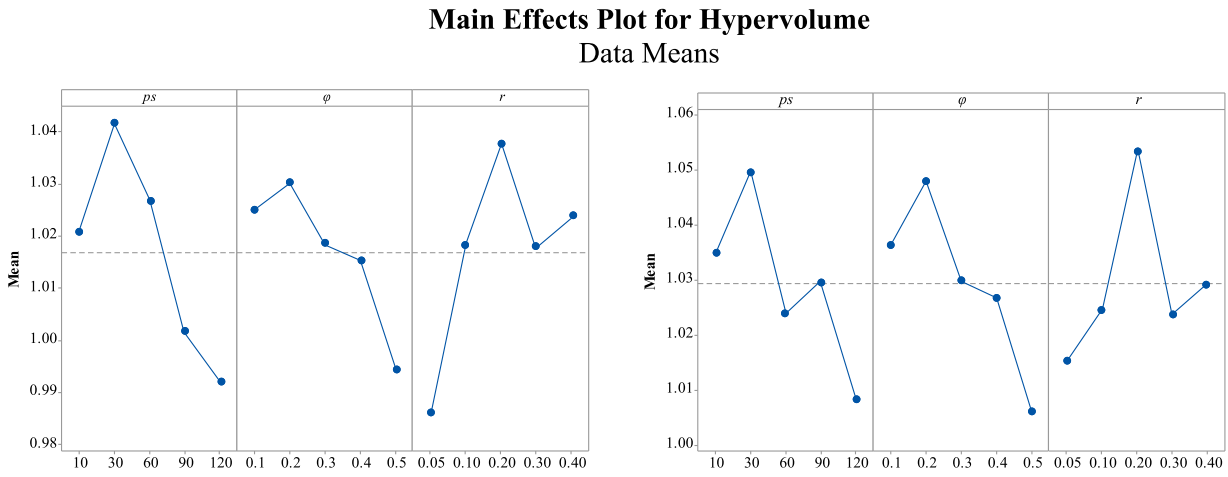
computer, actually almost 2.5 days are adopted to execute the whole experiment.

All experimental results are analyzed by means of the multi-factor Analysis of Variance (ANOVA), which has been widely applied as a powerful parametric statistical technique in many scheduling literatures (Shao et al., 2018; Sang et al., 2019; Pan et al., 2019). In the ANOVA, three main hypotheses (*i.e.*, normality, homoscedasticity, and independence of residuals), have to be checked and accepted. According to the residual analysis of the experimental results, all assumptions are easily satisfied. The ANOVA results of three parameters of MCEDA are reported in Tables 4 and 5. For the results of ANOVA, the *F*-ratio is regard as a clear indicator of significance when *p*-value is less than the confidence level. A large *F*-ratio means that the analyzed factor has a considerable effect on the response variable. As seen in Tables 4 and 5, all parameters are statistically significant since their *p*-values are smaller than the 0.05 confidence level for three performance metrics, *i.e.*, hypervolume, unary epsilon, and non-dominance ratio. Among these three parameters, the allowable maximum number of population (*ps*) achieves the largest *F*-ratio, which indicates that *ps* has the most significant effect on the performance of the proposed MCEDA for both the small-scale instances and the large-scale instances.

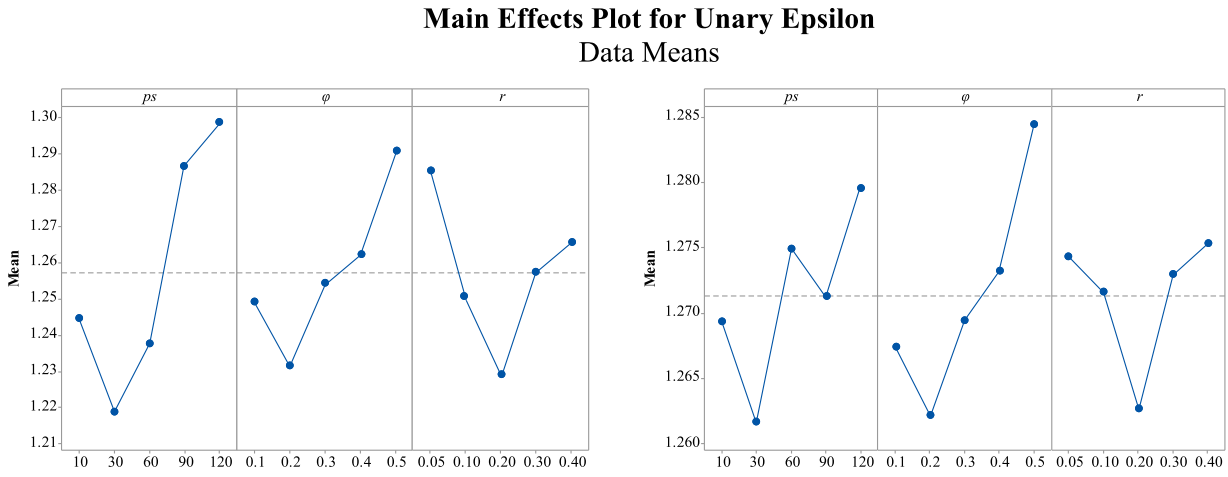
Fig. 9 provides the main effects plots of all parameters for different scale testing sets. It is clearly observed from this figure that the choice of $ps = 30$ yields the best performance while $ps = 120$ obtains the worst results. To be specific, a small population is favorable to perform more iterations and it is beneficial to achieve a deeper exploitation (local search) in the feasible solution space. However, if the population size is too small, then the size of the superior subpopulation may be affected, resulting the failure of the multi-dimensional probabilistic model to fully learn the excellent structural characteristics of the superior solutions. The results of the proposed MCEDA degrades with the increasing of the population size, and it can be observed from Fig. 9 that the performance has a considerable change, especially from $ps = 30$ to $ps = 120$. Although a larger population is helpful to prompt the diversity of the obtained non-dominated solutions, it also consumes much more computational cost and reduces the convergence speed, and thereby reduce the searching efficiency. In fact, if we adopt the large population, it may affect the algorithm's ability to perform more iterations, especially for addressing the large instances. Thus, the population size *ps* should be set as a relatively small value, *i.e.*, $ps = 30$. The second largest *F*-ratio value corresponds to the percentage of superior subpopulation φ. It can be observed in Fig. 9 that the value $\varphi = 0.2$ yields the best results, while $\varphi = 0.5$ results the worst performance. Moreover, a small-scale superior subpopulation is more conducive to accurately learn the information of both structural features and promising patterns from high-quality solutions, so the probabilistic model can be updated effectively.

From Tables 4-5, we can see that the factor *r* is the last significant parameter. The learning rate can control the balance of information fusion between the matrix cube and the multi-dimensional probabilistic model. To be specific, the larger *r* tends to learn more valuable information from the selected superior solutions in each generation, while the smaller *r* reinforces the accumulated information of superior solutions during the overall iterative process. Thus, *r* should be determined by considering the trade-off between the current knowledge and historical experience, and a suitable learning rate helps the algorithm to avoid premature convergence or slow convergence as much as possible (Wang & Wang, 2016). Fig. 9 reveals that the algorithm has good performance when *r* is equal to 0.2, which verifies the conclusion above. According to the parametric experiment results and analysis above, for two different scale testing sets, the best configuration of parameters for MCEDA is suggested as: $ps = 30$, $\varphi = 0.2$, $r = 0.2$.

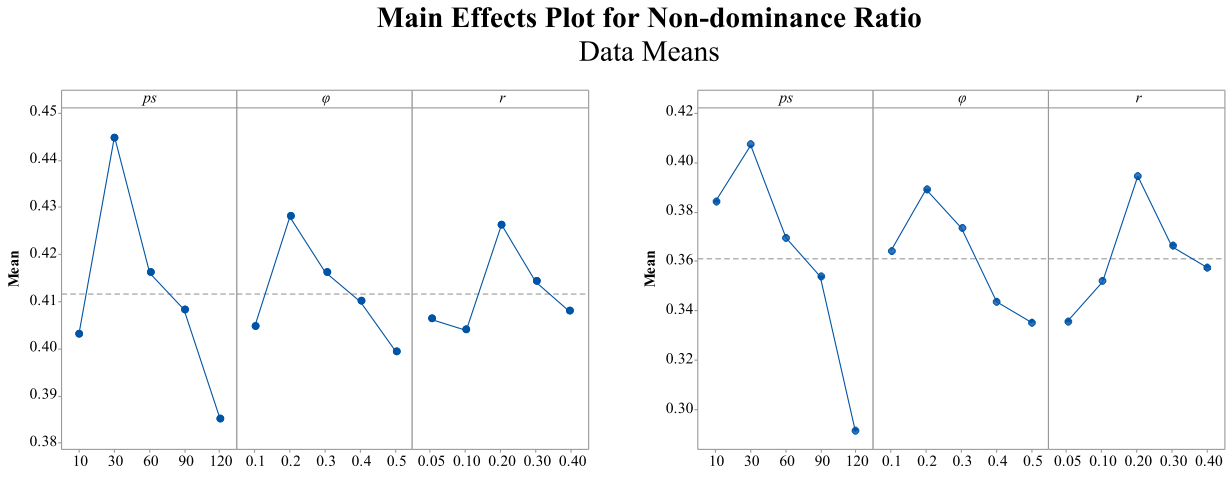
In order to ensure the fairness of the computational comparisons, this section further performs some additional parameter calibrations for NSGA-II, MMOIG, KCA and MOWSA by using the same multi-factor ANOVA technique. For all of the five compared algorithms, it should be noted that the population size *ps* is a common parameter which is selected as $ps = 30$ to make a fair comparison. In addition, to ensure the diversity of the initial population, five different discrete speed values of 1, 1.3, 1.55, 1.75, and 2.10 are adopted to yield the initial speed matrix, respectively. In NSGA-II, crossover probability (p_c) and mutation probability (p_m) are two crucial parameters, which are set as $p_c = 0.8$ and $p_m = 0.1$. In MMOIG, the number of destructed jobs (d) and mutation probability (ρ) in the destruction phase are two key parameters, which are set as $d = 3$ and $\rho = 0.4$ following the original literature. The best parameter combination of KCA is set as: the depth of local intensification $LS = 100$, and the proportion of EENEHFF2-based initialization $PE = 60$. The crossover probability (α) and mutation probability (β) in MOWSA are set as: $\alpha = 0.9$ and $\beta = 0.2$. Notice that, it is meaningless to determine all parameters by means of the main effects plot, if some significant interactions are existed between factors. Due to the space of the paper, 2-level interaction plots of pair factors (*i.e.*, *ps**φ, *ps***r* and φ**r*) on three performance metrics are provided only for the large-scale instances. It is clear in Fig. 10 that the significance of these 2-level interactions is relatively weak and these results are in accordance with the conclusions above. Meanwhile, it can be observed from Tables 4-5 that the *F*-ratio of each parameter is larger than that of their interactions, which further reveals the proposed MCEDA obtains the best performance when $ps = 30$, $\varphi = 0.2$, $r = 0.2$.



(*a*) The level trend for small-scale instances. (*b*) The level trend for large-scale instances.



(*c*) The level trend for small-scale instances. (*d*) The level trend for large-scale instances.



(*e*) The level trend for small-scale instances. (*f*) The level trend for large-scale instances.

Fig. 9. Main effects plots of parameters for hypervolume, unary epsilon, and non-dominance ratio.

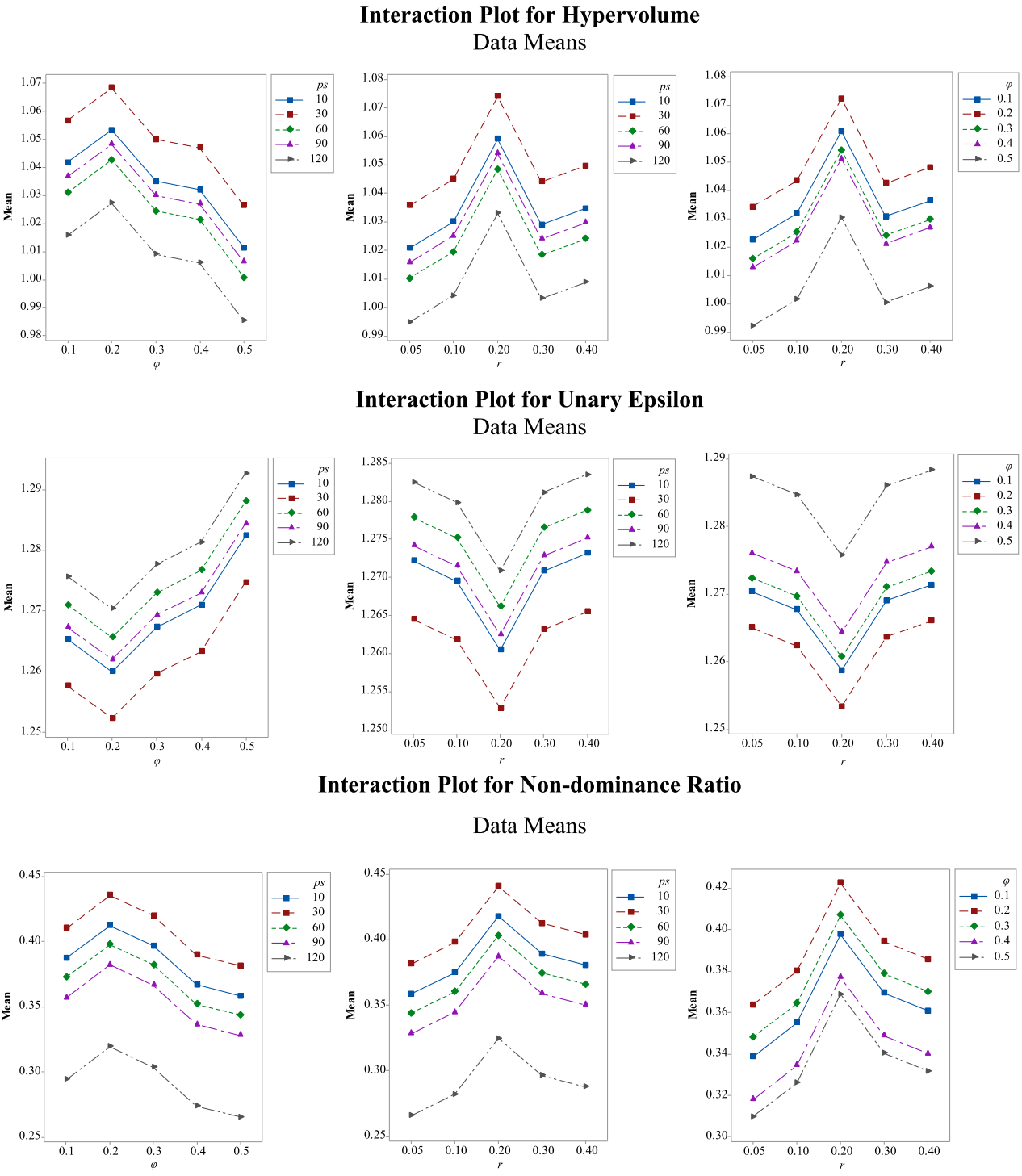


Fig. 10. Interaction plots for ps*φ, ps*r and φ*r for large-scale instances.

4.4. *Effectiveness of probabilistic models*

In MCEDA, the matrix-cube-based multi-dimensional probabilistic model is used to learn and estimate the characteristic distribution of promising pattern from some selected superior solutions, so as to guide the searching directions toward potential regions. Since various high-performing EDAs usually employ one or more two-dimensional probabilistic models to guide the global search direction (Jarboui et al., 2009; Pan & Ruiz, 2012; Tiwari et al., 2014; Wang & Wang, 2016), it is of necessity to make a fair investigation on the performance of EDA's global exploration. In this subsection, the global search framework of the proposed MCEDA (denoted as MCEDA$_{nls}$) is compared with three effective two-dimensional model-based EDAs, *i.e.*, an effective EDA designed by Wang and Wang (2016) (denoted as EEDA$_{nls}$), a state-of-the-art EDA presented by Jarboui et al. (2009) (denoted as JEDA$_{nls}$), and a modified PEDA developed by Pan and Ruiz (2012) (denoted as PEDA$_{nls}$). Notice that the corresponding local search parts are removed from each of these three types of EDAs and only the global search is retained. Moreover, the parameters of three compared algorithms are set the same as in the original literature. The comparison of MCEDA$_{nls}$ against

Table 6
Statistical results of MCEDA$_{nls}$, EEDA$_{nls}$, JEDA$_{nls}$ and PEDA$_{nls}$ for five performance metrics.

n	EEDA$_{nls}$					JEDA$_{nls}$					PEDA$_{nls}$					MCEDA$_{nls}$				
	DI_R	ρ_r	DS	I_H	I_ε^1	DI_R	ρ_r	DS	I_H	I_ε^1	DI_R	ρ_r	DS	I_H	I_ε^1	DI_R	ρ_r	DS	I_H	I_ε^1
8	0.252	0.228	2.373	1.312	1.082	0.413	0.082	2.852	1.146	1.168	0.322	0.163	2.632	1.236	1.133	**0.161**	**0.542**	**2.181**	**1.338**	**1.037**
12	0.214	0.263	2.431	1.283	1.096	0.352	0.096	2.931	1.125	1.192	0.273	0.192	2.771	1.212	1.146	**0.134**	**0.464**	**2.264**	**1.324**	**1.064**
16	0.231	0.282	2.613	1.276	1.108	0.421	0.088	3.184	1.107	1.243	0.315	0.231	2.854	1.183	1.164	**0.152**	**0.412**	**2.332**	**1.296**	**1.078**
20	0.283	0.244	2.882	1.262	1.115	0.514	0.062	3.262	0.982	1.284	0.381	0.154	3.122	1.162	1.255	**0.171**	**0.551**	**2.458**	**1.283**	**1.087**
24	0.317	0.272	3.065	1.224	1.131	0.573	0.083	3.347	0.958	1.316	0.442	0.182	3.274	1.145	1.273	**0.194**	**0.475**	**2.534**	**1.275**	**1.098**
Average	0.259	0.258	2.673	1.271	1.106	0.455	0.082	3.115	1.064	1.241	0.347	0.184	2.931	1.188	1.194	**0.162**	**0.489**	**2.354**	**1.303**	**1.073**

Table 7
Statistical results of all variants of MCEDA.

n	MCEDA$_{v1}$					MCEDA$_{v2}$					MCEDA$_{v3}$					MCEDA$_{nls}$				
	DI_R	ρ_r	DS	I_H	I_ε^1	DI_R	ρ_r	DS	I_H	I_ε^1	DI_R	ρ_r	DS	I_H	I_ε^1	DI_R	ρ_r	DS	I_H	I_ε^1
8	0.073	0.065	2.336	1.263	1.105	0.271	0.012	2.821	1.146	1.188	0.145	0.027	2.651	1.195	1.154	**0.012**	**0.907**	**2.154**	**1.325**	**1.014**
12	0.115	0.094	2.412	1.247	1.113	0.314	0.025	2.953	1.125	1.217	0.194	0.044	2.783	1.171	1.173	**0.014**	**0.846**	**2.271**	**1.311**	**1.023**
16	0.132	0.141	2.656	1.225	1.118	0.382	0.041	3.232	1.107	1.254	0.242	0.062	2.912	1.153	1.187	**0.032**	**0.755**	**2.354**	**1.295**	**1.036**
20	0.174	0.172	2.914	1.211	1.124	0.434	0.044	3.314	0.982	1.277	0.275	0.114	3.154	1.126	1.195	**0.054**	**0.687**	**2.442**	**1.283**	**1.058**
24	0.191	0.231	3.172	1.184	1.143	0.476	0.072	3.466	0.958	1.305	0.332	0.122	3.282	1.104	1.223	**0.061**	**0.576**	**2.558**	**1.228**	**1.082**
Average	0.137	0.141	2.698	1.226	1.121	0.375	0.039	3.157	1.064	1.248	0.238	0.074	2.956	1.150	1.186	**0.035**	**0.754**	**2.356**	**1.288**	**1.043**

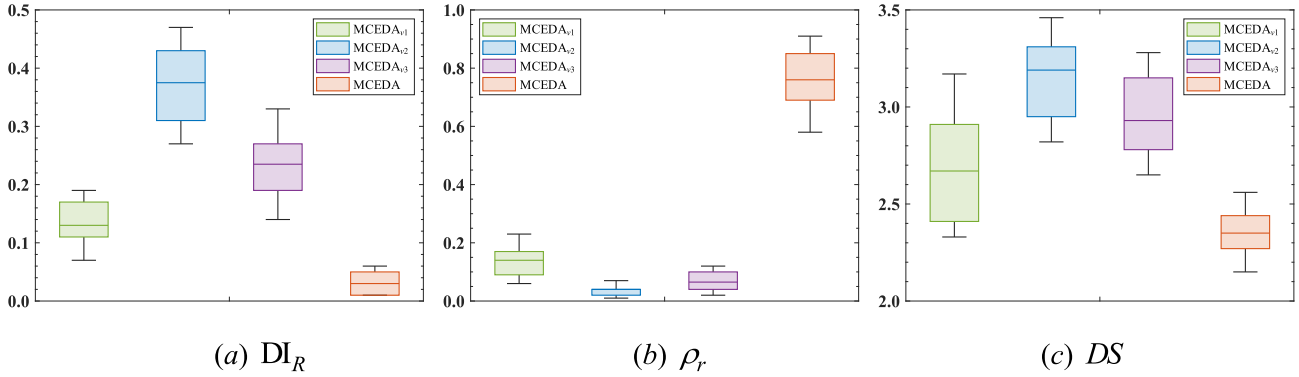


Fig. 11. The box plots of MCEDA$_{v1}$, MCEDA$_{v2}$, MCEDA$_{v3}$, and MCEDA.

EEDA$_{nls}$, JEDA$_{nls}$ and PEDA$_{nls}$ is executed on small-scale instances by using the same elapse CPU time as a termination criterion. All algorithms independently perform 30 times for each instance, and the average DI$_R$, ρ_r, *DS*, I_H, I_ε^1 obtained are used as the measure metrics.

The statistical results grouped by the number of jobs are reported in Table 6, where each cell is averaged across 180 small-scale instances and 30 replicates per instance (5400 values in total). The best value of each group is highlighted with **boldface** in Table 6. It is clear from Table 6 that the the average DI$_R$, ρ_r, and *DS* values obtained by MCEDA$_{nls}$ are obviously better than those obtained by PEDA$_{nls}$, JEDA$_{nls}$ and EEDA$_{nls}$ for all instances. As seen in Table 6, the MCEDA$_{nls}$ significantly outperforms other three probabilistic model based algorithms on the overwhelming major of instances in terms of both I_H and I_ε^1 indicators, which indicates the obtained approximated Pareto set is closer to the referenced Pareto front and better quality. The main reason is that the three-dimensional probabilistic model in MCEDA$_{nls}$ can save the valuable information of each superior individual in a more accurate and reasonable way. Nevertheless, the two-dimensional probabilistic models in the compared algorithms cannot save the position of each similar block and just simply record all of the same similar blocks in one place of the two-dimensional matrix. As a result, the similar blocks cannot be placed in the right positions when generating new individual, which leads a relatively poor search ability of these compared algorithms.

4.5. *Effectiveness of improvement strategies*

As stated in Section 3, the proposed MCEDA has three main improvement strategies: (1) the population initialization method in subection 3.1; (2) the critical path based local search in Subsection 3.3; (3) the speed adjustment strategy in Subsection 3.4. To investigate the effectiveness and efficiency of these improvement strategies, some variants of MCEDA are implemented in this subsection. To be specific, for the population initialization method, a variant of MCEDA (denoted as MCEDA$_{v1}$) is developed. MCEDA$_{v1}$ does not adpot the presented population initialization method, and it removes the heuristic method in Algorithm 1 and only use the random initialization to produce the initial population. For the critical path based local search, another variant of MCEDA named MCEDA$_{v2}$ is developed. MCEDA$_{v2}$ does not apply the the critical path based local search, and it is otherwise the same as MCEDA. For the speed adjustment strategy, we implement a variant of MCEDA without the speed adjustment strategy (denoted as MCEDA$_{v3}$) which is used to confirm the presented speed adjustment strategy whether promotes the quality of the non-dominated solutions obtained in the local search. It should be clarified that each variant only modifies a single component of the proposed MCEDA, and the performance of MCEDA, MCEDA$_{v1}$, MCEDA$_{v2}$, and MCEDA$_{v3}$ is compared based on small-scale testing set in the identical elapse CPU time as a termination criterion. The same parameter settings are used for MCEDA and its three variants, and all algorithms are independently performed for 30 times on each instance, and the average DI$_R$, ρ_r, and *DS* obtained are used as the measure metrics. All of the test instances are grouped according to the number of jobs, and the statistical results for MCEDA and its variants are reported in Table 7, in which each value is averaged across 180 test instances and 30 replicates per instance (5400 values). The best value of each group is highlighted with **boldface** in Table 7.

According to Table 7, it can be seen that the MCEDA significantly outperforms the other three variants on the overwhelming major of five different types of job sizes in terms of three metrics, DI$_R$, ρ_r, and *DS*, especially significantly better in the ρ_r metric, which indicates that all improvement strategies can effectively improve the quality of solutions. As can be observed from Table 7, MCEDA is better than MCEDA$_{v2}$ in all instance groups, which demonstrates that the effectiveness of the critical path based local search. In fact, since the critical path of the considered problem directly determines the maximum completion time, it is necessary to allocate the limited computing resource of local exploitation along the critical path direction to adjust the critical jobs instead of non-critical ones. The critical path-based neighborhood search effectively enhances the local intensification ability of the algorithm, and exhaustively exploit the potential area around promising solutions, making them as close to the optimal Pareto front as possible. As revealed in Table 7, the results of MCEDA outperform MCEDA$_{v3}$, which indicates the effectiveness of the speed adjustment strategy for the considered problem. The presented two types of speed adjustment strategies not only can control the production process according to different situations, but also can complement each other with regard to the two criteria to jointly improve the quality of the obtained non-dominated solutions. In addition, the energy-saving speed adjustment strategy can enable the algorithm to obtain more high-quality non-dominated solutions with low energy consumption during each iteration. As regards MCEDA$_{v1}$, its performance is inferior to MCEDA, which suggests that the adopted heuristic method make the initial candidate solutions converge toward a possible promising region during the early stage. Meanwhile, a random initialization is added to ensure the wide distribution of the initial solutions in the solution space. Thus, we can obtain initial solutions with higher quality and better diversity. In Table 7, we can also see that the results of MCEDA$_{v2}$ and MCEDA$_{v3}$ are inferior to MCEDA at all test instances, which implies that better performance can be reached by combining the critical path based local search and the speed adjustment strategy.

Moreover, the statistical results on *DS* metrics show the dispersion of the non-dominated solutions obtained by MCEDA is better. To analyze the results from a statistical perspective, Fig. 11 shows the box plots of MCEDA and its variants corresponding to the three performance metrics. As seen in these figures, MCEDA is significantly better than other variants on all test instances, which indicates the proposed MCEDA has the

Table 8
Statistical results on *C* metric of MCEDA with NSGA-II, MMOIG, KCA and MOWSA for small-scale instances.

$F \times n$	*C*(MCEDA, NSGA-II)	*C*(NSGA-II, MCEDA)	*C*(MCEDA, MMOIG)	*C*(MMOIG, MCEDA)	*C*(MCEDA, KCA)	*C*(KCA, MCEDA)	*C*(MCEDA, MOWSA)	*C*(MOWSA, MCEDA)
2×8	0.99	0.00	0.98	0.01	0.95	0.02	0.91	0.06
2×12	0.97	0.00	0.97	0.01	0.93	0.04	0.87	0.08
2×16	0.96	0.01	0.94	0.02	0.88	0.05	0.83	0.07
2×20	0.94	0.01	0.92	0.03	0.84	0.08	0.81	0.11
2×24	0.92	0.01	0.87	0.06	0.86	0.10	0.82	0.13
3×8	0.96	0.00	0.96	0.01	0.95	0.04	0.91	0.07
3×12	0.95	0.02	0.93	0.03	0.91	0.07	0.85	0.09
3×16	0.93	0.01	0.91	0.05	0.87	0.09	0.84	0.14
3×20	0.91	0.01	0.88	0.04	0.85	0.11	0.82	0.13
3×24	0.90	0.03	0.85	0.06	0.82	0.13	0.78	0.15
4×8	0.95	0.00	0.94	0.00	0.91	0.04	0.89	0.07
4×12	0.93	0.01	0.91	0.02	0.88	0.07	0.84	0.09
4×16	0.91	0.00	0.87	0.04	0.83	0.11	0.81	0.14
4×20	0.86	0.02	0.83	0.05	0.79	0.12	0.74	0.15
4×24	0.87	0.03	0.84	0.07	0.81	0.15	0.77	0.18
Average	0.93	0.01	0.91	0.03	0.87	0.08	0.83	0.11

best search capability and further confirms the conclusion drawn above. Therefore, according to the above results and analysis, it can be concluded that these improvement strategies have great promotion on improving the performance of MCEDA.

4.6. Comparisons of MCEDA and existing algorithms

To further validate the effectiveness of the proposed algorithm, MCEDA is compared with four state-of-the-art multi-objective algorithms, *i.e.*, NSGA-II, MMOIG, KCA, and MOWSA. Then, the relative quality of the non-dominated sets yielded by each algorithm is measured and evaluated on the *C* metric, respectively. To the best of our knowledge, no algorithms are recently presented in the literature to tackle the considered EE_DAPFSP. Since these high-performing compared algorithms are not directly designed for the EE_DAPFSP, we reimplement these four algorithms and adjust their objective evaluation functions in the original literature to make them be able to solve this problem. Notice that all of these algorithms are adapted to the sequence-based model, so they can be easily extended and participated in comparisons. All algorithms are performed on the same experiment environment which has same CPU power available. Each algorithm independently runs 30 and 5 replicates on two benchmark sets of different problem sizes. The Pareto reference set of each instance is jointly composed of all non-dominated sets obtained by all algorithms. The computational results are grouped according to different numbers of factories and jobs, denoted by $F \times n$, where 60 instances per average for each group of the small-scale benchmark set and 90 instances per average for each group of the large-scale benchmark set.

The statistical results of the coverage metric of the proposed MCEDA with NSGA-II, MMOIG, KCA and MOWSA on two benchmark sets of different sizes are reported in Tables 8 to 9, respectively. As seen from Tables 8-9, MCEDA yields good results that are, on average, almost better than that obtained by its counterparts for all testing instances. For the 900 small-scale instances, it is clear that almost 87% and 83% of the feasible solutions in the non-dominated set obtained by KCA and MOWSA are dominated by some of the feasible solutions in the non-dominated set obtained by MCEDA in the average sense. In other words, only an average of 8% and 11% of solutions obtained by MCEDA are dominated by some feasible solutions yielded by KCA and MOWSA, respectively. In addition, for these non-dominated sets obtained by other two types of classical multi-objective algorithms, *i.e.*, NSGA-II and MMOIG, almost average of 93% and 91% of feasible solutions are dominated by some solutions in the non-dominated set produced through MCEDA, which indicates that MCEDA has good performance in solving the small-scale instances of the EE_DAPFSP. Similarly, it can be clearly seen from Table 9 that, for the 810 large-scale instances, the performance of MCEDA is overwhelming on the coverage metric *C*. That is, the average 94%, 90%, 79%, and 74% of the non-dominated solutions obtained by four counterparts, *i.e.*, NSGA-II, MMOIG, KCA and MOWSA, are dominated by some of solutions in non-dominated set obtained by MCEDA, while only 1%∼9% of the solutions in the non-dominated set of MCEDA are dominated by some solutions obtained by NSGA-II, MMOIG, KCA and MOWSA in an average sense, which further demonstrates that the proposed MCEDA can also be addressing the large-scale instances of the EE_DAPFSP. Furthemore, it is noted that the recently proposed KCA and MOWSA are also two relatively excellent multi-objective algorithms compared with NSGA-II and MMOIG. According to above analysis, we can conclude that the proposed MCEDA is significantly better than four high-performing multi-objective algorithms, *i.e.*, NSGA-II, MMOIG, KCA and MOWSA, in terms of the quality of the obtained non-dominated solutions, which verifies the effectiveness of MCEDA for solving the EE_DAPFSP.

Table 9
Statistical results on *C* metric of MCEDA with NSGA-II, MMOIG, KCA and MOWSA for large-scale instances.

$F \times n$	*C*(MCEDA, NSGA-II)	*C*(NSGA-II, MCEDA)	*C*(MCEDA, MMOIG)	*C*(MMOIG, MCEDA)	*C*(MCEDA, KCA)	*C*(KCA, MCEDA)	*C*(MCEDA, MOWSA)	*C*(MOWSA, MCEDA)
4×100	0.97	0.00	0.93	0.01	0.86	0.03	0.81	0.06
4×200	0.94	0.00	0.89	0.00	0.83	0.02	0.77	0.05
4×500	0.93	0.00	0.87	0.01	0.78	0.04	0.73	0.07
6×100	0.98	0.01	0.94	0.02	0.87	0.06	0.82	0.09
6×200	0.95	0.00	0.91	0.01	0.81	0.05	0.79	0.08
6×500	0.92	0.00	0.85	0.03	0.74	0.07	0.65	0.14
8×100	0.94	0.00	0.94	0.00	0.81	0.02	0.78	0.06
8×200	0.92	0.01	0.90	0.02	0.76	0.02	0.72	0.11
8×500	0.93	0.00	0.83	0.01	0.69	0.05	0.63	0.13
Average	0.94	0.00	0.90	0.01	0.79	0.04	0.74	0.09

Table 10
Statistical results for MCEDA, NSGA-II, MMOIG, KCA, and MOWSA on evaluation metrics DI_R, ρ_r, DS, I_H, and I_ε^1 for the small-scale instances.

$F \times n$	NSGA-II					MMOIG					KCA					MOWSA					MCEDA				
	DI_R	ρ_r	DS	I_H	I_ε^1	DI_R	ρ_r	DS	I_H	I_ε^1	DI_R	ρ_r	DS	I_H	I_ε^1	DI_R	ρ_r	DS	I_H	I_ε^1	DI_R	ρ_r	DS	I_H	I_ε^1
2×8	0.033	0.011	1.567	1.216	1.173	0.021	0.041	**1.383**	1.231	1.121	0.017	0.163	2.124	1.247	1.094	0.010	0.251	2.122	1.265	1.062	**0.005**	**0.534**	1.674	**1.332**	**1.029**
2×12	0.041	0.008	1.838	1.177	1.189	0.025	0.022	**1.741**	1.217	1.133	0.035	0.134	2.344	1.229	1.112	0.021	0.225	2.141	1.231	1.077	**0.011**	**0.611**	2.237	**1.293**	**1.034**
2×16	0.062	0.011	2.316	1.163	1.165	0.042	0.017	**2.274**	1.196	1.117	0.024	0.117	2.460	1.213	1.087	0.022	0.203	2.469	1.222	1.053	**0.013**	**0.652**	2.378	**1.274**	**1.023**
2×20	0.044	0.009	2.523	1.144	1.203	0.027	0.015	**2.482**	1.162	1.179	0.032	0.105	2.640	1.176	1.126	0.031	0.194	2.638	1.185	1.095	**0.015**	**0.676**	2.572	**1.315**	**1.045**
2×24	0.031	0.004	2.773	1.105	1.294	0.046	0.012	2.731	1.131	1.232	0.053	0.084	2.784	1.145	1.154	0.053	0.161	2.788	1.153	1.108	**0.021**	**0.738**	**2.675**	**1.219**	**1.054**
3×8	0.055	0.002	2.231	1.159	1.312	0.031	0.025	2.174	1.178	1.245	0.034	0.106	2.347	1.182	1.169	0.031	0.206	2.340	1.204	1.116	**0.011**	**0.661**	**2.148**	**1.229**	**1.062**
3×12	0.051	0.010	2.349	1.231	1.356	0.024	0.019	**2.289**	1.267	1.271	0.038	0.095	2.424	1.283	1.188	0.033	0.244	2.424	1.291	1.127	**0.014**	**0.632**	2.311	**1.324**	**1.076**
3×16	0.072	0.008	2.512	1.243	1.384	0.036	0.038	**2.445**	1.284	1.334	0.045	0.106	2.614	1.295	1.242	0.041	0.223	2.614	1.312	1.155	**0.023**	**0.624**	2.474	**1.337**	**1.085**
3×20	0.067	0.011	2.851	1.126	1.352	0.032	0.021	2.777	1.155	1.286	0.036	0.072	2.927	1.164	1.191	0.032	0.212	2.923	1.185	1.111	**0.012**	**0.683**	**2.738**	**1.275**	**1.064**
3×24	0.075	0.013	2.893	1.114	1.377	0.043	0.062	**2.819**	1.143	1.309	0.067	0.067	2.954	1.151	1.226	0.064	0.102	2.959	1.173	1.141	**0.025**	**0.755**	2.869	**1.257**	**1.089**
4×8	0.034	0.017	2.449	1.163	1.391	0.027	0.077	2.373	1.187	1.314	0.015	0.073	2.543	1.209	1.237	0.013	0.099	2.545	1.216	1.157	**0.005**	**0.734**	**2.356**	**1.294**	**1.094**
4×12	0.056	0.007	2.561	1.156	1.412	0.032	0.054	2.540	1.175	1.322	0.024	0.081	2.677	1.186	1.241	0.021	0.115	2.674	1.194	1.168	**0.014**	**0.742**	**2.482**	**1.244**	**1.106**
4×16	0.094	0.012	2.798	1.145	1.443	0.051	0.045	2.741	1.173	1.344	0.051	0.084	2.816	1.192	1.254	0.046	0.211	2.817	1.215	1.153	**0.022**	**0.647**	**2.692**	**1.287**	**1.098**
4×20	0.083	0.018	3.114	1.053	1.467	0.045	0.057	2.970	1.119	1.364	0.057	0.082	3.247	1.136	1.273	0.051	0.219	3.245	1.142	1.172	**0.027**	**0.623**	**2.934**	**1.216**	**1.109**
4×24	0.076	0.025	3.278	1.129	1.515	0.059	0.039	3.217	1.152	1.387	0.069	0.074	3.372	1.173	1.291	0.063	0.188	3.371	1.183	1.233	**0.042**	**0.673**	**3.147**	**1.279**	**1.117**
Average	0.058	0.011	2.537	1.155	1.336	0.036	0.036	2.464	1.185	1.264	0.040	0.096	2.685	1.199	1.192	0.035	0.190	2.671	1.212	1.129	**0.017**	**0.666**	2.513	**1.278**	**1.072**

Table 11
Statistical results for MCEDA, NSGA-II, MMOIG, KCA, and MOWSA on evaluation metrics DI_R, ρ_r, DS, I_H, and I_ε^1 for the large-scale instances.

$F \times n$	NSGA-II					MMOIG					KCA					MOWSA					MCEDA				
	DI_R	ρ_r	DS	I_H	I_ε^1	DI_R	ρ_r	DS	I_H	I_ε^1	DI_R	ρ_r	DS	I_H	I_ε^1	DI_R	ρ_r	DS	I_H	I_ε^1	DI_R	ρ_r	DS	I_H	I_ε^1
4×100	0.096	0.063	2.821	1.214	1.179	0.075	0.098	2.742	1.234	1.149	0.062	0.119	2.869	1.252	1.115	0.038	0.159	2.651	1.275	1.089	**0.022**	**0.562**	**2.499**	**1.348**	**1.039**
4×200	0.107	0.049	3.007	1.193	1.231	0.086	0.078	2.892	1.218	1.161	0.052	0.101	3.166	1.237	1.122	0.027	0.138	3.177	1.254	1.081	**0.019**	**0.634**	**2.834**	**1.321**	**1.031**
4×500	0.088	0.062	3.185	1.175	1.248	0.061	0.092	**3.109**	1.206	1.188	0.037	0.125	3.351	1.226	1.131	0.028	0.181	3.453	1.243	1.102	**0.017**	**0.541**	3.266	**1.293**	**1.063**
6×100	0.133	0.042	2.877	1.148	1.284	0.094	0.088	**2.858**	1.194	1.218	0.051	0.121	3.233	1.217	1.154	0.032	0.179	3.367	1.236	1.144	**0.028**	**0.569**	3.189	**1.324**	**1.102**
6×200	0.124	0.056	3.293	1.121	1.325	0.117	0.073	3.252	1.174	1.244	0.062	0.112	3.351	1.885	1.168	0.036	0.164	3.447	1.203	1.132	**0.022**	**0.594**	**3.218**	**1.287**	**1.091**
6×500	0.126	0.089	3.445	1.114	1.365	0.118	0.097	3.409	1.159	1.272	0.086	0.149	2.629	1.162	1.172	0.042	0.143	2.486	1.183	1.119	**0.032**	**0.521**	**3.268**	**1.273**	**1.076**
8×100	0.097	0.066	2.969	1.186	1.381	0.075	0.099	2.914	1.182	1.294	0.044	0.145	2.925	1.213	1.179	0.037	0.175	3.252	1.227	1.131	**0.025**	**0.516**	**2.866**	**1.323**	**1.194**
8×200	0.131	0.042	3.123	1.182	1.425	0.112	0.112	3.042	1.221	1.331	0.078	0.155	3.113	1.253	1.213	0.046	0.166	3.419	1.261	1.166	**0.038**	**0.525**	**2.975**	**1.344**	**1.236**
8×500	0.112	0.067	3.286	1.109	1.352	0.144	0.103	3.252	1.153	1.313	0.111	0.161	3.374	1.167	1.185	0.075	0.195	3.643	1.183	1.134	**0.066**	**0.474**	**3.187**	**1.254**	**1.012**
Average	0.115	0.062	3.114	1.162	1.312	0.100	0.095	3.054	1.196	1.243	0.067	0.134	3.114	1.292	1.162	0.042	0.169	3.213	1.232	1.124	**0.032**	**0.551**	**3.036**	**1.309**	**1.096**

Table 12
Statistical results for all compared algorithms for the large-scale instances grouped by F, S, and n.

		NSGA-II					MMOIG					KCA					MOWSA					MCEDA				
		DI_R	ρ_r	DS	I_H	I_ε^1	DI_R	ρ_r	DS	I_H	I_ε^1	DI_R	ρ_r	DS	I_H	I_ε^1	DI_R	ρ_r	DS	I_H	I_ε^1	DI_R	ρ_r	DS	I_H	I_ε^1
F	4	0.105	0.073	2.951	1.143	1.179	0.088	0.051	**2.279**	1.167	1.146	0.069	0.106	3.134	1.184	1.113	0.058	0.148	3.121	1.196	1.096	**0.045**	**0.622**	2.283	**1.324**	**1.035**
	6	0.141	0.062	3.388	1.135	1.141	0.132	0.049	3.241	1.151	1.112	0.112	0.095	3.350	1.178	1.087	0.083	0.129	3.337	1.185	1.054	**0.062**	**0.665**	**3.064**	**1.285**	**1.032**
	8	0.123	0.047	3.129	1.097	1.157	0.115	0.061	3.074	1.132	1.123	0.103	0.076	3.126	1.155	1.105	0.096	0.112	3.104	1.171	1.076	**0.074**	**0.704**	**2.972**	**1.274**	**1.044**
S	30	0.121	0.034	2.931	1.083	1.225	0.108	0.073	**2.761**	1.117	1.188	0.093	0.107	2.855	1.137	1.143	0.077	0.136	2.845	1.157	1.093	**0.056**	**0.649**	3.078	**1.253**	**1.073**
	40	0.129	0.031	2.681	1.112	1.252	0.116	0.089	**2.545**	1.125	1.223	0.095	0.113	2.833	1.141	1.167	0.076	0.171	2.973	1.164	1.102	**0.066**	**0.595**	2.745	**1.263**	**1.084**
	50	0.117	0.038	3.123	1.153	1.312	0.104	0.083	2.953	1.136	1.279	0.099	0.089	3.141	1.164	1.221	0.088	0.178	3.135	1.187	1.152	**0.074**	**0.611**	**2.933**	**1.277**	**1.092**
n	100	0.147	0.023	2.937	1.121	1.293	0.131	0.061	**2.871**	1.128	1.221	0.118	0.102	2.996	1.155	1.186	0.082	0.117	2.983	1.175	1.126	**0.048**	**0.698**	2.886	**1.262**	**1.083**
	200	0.124	0.052	3.117	0.105	1.328	0.108	0.043	3.078	1.122	1.267	0.096	0.116	3.346	1.148	1.224	0.085	0.128	3.337	1.162	1.163	**0.056**	**0.662**	**3.029**	**1.258**	**1.108**
	500	0.153	0.044	3.295	0.938	1.355	0.142	0.064	3.223	1.163	1.323	0.129	0.127	3.551	1.186	1.187	0.105	0.157	3.545	1.193	1.095	**0.083**	**0.609**	**3.218**	**1.294**	**1.041**
Average		0.129	0.045	3.061	0.988	1.249	0.116	0.064	2.892	1.138	1.209	0.102	0.103	3.148	1.161	1.159	0.083	0.142	3.153	1.177	1.106	**0.063**	**0.646**	**2.912**	**1.277**	**1.066**

To ensure adequate comparisons, the statistical results for MCEDA, NSGA-II, MMOIG, KCA and MOWSA on measure metrics DI_R, ρ_r, DS, I_H, and I_ε^1 for small-scale and large-scale instances are given in Tables 10 and 11, respectively. Moreover, the statistical results of MCEDA with NSGA-II, MMOIG, KCA and MOWSA under different problem characteristic groups F, S, and n are reported in Table 12. The best value of each metric for each group in these tables is shown with **boldface**. As can be seen from Tables 10 to 12 that the proposed MCEDA is better than its counterparts on all performance indicators. For the small-scale instances, the results of MCEDA on the reference distance metric (DI_R) are zero or close to zero for almost all testing instances and their have a significant advantage both in hypervolume (I_H) and unary epsilon (I_ε^1) indicators, which indicates that the Pareto front consisting of the promising non-dominated set obtained by MCEDA is very close to the true Pareto front. For the large-scale test instances, the values of MCEDA on DI_R and I_ε^1 are also significantly smaller than that of other counterparts, which implies these algorithms have few contributions to the reference Pareto fronts and our proposed MCEDA is very preferred for solving the considered problem. Furthermore, it is clear in Table 12 that average values of the non-dominance ratio (ρ_r) obtained by MCEDA for addressing different size instances is 0.646, which is much better than those of NSGA-II (0.045), MMOIG (0.064), KCA (0.103), and MOWSA (0.142). Additionally, it can be seen from these tables that the non-dominance ratio (ρ_r) is clearer than hypervolume (I_H) and unary epsilon (I_ε^1), and the proposed MCEDA performs very stable on different scale instances, which indicates that MCEDA has good robustness and stability. As seen from all comparisons on the distribution spacing metric DS in Tables 10 to 12, compared with NSGA-II, MMOIG, KCA and MOWSA, the DS values obtained by MCEDA for solving both the small-scale instances and large-scale instances are smaller than four compared algorithms, which means the dispersion of non-dominated solutions in Pareto archive obtained by MCEDA is better. Thus, the proposed MCEDA can obtain more different high-quality feasible scheduling solutions, which can provide decision makers to choose the appropriate scheduling schemes according to practical preferences.

To further check whether the observed differences in Tables 10-12 are indeed statistically meaningful, the statistical multifactor analysis of variance (ANOVA) experiments with a confidence level of 95% are also carried out to analyze these numerical results. The main effect of each factor is investigated where the type of algorithm is considered as a factor and two measure metrics DI_R and ρ_r are employed as the response variables, respectively. The ANOVA is a powerful parametric statistical technique which is widely used to detect statistical significance (Pan & Ruiz, 2012). Notice that the three main hypotheses (*i.e.*, normality, homogeneity of variance and independence of residuals) are also checked in all computational comparisons. The checked results indicate that no significant deviations are found in the fulfillment of these hypotheses. Moreover, the p-value is an important indicator to determine whether there is a significant difference among all factors. If p-value is smaller than 0.05, the difference between algorithms is significant. It is remarkable that the overlapping intervals indicate that there is insignificant difference between their mean performance. The means plots and 95% Tukey HSD confidence intervals for interactions between the type of algorithms and the number of factories for NSGA-II, MMOIG, KCA, MOWSA and MCEDA under four metrics DI_R, ρ_r, I_H and I_ε^1 are depicted in Fig. 12, respectively. It clearly reveals from Fig. 12 that MCEDA is significantly superior to other four algorithms in solving different groups of instances in terms of DI_R, ρ_r, I_H and I_ε^1 respectively. Additionally, three competitive algorithms, *i.e.*, MMOIG, KCA and MOWSA, outperform the traditional NSGA-II, while no statistical significance can be detected among the MMOIG, KCA and MOWSA. Therefore, it can be safely concluded that the proposed MCEDA is an effective and efficient algorithm for the EE_DAPFSP.

For the purpose of visualizing the performance of all compared algorithms, Fig. 13 illustrates the Pareto front distribution graphs of all

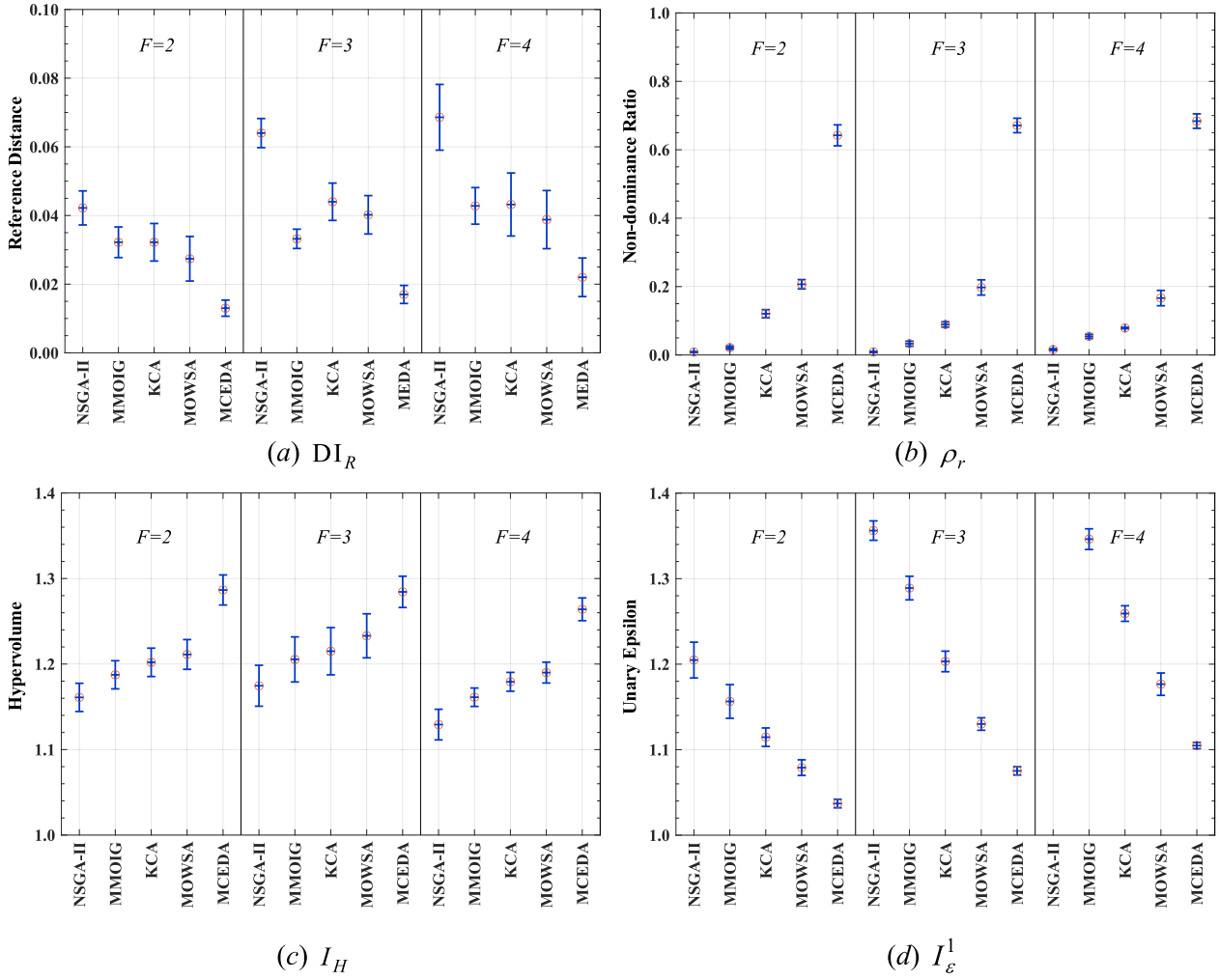


Fig. 12. The means plots and 95% Tukey HSD confidence intervals for the interaction between the type of algorithm and the number of factories for NSGA-II, MMOIG, KCA, MOWSA and MCEDA.

non-dominated sets obtained by NSGA-II, MMOIG, KCA, MOWSA and MCEDA for solving the nine typical instances with respect to different sizes (*i.e.*, small, medium and large scale), respectively. It can be clearly seen from Fig. 13 that the non-dominated solutions obtained by MCEDA almost dominate the other solutions yielded by its counterparts regardless of both the small-scale instances and the large-scale instances, which indicates that the superiority of the proposed MCEDA is obvious. The main reason is that MCEDA has a powerful search engine to drive both global exploration and local exploitation. Furthermore, the distribution of the union Pareto fronts found by MCEDA is more decentralized and diversified, and the quality of the non-dominated solutions yielded by MCEDA is relatively high, which can provide a variety of satisfactory scheduling schemes for decision makers and can achieve a reasonable compromise between the maximum completion time and total energy consumption. In conclusion, the MCEDA proposed in this paper can effectively and efficiently solve the energy-efficient DAPFSP.

5. Conclusions

Energy saving has become a hot issue of global concern. Energy-efficient production scheduling problem is one of the most fundamental and difficult scheduling problems encountered in many kinds of real-life manufacturing industries. This paper considered the energy-efficient distributed assembly permutation flow-shop scheduling problem (EE_DAPFSP), whose objectives are to minimize the maximum completion time and the total carbon emission at the same time. To deal with this strongly NP-hard problem, a novel matrix-cube-based distribution estimation algorithm (MCEDA) was proposed. Based on the characteristics of the EE_DAPFSP, the hybrid initialization strategy, the more guided global search, the deep local search, and the speed adjustment strategies were designed, respectively. The effectiveness of these strategies was analyzed. Extensive computational experiments reveal that our MCEDA statically outperforms several state-of-the-art algorithms. To the best of our knowledge, this is the first report to propose an EDA-based algorithm for the energy-saving production scheduling problem.

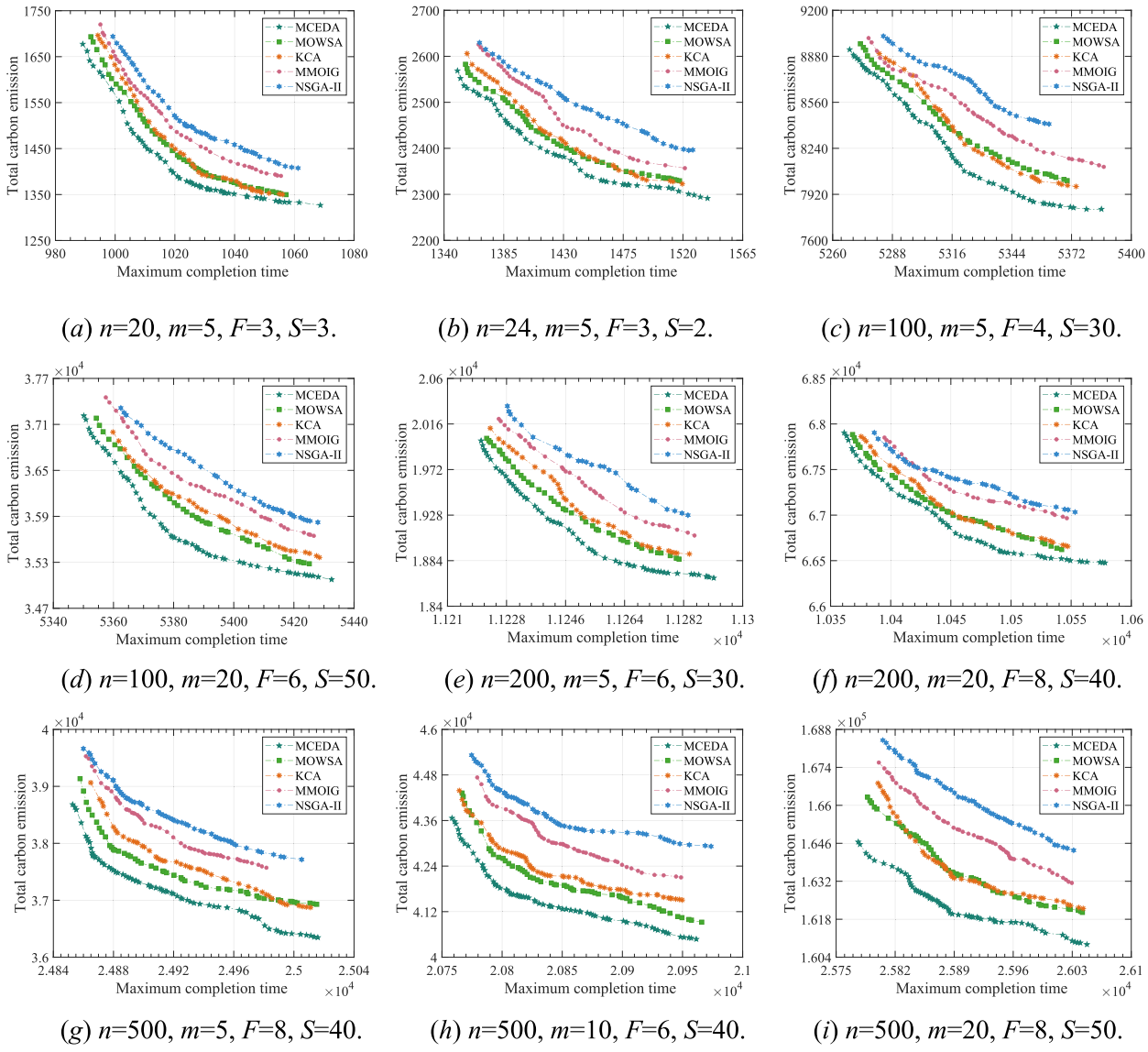


Fig. 13. The Pareto front distribution of non-dominated solutions obtained by NSGA-II, MMOIG, KCA, MOWSA and MCEDA.

There are mainly two important directions for future research. First, we would like to develop several knowledge-based stargates to further enhance the guidance ability of MCEDA's global search. Second, it would be meaningful to extend the proposed MCEDA to the dynamical DAPFSP as well as the distributed production and transportation integrated scheduling problems.

CRediT authorship contribution statement

Zi-Qi Zhang: Investigation, Methodology, Software, Writing – original draft. **Rong Hu:** Methodology, Funding acquisition, Supervision, Writing – review & editing. **Bin Qian:** Methodology, Funding acquisition, Investigation, Writing – review & editing. **Huai-Ping Jin:** . **Ling Wang:** Supervision, Project administration. **Jian-Bo Yang:** 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.

Acknowledgements

This research is partially supported by the National Natural Science Foundation of China (62173169, 61963022, 61873328), , and the Basic Research Key Project of Yunnan Province (202101AS070097).

References

Abedi, M., Chiong, R., Noman, N., & Zhang, R. (2020). A multi-population, multi-objective memetic algorithm for energy-efficient job-shop scheduling with deteriorating machines. *Expert Systems with Applications, 157*, 113348.

Chen, J. F., Wang, L., & Peng, Z. P. (2019). A collaborative optimization algorithm for energy-efficient multi-objective distributed no-idle flow-shop scheduling. *Swarm and Evolutionary Computation, 50*, 100557.

Ciavotta, M., Minella, G., & Ruiz, R. (2013). Multi-objective sequence dependent setup times permutation flowshop: A new algorithm and a comprehensive study. *European Journal of Operational Research, 227*, 301–313.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*, 182–197.

Ding, J. Y., Song, S. J., & Wu, C. (2016). Carbon-efficient scheduling of flow shops by multi-objective optimization. *European Journal of Operational Research, 248*, 758–771.

Hatami, S., Ruiz, R., & Andres-Romano, C. (2013). The Distributed Assembly Permutation Flowshop Scheduling Problem. *International Journal of Production Research, 51*, 5292–5308.

Ishibuchi, H., Yoshida, T., & Murata, T. (2003). Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation, 7*, 204–223.

Jarboui, B., Eddaly, M., & Siarry, P. (2009). An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems. *Computers & Operations Research, 36*, 2638–2646.

Jiang, S. L., & Zhang, L. (2019). Energy-oriented Scheduling for Hybrid Flow Shop With Limited Buffers Through Efficient Multi-Objective Optimization. *IEEE Access, 7*, 34477–34487.

Larrañaga, P., & Lozano, J. A. (2001). *Estimation of distribution algorithms: A new tool for evolutionary computation*. Springer Science & Business Media.

Lin, J., Wang, Z. J., & Li, X. D. (2017). A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem. *Swarm and Evolutionary Computation, 36*, 124–135.

Lin, J., & Zhang, S. (2016). An effective hybrid biogeography-based optimization algorithm for the distributed assembly permutation flow-shop scheduling problem. *Computers & Industrial Engineering, 97*, 128–136.

Lu, C., Gao, L., Li, X. Y., Pan, Q. K., & Wang, Q. (2017). Energy-efficient permutation flow shop scheduling problem using a hybrid multi-objective backtracking search algorithm. *Journal of Cleaner Production, 144*, 228–238.

May, G., Stahl, B., Taisch, M., & Prabhu, V. (2015). Multi-objective genetic algorithm for energy-efficient job shop scheduling. *International Journal of Production Research, 53*, 7071–7089.

Minella, G., Ruiz, R., & Ciavotta, M. (2008). A Review and Evaluation of Multiobjective Algorithms for the Flowshop Scheduling Problem. *Informs Journal on Computing, 20*, 451–471.

Montgomery, D. C. (2008). *Design and Analysis of Experiments* (Second ed.). United States: John Wiley & Sons, United States.

Pan, Q. K., Gao, L., Li, X. Y., & Jose, F. M. (2019). Effective constructive heuristics and meta-heuristics for the distributed assembly permutation flowshop scheduling problem. *Applied Soft Computing, 81*, 105492.

Pan, Q. K., & Ruiz, R. (2012). An estimation of distribution algorithm for lot-streaming flow shop problems with setup times. *Omega-International Journal of Management Science, 40*, 166–180.

Ruiz, R., & Stutzle, T. (2008). An Iterated Greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research, 187*, 1143–1159.

Sang, H. Y., Pan, Q. K., Li, J. Q., Wang, P., Han, Y. Y., Gao, K. Z., & Duan, P. (2019). Effective invasive weed optimization algorithms for distributed assembly permutation flowshop problem with total flowtime criterion. *Swarm and Evolutionary Computation, 44*, 64–73.

Shao, Z., Pi, D., & Shao, W. (2018). A multi-objective discrete invasive weed optimization for multi-objective blocking flow-shop scheduling problem. *Expert Systems with Applications, 113*, 77–99.

Tiwari, A., Chang, P.-C., Tiwari, M. K., & Kollanoor, N. J. (2014). A Pareto block-based estimation and distribution algorithm for multi-objective permutation flow shop scheduling problem. *International Journal of Production Research, 53*, 793–834.

Wang, G., Gao, L., Li, X., Li, P., & Tasgetiren, M. F. (2020). Energy-efficient distributed permutation flow shop scheduling problem using a multi-objective whale swarm algorithm. *Swarm and Evolutionary Computation, 57*, 100716.

Wang, J. J., & Wang, L. (2020). A Knowledge-Based Cooperative Algorithm for Energy-Efficient Scheduling of Distributed Flow-Shop. *IEEE Transactions on Systems, Man, Cybernetics: Systems, 50*, 1805–1819.

Wang, L., Wang, S. Y., Xu, Y., Zhou, G., & Liu, M. (2012). A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem. *Computers & Industrial Engineering, 62*, 917–926.

Wang, S. Y., & Wang, L. (2016). An Estimation of Distribution Algorithm-Based Memetic Algorithm for the Distributed Assembly Permutation Flow-Shop Scheduling Problem. *IEEE Transactions on Systems, Man, Cybernetics: Systems, 46*, 139–149.

Zhang, Z. Q., Qian, B., Hu, R., Jin, H. P., & Wang, L. (2021). A matrix-cube-based estimation of distribution algorithm for the distributed assembly permutation flow-shop scheduling problem. *Swarm and Evolutionary Computation, 60*, 100785.

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation, 3*, 257–271.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., & Fonseca, V. G.d. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation, 7*, 117–132.