Expert Systems With Applications 205 (2022) 117602

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 blocking flow-shop scheduling problem with sequence-dependent setup times

Zi-Qi Zhang [a,b,c], Bin Qian [a,b,c,*], Rong Hu [a,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, PR China
[b] School of Mechanical and Electrical Engineering, Kunming University of Science and Technology, Kunming 650500, PR China
[c] Yunnan Key Laboratory of Artificial Intelligence, Kunming University of Science and Technology, Kunming 650500, PR China
[d] Department of Automation, Tsinghua University, Beijing 100084, PR China
[e] Alliance Manchester Business School, The University of Manchester, Manchester M15 6PB, United Kingdom

ARTICLE INFO

Keywords:
Blocking flow-shop scheduling
Estimation of distribution algorithm
Setup times
Multi-dimensional probabilistic model
Iterated local search

ABSTRACT

The blocking flow-shop scheduling problem with sequence-dependent setup times (BFSP_SDST) is a strong NP-hard problem that exists widely in practice. However, research on this issue is still quite limited. Hence, this paper presents a novel matrix-cube-based estimation of distribution algorithm (MCEDA) to minimize the makespan criterion of the BFSP_SDST. In MCEDA's global search, a matrix cube is devised to reasonably learn the promising patterns in excellent solutions or individuals, and then a matrix-cube-based probabilistic model is developed to quickly guide global search toward the potential promising regions in solution space. A diversity controlling mechanism is also added to avoid the stagnation of global search. In MCEDA's local search, an iterated multi-neighborhood local search controlled by the probabilistic model in global search is designed to execute deeper exploitation from those promising regions. Additionally, two constructive heuristics for generating high-quality initial individuals and one fast *Insert*-based neighbor evaluation method for accelerating the efficiency of local search are presented based on an analysis of the problem's features. MCEDA's efficacy and superiority in solving the BFSP SDST are demonstrated through comprehensive comparisons with 22 state-of-the-art algorithms.

1. Introduction

Production scheduling has been recognized as a realistic and reliable decision-making approach for allocating restricted resources within a certain time period in order to achieve one or more decision-maker-defined objectives (Pinedo, 2015). As a hot research topic in the field of production scheduling, the flow-shop scheduling problem (FSP) has a wide range of applications in numerous manufacturing systems, production and assembly lines, and information service facilities. For the typical FSP, it is commonly assumed that there are infinitely storage facilities or buffer units between any two adjacent machines, where finished jobs can be stored in these buffer units for an unlimited amount of time. However, in many real-world manufacturing situations, due to production characteristics and technical constraints, there are usually no intermediate storage units between machines (Grabowski & Pempera, 2007). In this sense, the traditional FSP is converted into the blocking FSP (BFSP), which is a typical NP-hard problem in the strong sense (Hall and Sriskandarajah, 1996; Ronconi & Henriques, 2009; Wang, et al., 2010). As a significant subfield of FSP, BFSP has attracted the considerable attention and interest from both researchers and practitioners in recent decades. A wide variety of real-world industrial processes and manufacturing systems can be modeled as the BFSP, such as chemical and pharmaceutical manufacturing (Ronconi, 2004), iron and steel manufacturing (Gong, et al., 2010), robotic cells (Elmi & Topaloglu, 2013), serial manufacturing processes (Koren, et al., 2017), and waste treatment (Riahi, et al., 2017). Nowadays, the BFSP has garnered the tremendous attention and interest of both researchers and practitioners (see Section 2).

Setup time is prevalent in a variety of real-life manufacturing systems. In many factories, setup time is frequently derived from non-productive activities such as cleaning devices, adjusting equipment, switching machines, repairing or releasing jobs, especially in chemical

* Corresponding author.
E-mail addresses: zhangziqi@kust.edu.cn (Z.-Q. Zhang), bin.qian@vip.163.com (B. Qian), wangling@tsinghua.edu.cn (L. Wang), jian-bo.yang@umist.ac.uk (J.-B. Yang).

https://doi.org/10.1016/j.eswa.2022.117602
Received 6 March 2022; Received in revised form 4 May 2022; Accepted 14 May 2022
Available online 21 May 2022

or pharmaceutical plants. Although in almost all the existing research works related to BFSPs, it is usually assumed that the setup time is negligible or included in processing time, however, substantial setup times should be separable (Shao, et al., 2018b). Nevertheless, the improper handling of setup operations may result in the consumption of more than 20% of the available machine capacity (Pinedo, 2015). To the best of our knowledge, there are still very few works on BFSP that involve setup time, especially for sequence-dependent setup times (SDST) (Shao, et al., 2018b). Therefore, this paper investigates an extension of the BFSP, namely the BFSP with SDST (BFSP_SDST), whose criterion is to minimize makespan (*i.e.*, C_{max}). The SDST indicates that the setup time of each job on each machine depends not only on the job itself but also on its immediately preceding job. According to the widely used three-field notation $\alpha|\beta|\gamma$ proposed by Graham, et al. (1979), the BFSP under the makespan criterion and the studied problem herein can be denoted as $Fm|blocking|C_{max}$ and $Fm|blocking, ST_{sd}|C_{max}$, respectively. Since $Fm|blocking|C_{max}$ is already recognized as strongly NP-hard, and it is obviously reduced to $Fm|blocking, ST_{sd}|C_{max}$, it can be concluded that $Fm|blocking, ST_{sd}|C_{max}$ is also NP-hard in the strong sense.

For the NP-hard scheduling problems, existing mathematical algorithms are often of limited use due to their excessive computation time or poor performance under reasonable runtime. Hence, numerous hybrid intelligent optimization algorithms (HIOAs) have been developed to tackle this issue, aiming to achieve satisfactory solutions for a wide variety of traditional scheduling problems within several seconds or tens of seconds. Among these algorithms, the hybrid estimation of distribution algorithm (HEDA) is a unique one. Unlike the crossover and mutation operators in most existing HIOAs (*e.g.*, hybrid genetic algorithm, hybrid particle swarm optimization algorithm, hybrid differential evolution algorithm), HEDA generates the offspring population by sampling an EDA-based probability model, which can learn and accumulate valuable information about excellent individuals from a macro perspective, as well as establish explicit probability models to effectively estimate the distribution of superior solutions and to predict promising regions in the feasible solution space. To a certain extent, such novel population generation mechanism can avoid the destruction of the blocks (*i.e.*, the partial ordered patterns) in excellent individuals or solutions to a certain extent (Larranga & Lozano, 2001). Due to its stronger global exploration, simpler framework, and faster convergence speed, HEDA has been widely utilized to solve various scheduling problems (Faraji Amiri & Behnamian, 2020; Jarboui, et al., 2009; Pan & Ruiz, 2012; Qian, et al., 2017; Wang, et al., 2014; Wang, et al., 2013; Wu, et al., 2021). These successful applications have indicated that HEDA has considerable competitive advantage against other algorithms. Therefore, HEDA is selected as the main framework of our proposed algorithm for $Fm|blocking, ST_{sd}|C_{max}$.

Unfortunately, the majority of currently available HEDAs have two drawbacks. The first drawback is that most existing HEDAs commonly use one or more two-dimensional probabilistic models or matrices to learn the characteristic information of excellent individuals. The structure of two-dimensional matrix directly determines that only the matrix's elements and the subscripts of these elements can be utilized to store information. For the two-dimensional matrix $\boldsymbol{M}_{n\times n}$, its element $M_{n\times n}(x,y)$ is used to record the occurrence frequency of the block $[x,y]$ in excellent individuals, while the subscript (x,y) is only enough to save the information of one block's structure or pattern. There is no extra space to record the position of this block $[x,y]$ in each corresponding excellent individual. This makes it difficult for two-dimensional probabilistic models to correctly guide the search direction, so that the practical performance of the existing HEDAs is relatively limited (see Subsection 4.2.1). The second drawback is that almost all existing HEDAs and other HIOAs lack substantive interaction between their global and local searches. In each of these algorithms, the local search can only execute the neighborhood exploitation by using a very limited number of pre-defined common neighborhood operators (*e.g.*, *Insert*, *Swap*, and *Interchange*). The lack of global exploration information to assist the local search undoubtedly limits the depth of the local search, resulting in the algorithm's overall practical performance being constrained. To overcome the aforementioned defects, a novel matrix-cube-based HEDA, namely MCEDA, is proposed to address the considered problem.

The main characteristics of our MCEDA are summarized as follows.

- A three-dimensional matrix (*i.e.*, matrix cube) is devised to reasonably record and reserve the valuable patterns in excellent individuals or solutions. For a three-dimensional matrix, the z in its subscript (x, y, z) is used to record the position of job block $[x,y]$ in the corresponding excellent solutions. Meanwhile, a matrix-cube-based probabilistic model with a sampling strategy is developed to estimate the distribution of excellent solutions in solution space and correctly guide global search to promising regions. Moreover, a simple diversity controlling mechanism is designed to avoid the stagnation of global search.
- Different from most existing HIOA's local searches that execute local search independently, a new iterated multi-neighborhood local search controlled by the matrix-cube-based probabilistic model in global search is presented to undertake deeper exploitation from those promising regions. This novel local search utilizes the block patterns saved in the probability model to approximately evaluate neighbors and dynamically create promising neighborhoods for performing fast and rich search.
- Based on the problem's characteristics, two effective constructive heuristics are designed to ensure the quality and diversity of the initial population. Meanwhile, a fast *Insert*-based neighbor evaluation method is presented to improve search efficiency.
- The proposed MCEDA is compared against twenty-two state-of-the-art algorithms on different instances. The statistical results demonstrate the efficacy and superiority of MCEDA.

The remainder of this paper is organized as follows. Section 2 briefly reviews the related literature. Section 3 describes the model of the problem. Section 4 presents MCEDA after explaining two effective heuristics for initialization, the matrix cube based global search, and the probabilistic model controlled local search. The comparison results and statistical analysis are provided in Section 5. Finally, Section 6 gives some concluding remarks and suggestions for future research.

2. Literature review of BFSP and BFSP_SDST

The comprehensive review of the BFSP can be found in (Miyata & Nagano, 2019). Since 2010, there have been mainly three types of algorithms for the BFSP.

The first is the HIOA. The existing studies have mainly concentrated on minimizing the makespan criterion. Wang, et al. (2010) presented a hybrid discrete differential evolution (HDDE), in which a speedup method was utilized to evaluate the *Insert*-based neighbor solutions. The test results showed that HDDE outperformed the famous tabu search (TS) algorithm (Grabowski & Pempera, 2007). Wang, et al. (2011) developed a hybrid modified global-best harmony search (hmgHS), which performed better than HGA (Wang, et al., 2011) and TS (Grabowski & Pempera, 2007). Wang, et al. (2012) devised a three-phase algorithm (TPA), in which a priority rule, a NEH's variant, and a modified simulated annealing are utilized in three phases, respectively. The comparative results demonstrated that TPA was relatively more efficient than HDDE (Wang, et al., 2010). Lin and Ying (2013) proposed a revised artificial immune system (RAIS) algorithm, where a simple iterated greedy algorithm (IGA) was embedded to intensively exploit around the better solutions. The test results indicated that the RAIS was superior to both HDDE (Wang, et al., 2010) and IGA (Ribas, et al., 2011). Han, et al. (2015) designed a discrete artificial bee colony algorithm incorporating differential evolution (DE_ABC). The test results demonstrated that DE_ABC was superior to the compared algorithms.

Tasgetiren, et al. (2015) presented a populated local search with differential evolution (DE_PLS). The test results showed that DE_PLS performed better than some of the best performing algorithms from the literature. Han, et al. (2016) introduced a modified fruit fly optimization (MFFO) algorithm, in which a problem-specific heuristic, a neighborhood strategy, and a speedup insert-neighborhood based local search are employed. Shao, et al. (2018a) developed an EDA with a path relinking technique (P-EDA). The path relinking technique here is utilized to avoid performing the blind search. The results compared with various other high-performing algorithms verified the effectiveness of P-EDA. Shao, et al. (2019) proposed a discrete invasive weed optimization (DIWO), in which a random-insertion-based spatial dispersal, a shuffle-based referenced local search, and an improved competitive exclusion are devised. The results demonstrated that DIWO outperformed the compared algorithms. Besides the makespan minimization, Ribas, et al. (2015) devised an effective discrete artificial bee colony algorithm (DABC_RCT) for minimizing the total flowtime criterion. Shao, et al. (2017) presented a self-adaptive discrete invasive weed optimization (SaDIWO), and Nagano, et al. (2017) designed an evolutionary clustering search (ECS) algorithm to minimize the total tardiness criterion.

The second is the constructive heuristic. As for the makespan minimization, Pan and Wang (2012) introduced six effective heuristics, namely PF-NEH(x), wPF-NEH(x), PW-NEH(x), PF-NEH$_{LS}$(x), wPF-NEH$_{LS}$(x), and PW-NEH$_{LS}$(x), in which the variable x is employed to control the number of sequences generated. The test results demonstrated that PW-NEH$_{LS}$(5) beat NEH (Nawaz, et al., 1983), MME (Ronconi, 2004), and PFE (Ronconi, 2004). As for the total flowtime minimization, Tasgetiren, et al. (2016) developed a variable block insertion heuristic (VBIH), and Fernandez-Viagas, et al. (2016) proposed an effective beam-search- based heuristic (BSH).

The third is the iterated greedy algorithm (IGA). As for the makespan minimization, Tasgetiren, et al. (2017) devised two enhanced IGAs, *i.e.*, iterated greedy with jumping probability (IG_IJ) and iterated greedy with RIS local search (IG_RIS), which combined an effective constructive heuristic and employed two speedup methods for the insert-based and swap-based neighborhood searches, respectively. Extensive experimental results demonstrated that the devised algorithms achieved better results than most state-of-the-art algorithms. Ribas, et al. (2013) proposed two competitive variable neighborhood search methods (namely SVNS_S and SVNS_D). The experimental results revealed that SVNS outperformed both HDDE (Wang, et al., 2010) and IGA (Ribas, et al., 2011). Moslehi and Khorasanian (2014) developed a hybrid variable neighborhood search (HVNS), whose performance surpassed several state-of-the-art algorithms. As for the total flowtime minimization, Khorasanian and Moslehi (2012) presented an IGA, in which a modified NEH was employed to generate the initial solution. Ding, et al. (2016) investigated several properties of the BFSP and presented an IGA based on these problem-specific properties.

Recently, several researchers studied the BFSP_SDST and the other BFSP's variants, and they all adopted HIOA to address the corresponding problems. Shao, et al. (2018b) devised a novel discrete water wave optimization (DWWO) to minimize the makespan of the BFSP_SDST. In DWWO, a path relinking technique and a variable neighborhood search (VNS) are employed to further improve the algorithm's performance. The test results indicated that DWWO defeated several highly effective algorithms. Nouri and Ladhari (2018) introduced a multi-objective genetic algorithm (MBGA) for the BFSP that considers minimizing both the makespan and the total completion time. Gong, et al. (2018) developed a hybrid artificial bee colony (HABC) to minimize the makespan and the earliness of the lot-streaming BFSP. Han, et al. (2019) designed a robust multi-objective evolutionary algorithm to minimize the makespan, the tardiness and the robustness of the lot-streaming BFSP with machine breakdowns. Han, et al. (2020) presented an effective multi-objective discrete evolutionary optimization (MDEO) to minimize the makespan and the energy consumption of the energy-efficient BFSP_SDST. Ribas, et al. (2021) proposed an enhanced IGA to minimize the makespan of the parallel BFSP_SDST. Shao, et al. (2021) devised an effective constructive heuristic and an IGA to minimize the makespan of the distributed mixed BFSP. Zhao, et al. (2022) developed an effective water wave optimization to minimize the total tardiness of the distributed assembly BFSP.

From the above literature, it can be seen that although researchers have undertaken much research on the BFSP and its variants, only Shao, et al. (2018b) considers the SDST in the BFSP. Thus, it is necessary and meaningful to consider such a significant problem.

Table 1
Notations applied in the model of BFSP_SDST.

Parameters	
n	The total number of jobs.
m	The total number of machines.
J	The set of jobs, *i.e.*, $J = \{J_1, J_2, ..., J_n\}$.
M	The set of machines, *i.e.*, $M = \{M_1, M_2, ..., M_m\}$.
Indices	
i	The index of jobs ($i = 1, 2, ..., n$).
j	The index of machines ($j = 1, 2, ..., m$).
Variables	
$\boldsymbol{\pi}$	The processing order of jobs, *i.e.*, $\boldsymbol{\pi} = [\pi(1), \pi(2), ..., \pi(n)]$. $\pi(0)$ is a dummy job.
Π	The set of all feasible schedules.
$O_{i,j}$	The operation corresponding to processing of job J_i on machine M_j.
$p_{\pi(i),j}$	The processing time of operation $O_{\pi(i),j}$.
$S_{\pi(i-1),\pi(i),j}$	The setup time between two consecutive jobs $\pi(i-1)$ and $\pi(i)$ on machine M_j. $S_{\pi(0),\pi(i),j}$ is the initial setup time of job $\pi(i)$.
$d_{\pi(i),j}$	The departure time of job $\pi(i)$ on machine M_j. $d_{\pi(i),0}$ is the start time of job $\pi(i)$ on machine M_1.
$f_{\pi(i),j}$	The duration time between the starting time of $\pi(i)$ on M_j and the starting time of the last job on the same machine.
$C_{\max}(\boldsymbol{\pi})$	The makespan of a sequence or schedule $\boldsymbol{\pi}$.

3. Problem statement

3.1. permutation-based model

The BFSP_SDST can be briefly described as follows. There are n jobs and m machines in a flow shop without intermediate buffers. Each job $J_i \in J$ has a sequence of operations $\{O_{i,1}, O_{i,2}, ..., O_{i,m}\}$ to be processed sequentially on machine M_1, M_2, and so on until machine M_m. The operation $O_{i,j}$ of job J_i should be executed on machine M_j with a period of processing time $p_{i,j}$. Since there are no buffers between consecutive machines and each machine has to take some time to prepare before processing, jobs that have completed all operations must remain on the current machine if the downstream machine is not free or not prepared for processing. Setup times are considered sequence-dependent and separable from processing times. In addition, the following assumptions must be met:

- The processing time of each job on each machine is a positive integer and predetermined.
- The release time and transportation time of all jobs are negligible. All jobs are independent and available from zero onwards.
- At any time, each job can be processed on at most one machine, and each machine can only process at most one job.
- Preemption is not permitted. Each job is processed without interruption on each machine.

The related notations are provided in Table 1. According to the above description, the permutation-based model of BFSP_SDST can be established as follows.

$$d_{\pi(0),j} = 0,\ j = 1, 2, ..., m, \tag{1}$$

$$d_{\pi(i),0} = d_{\pi(i-1),1} + S_{\pi(i-1),\pi(i),1},\ i = 1, 2, ..., n, \tag{2}$$

Table 2
Processing and setup times of an example of BFSP_SDST.

	Process time			Sequence-dependent setup time								
				Machine 1			Machine 2			Machine 3		
	M_1	M_2	M_3	$\pi(1)$	$\pi(2)$	$\pi(3)$	$\pi(1)$	$\pi(2)$	$\pi(3)$	$\pi(1)$	$\pi(2)$	$\pi(3)$
$\pi(0)$	0	0	0	2	1	4	1	2	3	2	3	3
$\pi(1)$	1	3	2	0	1	2	0	3	1	0	1	4
$\pi(2)$	1	2	1	3	0	4	4	0	2	1	0	1
$\pi(3)$	3	2	1	2	2	0	3	4	0	2	3	0

$$d_{\pi(i),j} = \max\left\{d_{\pi(i),j-1} + p_{\pi(i),j}, d_{\pi(i-1),j+1} + S_{\pi(i-1),\pi(i),j+1}\right\}, \quad i = 1,2,...,n;\ j = 1,2,...,m-1, \tag{3}$$

$$d_{\pi(i),m} = d_{\pi(i),m-1} + p_{\pi(i),m},\ \ i = 1,2,...,n, \tag{4}$$

$$C_{\max}(\boldsymbol{\pi}) = d_{\pi(n),m}. \tag{5}$$

Eq. (1) determines the departure time of the dummy job $\pi(0)$ on all machines. Eq. (2) calculates the starting time of each job on the first machine. Eq. (3) calculates the departure time of each job on all machines except the last machine. Eq. (4) calculates the departure time or completion time of each job on the last machine. Eq. (5) is the maximum completion time (*i.e.*, makespan) of $\boldsymbol{\pi}$. The aim of $Fm|blocking, ST_{sd}|C_{max}$ is to find an optimal solution π^* in the schedule set Π such that.

$$\pi^* = \underset{\pi\in\Pi}{\operatorname{argmin}}\{C_{\max}(\pi)\}. \tag{6}$$

In contrast to mathematical models of FSPs, the permutation-based models have no explicitly expressed constraints. For the permutation-based model of BFSP_SDST, the constraints are implicit in Eqs. (1)-(4), which stipulate that each job $\pi(i)$ on the current machine M_j can only depart to the next machine M_{j+1} under three conditions: (1) the operation $O_{\pi(i),j}$ has been completed, (2) the job $\pi(i+1)$ has already departed machine M_{j+1}, and (3) the setup operation between $\pi(i)$ and $\pi(i+1)$ on machine M_{j+1} has been completed. When a solution $\boldsymbol{\pi}$ contains different jobs, the departure times of these jobs on each machine are determined by using Eqs. (1)-(4) and no constraints are violated. Thus, a solution $\boldsymbol{\pi}$ is feasible if and only if all jobs in that solution are different from each other.

In the following proposed MCEDA, both the new population sampling strategy in global search (see Section 4.2) and the new neighbor generation methods in local search (see Section 4.3) can ensure that the jobs in each solution $\boldsymbol{\pi}$ (*i.e.*, individual or neighbor) are different from each other. That is to say, any solution $\boldsymbol{\pi}$ obtained in MCEDA is always feasible. Indeed, the most of existing HIOAs optimize the variables of the permutation-based models since researchers using such models can concentrate on the design of the algorithm without considering complex constraint handling.

3.2. permutation-based model using backward calculation

According to the computational reversibility of the permutation-based model of FSPs (Tasgetiren, et al., 2017; Wang, et al., 2010), the makespan (*i.e.* $C_{\max}(\boldsymbol{\pi})$) of the solution $\boldsymbol{\pi}$ can be computed by traversing the solution $\boldsymbol{\pi}$ in reverse order. The backward calculation is described as follows:

$$f_{\pi(n),m+1} = 0, \tag{7}$$

$$f_{\pi(n),j} = f_{\pi(n),j+1} + p_{\pi(n),j},\ \ j = m, m-1, ..., 2, \tag{8}$$

$$f_{\pi(i),m+1} = f_{\pi(i+1),m} + S_{\pi(i),\pi(i+1),m}, \quad i = n-1, ..., 1, 0, \tag{9}$$

$$f_{\pi(i),j} = \max\left\{f_{\pi(i),j+1} + p_{\pi(i),j}, f_{\pi(i+1),j-1} + S_{\pi(i),\pi(i+1),j}\right\}, \quad i = n-1, ..., 2, 0;\ j = m, m-1, ..., 2, \tag{10}$$

$$f_{\pi(i),1} = f_{\pi(i),2} + p_{\pi(i),1},\ \ i = n, n-1, ..., 0, \tag{11}$$

$$C_{\max}(\boldsymbol{\pi}) = f_{\pi(0),1}. \tag{12}$$

So, $C_{\max}(\boldsymbol{\pi})$ can be calculated not only forward via Eqs. (1)-(5) but also backward via Eqs. (7)-(12) with complexity of $O(nm)$. With Eqs. (7)-(12) and Eq. (6), another form of the permutation model of $Fm|blocking, ST_{sd}|C_{max}$ can be established.

Furthermore, for $1 \leqslant i \leqslant n-1$, it has.

$$C_{\max}(\boldsymbol{\pi}) = \max_{j=1,2,...,m}\left\{d_{\pi(i),j} + S_{\pi(i),\pi(i+1),j} + f_{\pi(i+1),j}\right\}. \tag{13}$$

With Eq. (13), a fast neighbor evaluation method (see Subsection 4.3.1) can be devised to calculate the objective functions of the solutions in the insertion neighborhood for $Fm|blocking, ST_{sd}|C_{max}$. This fast neighbor evaluation method is adopted to speed up the efficiency in MCEDA's local search.

3.3. Small numerical example of the forward and backward calculations

To illustrate the forward and backward calculations, a small example with three jobs and three machines is provided. Table 2 shows the processing and setup times of jobs. Let the processing order of jobs be $\pi = [\pi(2), \pi(1), \pi(3)]$. The departure time of each job is determined by using Eqs. (1)-(4) as follows:

$d_{\pi(2),0} = d_{\pi(0),1} + S_{\pi(0),\pi(2),1} = 3; d_{\pi(2),1} = \max\{d_{\pi(0),2} + S_{\pi(0),\pi(2),2}, d_{\pi(2),0} + p_{\pi(2),1}\} = 6; d_{\pi(2),2} = \max\{d_{\pi(0),3} + S_{\pi(0),\pi(2),3}, d_{\pi(2),1} + p_{\pi(2),2}\} = 10; d_{\pi(2),3} = d_{\pi(2),2} + p_{\pi(2),3} = 14; d_{\pi(1),0} = d_{\pi(2),1} + S_{\pi(2),\pi(1),1} = 9; d_{\pi(1),1} = \max\{d_{\pi(2),2} + S_{\pi(2),\pi(1),2}, d_{\pi(1),0} + p_{\pi(1),1}\} = 13; d_{\pi(1),2} = \max\{d_{\pi(2),3} + S_{\pi(2),\pi(1),3}, d_{\pi(1),1} + p_{\pi(1),2}\} = 18; d_{\pi(1),3} = d_{\pi(1),2} + p_{\pi(1),3} = 20; d_{\pi(3),0} = d_{\pi(1),1} + S_{\pi(1),\pi(3),1} = 15; d_{\pi(3),1} = \max\{d_{\pi(1),2} + S_{\pi(1),\pi(3),2}, d_{\pi(3),0} + p_{\pi(3),1}\} = 20; d_{\pi(3),2} = \max\{d_{\pi(1),3} + S_{\pi(1),\pi(3),3}, d_{\pi(3),1} + p_{\pi(3),2}\} = 24; d_{\pi(3),3} = d_{\pi(3),2} + p_{\pi(3),3} = 27.$

Then, based on the forward calculation (see Eqs. (1)-(5)), it has $C_{\max}(\boldsymbol{\pi}) = d_{\pi(3),3} = 27.$

According to Eqs. (7)-(11), the duration time of each job in π can be computed as follows:

$f_{\pi(3),3} = f_{\pi(3),4} + p_{\pi(3),3} = 3; f_{\pi(3),2} = f_{\pi(3),3} + p_{\pi(3),2} = 5; f_{\pi(3),1} = f_{\pi(3),2} + p_{\pi(3),1} = 7; f_{\pi(1),4} = f_{\pi(3),3} + S_{\pi(1),\pi(3),3} = 7; f_{\pi(1),3} = \max\{f_{\pi(3),2} + S_{\pi(1),\pi(3),2}, f_{\pi(1),4} + p_{\pi(1),3}\} = 9; f_{\pi(1),2} = \max\{f_{\pi(3),1} + S_{\pi(1),\pi(3),1}, f_{\pi(1),3} + p_{\pi(1),2}\} = 12; f_{\pi(1),1} = f_{\pi(1),2} + p_{\pi(1),1} = 14; f_{\pi(2),4} = f_{\pi(1),3} + S_{\pi(2),\pi(1),3} = 13; f_{\pi(2),3} = \max\{f_{\pi(1),2} + S_{\pi(2),\pi(1),2}, f_{\pi(2),4} + p_{\pi(2),3}\} = 17; f_{\pi(2),2} = \max\{f_{\pi(1),1} + S_{\pi(2),\pi(1),1}, f_{\pi(2),3} + p_{\pi(2),2}\} = 21; f_{\pi(2),1} = f_{\pi(2),2} + p_{\pi(2),1} = 24; f_{\pi(0),4} = f_{\pi(2),3} + S_{\pi(0),\pi(2),3} = 20; f_{\pi(0),3} = \max\{f_{\pi(2),2} + S_{\pi(0),\pi(2),2}, f_{\pi(0),4} + p_{\pi(0),3}\} = 25; f_{\pi(0),2} = \max\{f_{\pi(2),1} + S_{\pi(0),\pi(2),1}, f_{\pi(0),3} + p_{\pi(0),2}\} = 27; f_{\pi(0),1} = f_{\pi(0),2} + p_{\pi(0),1} = 27.$

Then, based on the backward calculation (see Eqs. (7)-(12)), it has $C_{\max}(\boldsymbol{\pi}) = f_{\pi(0),1} = 27.$

To be more intuitive, we draw Gantt charts illustrating the forward and backward calculations in Fig. 1. As shown in Fig. 1, the front delay is determined by the first job in π, and the non-processing time of the machine includes both the blocking time and idle time.

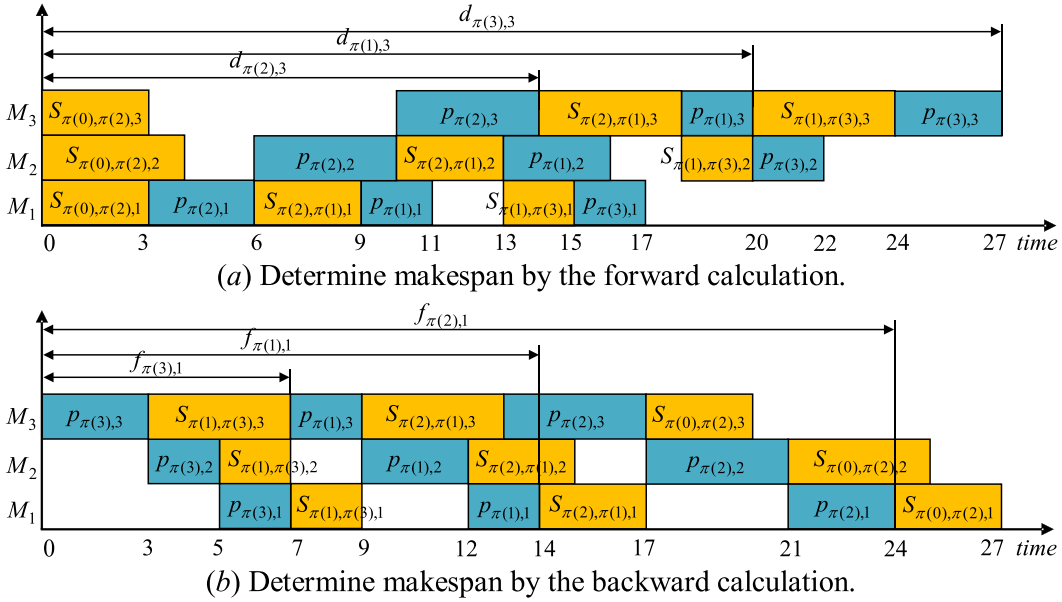

Fig. 1. The Gantt chart of BFSP_SDST with two calculations.

4. MCEDA for BFSP_SDST

In this section, the matrix-cube-based estimation of distribution algorithm (MCEDA) is proposed to address the BFSP_SDST with makespan criterion. In the following subsections, the heuristic and initialization, the multi-dimensional probabilistic model, the diversity controlling mechanism, the multi-neighborhood based local search are firstly described in detail, and then the MCEDA's framework is outlined. Meanwhile, the analysis of MCEDA's computational complexity is provided.

4.1. Heuristic and initialization

The solution representation has a significant effect on the performance of HIOAs. As is reported in the literature, the permutation-based representation has been widely used for various FSPs (Shao, et al., 2017, 2018a, 2018b). Therefore, we utilize the permutation-based encoding scheme to describe feasible solutions to the BFSP_SDST. Each solution corresponds to a specific scheduling scheme for the problem under consideration. Note that the quality of the initial population has an important impact on the search efficiency of HIOAs. If all initial solutions are generated randomly, their quality cannot be guaranteed. Conversely, the initial population formed only by constructive heuristics may be deficient in diversity and dynamism, resulting in premature convergence (Wang, et al., 2010). Some better initial solutions can narrow the search scope suitably, especially for the large-scale instances. Therefore, the initial population should be constructed with a certain quality, *i.e.*, only a few high-quality individuals should be formed via heuristics, while the others are generated randomly. To balance quality and diversity, the population is initialized by using a hybrid strategy. In this section, combining the problem characteristics of BFSP_SDST, the PFT_NEH(x) heuristic based on the PFT and NEH heuristics (Tasgetiren, et al., 2017) and the PFZ_RZ(x) heuristic based on the PFZ and RZ heuristics (Rajendran & Ziegler, 1997) are proposed to generate some initial individuals.

4.1.1. PFT_NEH(x) heuristic

The NEH heuristic (Nawaz, et al., 1983) is a straightforward but pretty powerful constructive heuristic for PFSP and BFSP with makespan criterion in the literature (Pan & Wang, 2012; Tasgetiren, et al., 2017; Wang, et al., 2010). The basic idea behind the NEH heuristic is that jobs with a longer total processing time should be given higher priority. However, when blocking constraints are taken into account, providing higher priority to jobs with a longer total processing time may result in blocking of jobs between machines, yielding in a larger front delay (Wang, et al., 2010). With a longer front delay, the total idle and blocking times may be increased, resulting in decreased machine utilization and increased maximum completion time. Therefore, a suitable strategy for the BFSP is to prioritize jobs with both smaller total processing time and shorter front delay. As illustrated in Fig. 1, when determining the priority of jobs, the total idle and blocking times of machines, as well as the front delay of jobs, must be considered. Numerous studies in the existing literature have shown that after producing the initial sequence, applying the NEH heuristic (Nawaz, et al., 1983) may considerably improve the solution quality. Thus, the PF heuristic (McCormick, et al., 1989) coupled with the NEH heuristic, *i.e.*, PF_NEH(x), is proposed to solve the BFSP with makespan criterion (Pan & Wang, 2012). The outline of the PF heuristic is given in Algorithm 1. For the PF heuristic, the initial job at the first position in the partial sequence is determined by the shortest total processing time. Then, it prioritizes the other jobs by using the total idle and blocking times as a cost function. However, it is obvious that the front delay of the first job cannot be ignored and should be taken into account (Ribas, et al., 2015). Tasgetiren, et al. (2017) tackled such issue by extending the PF heuristic and developing the PFT heuristic, which is an effective heuristic for solving $Fm/blocking/C_{max}$. In this subsection, the PFT heuristic is also adapted to address the $Fm/blocking,SDSTs/C_{max}$. In the PFT heuristic, it prioritizes the jobs by an indicator $I(i)$ that contains the front delay and total processing time. After ascending each job according to $I(i)$, an initial sequence of jobs is obtained. The indicator $I(i)$ can be calculated by the formula in Eq. (14).

$$I(i) = \left(\sum_{j=1}^{m}(m-j)p_{i,j}\right)\cdot\frac{2}{m-1} + \sum_{j=1}^{m}p_{i,j}, \quad i = 1, 2, \ldots, n. \tag{14}$$

According to Eq. (14), the job with the lowest priority indicator is obtained, and such job is chosen as the first job in the initial sequence. Then, the rest of the jobs are added to the initial sequence via the cost index σ_i to produce a complete candidate solution. To be specific, let U be the set of unscheduled jobs and $\tilde{\boldsymbol{\pi}} = [\tilde{\pi}(1), \tilde{\pi}(2), \ldots, \tilde{\pi}(i-1)]$ be a partial sequence containing $i-1$ jobs. In order to determine the job $\tilde{\pi}(i)$ at the ith position in $\tilde{\boldsymbol{\pi}}$, each of the $n-i+1$ unscheduled jobs is attempted to be placed at such position, and the job with the lower cost index value is

placed at the ith position of $\widetilde{\boldsymbol{\pi}}$. The cost index σ_i is given in Eq. (15).

$$\sigma_i = (n-i-2)\sum_{j=1}^{m}\left(d_{i,j} - d_{i-1,j} - p_{i,j}\right) + d_{i,m} \tag{15}$$

Obviously, if job $\widetilde{\pi}(i)$ is added to the partial sequence $\widetilde{\boldsymbol{\pi}}$ with a minimum cost index σ_i, it means that the total idle and blocking time, as well as the departure time of job $\widetilde{\pi}(i)$ on the last machine, are all minimized. Therefore, the PFT heuristic is used to determine an initial feasible solution $\boldsymbol{\pi} = [\pi(1), \pi(2), ..., \pi(n)]$, and the NEH heuristic is used to further improve the quality of such solution. Let $\widetilde{\boldsymbol{\pi}}^{\mathbf{1}} = [\pi(1), \pi(2), ..., \pi(n-\lambda)]$ and $\widetilde{\boldsymbol{\pi}}^{\mathbf{2}} = [\pi(n-\lambda+1), \pi(n-\lambda+2), ..., \pi(n)]$ be two subsequences of $\boldsymbol{\pi}$. Each job in $\widetilde{\boldsymbol{\pi}}^{\mathbf{2}}$ is extracted and reinserted into all possible positions in $\widetilde{\boldsymbol{\pi}}^{\mathbf{1}}$, and the best position for each job extracted in $\widetilde{\boldsymbol{\pi}}^{\mathbf{2}}$ is determined and then inserted it into $\widetilde{\boldsymbol{\pi}}^{\mathbf{1}}$ till a new solution $\boldsymbol{\pi}'$ is formed. According to the PF-NEH(x) proposed by Pan and Wang (2012), the PFT heuristic that incorporates the NEH heuristic is denoted as PFT_NEH(x), as detailed in Algorithm 2.

Algorithm 1: Profile fitting (PF)

Input: The processing time of jobs $p_{\pi(i),j}, i = 1,...,n;\ j = 1,...,m$.

Output: An initial solution $\boldsymbol{\pi}$.

for $i = 1$ *to* n **do**
  Calculate the total processing time of each job, *i.e.*,
  $TP(i) = \sum_{j=1}^{m} p_{i,j}$; *//obtain total processing time.*
end for
Select a job with the smallest TP as the initial job. Set $\boldsymbol{\pi} = [\pi(1)]$.
$U \leftarrow J - \{\pi(1)\}$. Let job in U be $\pi_u(t)$, $t = 1,2,...,|U|$.
for $i = 1$ *to* $n-2$ **do**
  Compute the departure time $d_{\pi(i),j}$ for $\boldsymbol{\pi} = [\pi(1), \pi(2), ..., \pi(i)]$.
  for $t = 1$ *to* $|U|$ **do**
    Compute the departure time $d_{\pi_u(t),j} = d_{\pi(i+1),j}$. If job $\pi_u(t)$ is appended and became the $(i+1)$th job in partial sequence $\boldsymbol{\pi}$.
    Compute the sum of idle and blocking time.
    $$\delta_{\pi_u(t),i} = \sum_{j=1}^{m}\left(d_{\pi(i+1),j} - d_{\pi(i),j} - p_{\pi_u(t),j}\right).$$
    Select a job resulting in smallest $\delta_{\pi_u(t),i}$ as the $(i+1)$th job in $\boldsymbol{\pi}$.
    Remove the selected job $\pi_u(t)$ from the unscheduled job set U.
  end for
end for
The last job in U is directly appended to $\boldsymbol{\pi}$ as the nth job of $\boldsymbol{\pi}$.
return $\boldsymbol{\pi}$.

Algorithm 2:PFT_NEH(x)

Input: $\boldsymbol{\pi}$, $\bar{\boldsymbol{\Pi}}$, x, λ.
Output: The best solution $\boldsymbol{\pi}$ in $\bar{\boldsymbol{\Pi}}$.
Initial solution set $\bar{\boldsymbol{\Pi}}$. Set $\bar{\boldsymbol{\Pi}} \leftarrow \varnothing$.
Initial sequence $\hat{\boldsymbol{\pi}} = [\hat{\pi}(1), \hat{\pi}(2), ..., \hat{\pi}(n)]$ is obtained by ascending jobs in unscheduled job set U according to the priority indicator $I(i)$.
for $i = 1$ *to* x **do**
  Select job $\hat{\pi}(i)$ from $\hat{\boldsymbol{\pi}} = [\hat{\pi}(1), \hat{\pi}(2), ..., \hat{\pi}(n)]$ as the initial job in $\boldsymbol{\pi}$.
  Perform the PFT heuristic to produce a solution $\boldsymbol{\pi}$. Set $\bar{\boldsymbol{\Pi}} \leftarrow \bar{\boldsymbol{\Pi}} \cup \boldsymbol{\pi}$.
  Divide $\boldsymbol{\pi} = [\pi(1), \pi(2), ..., \pi(n)]$ into two partial sequences by λ, $\tilde{\boldsymbol{\pi}}^{\mathbf{1}} = [\pi(1), \pi(2), ..., \pi(n-\lambda)]$, $\tilde{\boldsymbol{\pi}}^{\mathbf{2}} = [\pi(n-\lambda+1), \pi(n-\lambda+2), ..., \pi(n)]$.
  for $k = n - \lambda + 1$ *to* n **do** //*NEH heuristic*.
    Select job $\pi(k)$ from $\boldsymbol{\pi}$ and reinsert it into $n - \lambda + 1$ possible positions of $\tilde{\boldsymbol{\pi}}^{\mathbf{1}}$ to obtain $n - \lambda + 1$ partial sequences.
    Select a partial sequence resulting in the lowest makespan.
  end for
  Obtain a new feasible solution $\boldsymbol{\pi}'$. Set $\bar{\boldsymbol{\Pi}} \leftarrow \bar{\boldsymbol{\Pi}} \cup \boldsymbol{\pi}'$.
end for
return $\bar{\boldsymbol{\Pi}}$.

In Algorithm 2, the value of x in the first layer loop directly determines the number of solutions obtained in $\bar{\boldsymbol{\Pi}}$, and the value of λ in the second loop controls the number of jobs to be inserted by the NEH heuristic. Note that the tie-breaking strategy is utilized in both the ordering and constructing phases if two jobs with the same $I(i)$ or two positions result in the same σ_i. It is clear that there are $(2n-\lambda+1)\lambda/2$ partial sequences that need to be calculated in the NEH heuristic of the PFT_NEH(x). It should be noted that the fast *Insert*-based neighbor evaluation method explained in Subsection 4.3.1 is employed in the NEH heuristic, and its complexity can be reduced from $O(mn^3)$ to $O(mn^2)$. Thus, Algorithm 2 has a total complexity of about $O(xmn^2)$.

4.1.2. PFZ_RZ(x) heuristic

The RZ heuristic is an effective heuristic proposed by Rajendran and Ziegler (1997). As with the PFT heuristic, the RZ heuristic can construct a complete solution sequence from a partial sequence by using basic insertion neighborhood. However, when the RZ heuristic is used to construct a solution, the jobs to be inserted are chosen according to a specified reference sequence. Consider two solution sequences [3,5,4,1,2] and [1,2,3,4,5], each consisting of five jobs, and assume that the former is a reference sequence and the latter is an incumbent sequence. The RZ heuristic first removes job 3 from the incumbent sequence [1,2,3,4,5] and reinserts it into all possible positions of the incumbent sequence. Then, it removes job 2 from the incumbent sequence to execute the insertion operation. This procedure is repeated until all jobs in the reference sequence are already picked and the best feasible solution is produced. It is crucial to create the reference sequence of the RZ heuristic as the baseline. As shown in Eq. (14), two sorts of indicators, namely the front delay of the first job (*i.e.*, $\sum_{j=1}^{m}(m-j)p_{i,j}$) and the total processing time of each job (*i.e.*, $\sum_{j=1}^{m}p_{i,j}$), have an impact on the performance of heuristics for solving BFSP. However, besides these two indicators, the average processing time, the standard deviation of processing time, and the skewness should be considered depending on the problem's characteristics. Since the strong correlation between skewness and front delay, if skewness is small, front delay may likewise decrease. The formulas for these indicators are given in Eqs. (16) to (18):

$$T_{avg}(i) = \frac{1}{m}\left(\sum_{j=1}^{m} p_{i,j}\right), \tag{16}$$

$$T_{std}(i) = \sqrt{\frac{1}{m-1}\sum_{j=1}^{m}\left(p_{i,j} - T_{avg}(i)\right)^2}, \tag{17}$$

$$T_{ske}(i) = \frac{\frac{1}{m}\left(\sum_{j=1}^{m}\left(p_{i,j} - T_{avg}(i)\right)^3\right)}{\left(\sqrt{\frac{1}{m}\left(\sum_{j=1}^{m}\left(p_{i,j} - T_{avg}(i)\right)^2\right)}\right)^3}. \tag{18}$$

Based on Eqs. (17)-(18), the newly proposed job priority indicator $I_Z(i)$ is shown in Eq. (19).

$$I_Z(i) = \left(\sum_{j=1}^{m}(m-j)p_{i,j}\right)\cdot\frac{2}{m-1} + \left(T_{std}(i) + T_{ske}(i)\right), i = 1, 2, ..., n. \tag{19}$$

Each job J_i in the job set J is arranged in ascending order according to the $I_Z(i)$ in Eq. (19) to produce a job sequence that serves as a reference sequence for the RZ heuristic. As with PFT_NEH(x) in Subsection 4.1.1, the initial solution for the RZ heuristic is produced by using a newly presented heuristic, namely the PFZ heuristic, and then the *Insert*-based neighborhood search is performed on the initial solution in accordance with the order of jobs in the reference sequence. If an improvement is achieved, the worse solution is replaced by a new one and then the cycle continues. According to the above considerations, the PFZ_RZ(x) heuristic is proposed in this subsection by incorporating the PFZ heuristic and the RZ heuristic (see Algorithm 3). In PFZ_RZ(x), a total of x feasible solutions are generated. Each solution produced by PFZ_RZ(x) needs to

calculate $(n-1)^2$ neighbor solutions. In order to reduce the computational cost and speed up the neighborhood search, the fast *Insert*-based neighbor evaluation method can be used in the implementation of Algorithm 3 to effectively decrease the total time complexity. It is clear that the complexity of Algorithm 3 is $O(xmn^2)$, which is the same as that of Algorithm 2. According to the relevant conclusions in the literature (Pan, et al., 2013), the parameter of two heuristics, PFT_NEH(x) and PFZ_RZ(x), is set to $\lambda = 20$ and $x = 5$.

To be specific, 10% of the solutions in the initial population are created by PFT_NEH(x), whereas 10% are produced by PFZ_RZ(x). The remaining 80% of solutions are randomly generated. Due to the fact that the two proposed heuristics are designed according to the problem's characteristics, the initial population with high-quality individuals or solutions may contain rich and valuable structural features and promising patterns. Hence, it may be desirable to expend a certain amount of computational effort to generate high-quality initial population.

Algorithm 3:PFZ_RZ(x)

Input: $\boldsymbol{\pi}$, $\bar{\boldsymbol{\Pi}}$, x, λ.
Output: The best solution $\boldsymbol{\pi}$ in $\bar{\boldsymbol{\Pi}}$.
Initial solution set $\bar{\boldsymbol{\Pi}}$. Set $\bar{\boldsymbol{\Pi}} \leftarrow \varnothing$.
Initial sequence $\hat{\boldsymbol{\pi}} = [\hat{\pi}(1), \hat{\pi}(2), ..., \hat{\pi}(n)]$ is obtained by ascending jobs in the unscheduled job set U according to the priority indicator $I(i)$.
Obtain reference sequence $\boldsymbol{\pi}^{\mathbf{s}} = [\pi^s(1), \pi^s(2), ..., \pi^s(n)]$ by ascending jobs in U according to the priority indicator $I_Z(i)$ in Eq. (19).
for $i = 1$ *to* x **do**
  Select job $\hat{\pi}(i)$ from $\hat{\boldsymbol{\pi}} = [\hat{\pi}(1), \hat{\pi}(2), ..., \hat{\pi}(n)]$ as the initial job in $\boldsymbol{\pi}$.
  Perform the PFZ heuristic to produce a solution $\boldsymbol{\pi}$. Set $\boldsymbol{\pi}^{\mathbf{t}} = \boldsymbol{\pi}$.
  for $k = n - \lambda + 1$ *to* n **do** //*NEH heuristic*.
    Remove the job $\pi^s(k) \in \boldsymbol{\pi}^{\mathbf{s}}$ in $\boldsymbol{\pi}^{\mathbf{t}}$ and reinsert the job $\pi^s(k)$ into n possible positions in $\boldsymbol{\pi}^{\mathbf{t}}$ to obtain n partial sequences.
    Select solution $\boldsymbol{\pi}'$ with the lowest makespan after removing repetition.
    Update $\boldsymbol{\pi}^{\mathbf{t}}$. Set $\boldsymbol{\pi}^{\mathbf{t}} = \boldsymbol{\pi}'$.
    if $C_{\max}(\boldsymbol{\pi}^{\mathbf{t}}) < C_{\max}(\boldsymbol{\pi})$ **then**
      $\boldsymbol{\pi} = \boldsymbol{\pi}^{\mathbf{t}}$.
    end if
  end for
  Obtain a feasible solution $\boldsymbol{\pi}'$. Let $\bar{\boldsymbol{\Pi}} \leftarrow \bar{\boldsymbol{\Pi}} \cup \boldsymbol{\pi}'$.
end for
return $\bar{\boldsymbol{\Pi}}$.

4.2. Global search guided by multi-dimensional probabilistic model

4.2.1. Multi-dimensional probabilistic model

In general, permutation-based solutions have a number of distinguishing features, including the priority order of jobs and the distribution characteristics of job blocks. In this subsection, a multi-dimensional probabilistic model is proposed to capture promising patterns and adequately accumulate valuable structural information, which can effectively drive the search toward high-quality regions.

4.2.1.1. Matrix cube. In order to effectively extract excellent structural features or promising patterns from quality individuals in a population, a matrix cube structure is designed to capture these valuable structural features and to reasonably retain promising patterns. Let $\boldsymbol{Pop}(\boldsymbol{gen})$ be the population at generation *gen*, and $\boldsymbol{SPop}(\boldsymbol{gen})$ be the high-quality sub-population or superior solutions derived from $\boldsymbol{Pop}(\boldsymbol{gen})$, *i.e.*,$\boldsymbol{SPop}(\boldsymbol{gen}) = \{\boldsymbol{\pi}_{\boldsymbol{Sbest}}^{\boldsymbol{gen,1}}, \boldsymbol{\pi}_{\boldsymbol{Sbest}}^{\boldsymbol{gen,2}}, ..., \boldsymbol{\pi}_{\boldsymbol{Sbest}}^{\boldsymbol{gen,spsize}}\}$, where *popsize* and *spsize* respectively represent the size of $\boldsymbol{Pop}(\boldsymbol{gen})$ and $\boldsymbol{SPop}(\boldsymbol{gen})$, $gen = 1, 2, \ldots, maxgen$. *maxgen* is the maximum number of runs of the algorithm. Let $\boldsymbol{\pi}_{\boldsymbol{Sbest}}^{\boldsymbol{gen,k}}$ be the kth individual in $\boldsymbol{SPop}(\boldsymbol{gen})$, *i.e.*, $\boldsymbol{\pi}_{\boldsymbol{Sbest}}^{\boldsymbol{gen,k}} = [\pi_{Sbest}^{gen,k}(1), \pi_{Sbest}^{gen,k}(2), ..., \pi_{Sbest}^{gen,k}(n)]$, $k = 1, ..., spsize$. Without loss of generality, $\boldsymbol{MC}_{\boldsymbol{n\times n\times n}}^{\boldsymbol{gen}}$ is defined as the matrix cube at generation *gen*. $MC_{n\times n\times n}^{gen}(x, y, z), x, y, z \in \{1, 2, ..., n\}$ is the element in $\boldsymbol{MC}_{\boldsymbol{n\times n\times n}}^{\boldsymbol{gen}}$ with the ternary subscript (x, y, z), where x corresponds to the xth position of the solution sequence, and (y, z) is used to represent the job block $[y, z]$ at that position. Firstly, for the kth individual $\boldsymbol{\pi}_{\boldsymbol{Sbest}}^{\boldsymbol{gen,k}}$ in $\boldsymbol{SPop}(\boldsymbol{gen})$, the sequential relationship of the job $\pi_{Sbest}^{gen,k}(x+1)$ that appears immediately after the job $\pi_{Sbest}^{gen,k}(x)$ located at the xth position in $\boldsymbol{\pi}_{\boldsymbol{Sbest}}^{\boldsymbol{gen,k}}$ can be recorded separately by using the indicator function $IF_{n\times n\times n}^{gen,k}(x, y, z)$, as given in Eq. (20).

$$IF_{n\times n\times n}^{gen,k}(x, y, z) = \begin{cases} 1, & if\ y = \pi_{Sbest}^{gen,k}(x)\ and\ z = \pi_{Sbest}^{gen,k}(x+1) \\ 0, & else \end{cases}, \quad x = 1, ..., n-1; y, z = 1, ..., n; k = 1, ..., spsize. \tag{20}$$

Then, the characteristic information about the order of jobs and the

distribution of blocks for each of the selected superior solutions is obtained based on Eq. (21).

$$MC^{gen}_{n\times n\times n}(x,y,z)=\sum_{k=1}^{spsize} IF^{gen,k}_{n\times n\times n}(x,y,z),\quad x=1,\ldots,n-1;y,z=1,\ldots,n. \tag{21}$$

Finally, the detailed definition of the proposed matrix cube is described below.

$$\boldsymbol{MC^{gen}_{n\times n\times n}}(\boldsymbol{x},\boldsymbol{y})=\left[MC^{gen}_{n\times n\times n}(x,y,1),MC^{gen}_{n\times n\times n}(x,y,2),\ldots,MC^{gen}_{n\times n\times n}(x,y,n)\right]_{1\times n},\quad x=1,2,\ldots,n-1;y=1,2,\ldots,n. \tag{22}$$

$$\boldsymbol{MC^{gen}_{n\times n\times n}}(\boldsymbol{x})=\begin{bmatrix}\boldsymbol{MC^{gen}_{n\times n\times n}}(\boldsymbol{x},\boldsymbol{1})\\ \vdots\\ \boldsymbol{MC^{gen}_{n\times n\times n}}(\boldsymbol{x},\boldsymbol{n})\end{bmatrix}_{n\times 1}=\begin{bmatrix}MC^{gen}_{n\times n\times n}(x,1,1) & \cdots & MC^{gen}_{n\times n\times n}(x,1,n)\\ \vdots & \ddots & \vdots\\ MC^{gen}_{n\times n\times n}(x,n,1) & \cdots & MC^{gen}_{n\times n\times n}(x,n,n)\end{bmatrix}. \tag{23}$$

According to Eq. (23), the two-dimensional submatrix $\boldsymbol{MC^{gen}_{n\times n\times n}}(\boldsymbol{x})$ can record the characteristic information about job order and job block distribution at the xth position in all superior solutions. In other words, the matrix cube structure can exactly determine the priority of each job and the distribution of the job block $[y,z]$ located in the xth position of the kth individual in $\boldsymbol{SPop}(\boldsymbol{gen})$, *i.e.*, $[\pi^{gen,k}_{Sbest}(x),\pi^{gen,k}_{Sbest}(x+1)]$. it can retain total order relationships via a series of position-based submatrices, *i.e.*, $\boldsymbol{MC^{gen}_{n\times n\times n}}(\boldsymbol{1})$, $\boldsymbol{MC^{gen}_{n\times n\times n}}(\boldsymbol{2})$, ..., $\boldsymbol{MC^{gen}_{n\times n\times n}}(\boldsymbol{n})$. Thus, these advantageous characteristics or promising patterns derived from superior solutions can be effectively and intuitively recognized and retained. By adopting the matrix cube $\boldsymbol{MC^{gen}_{n\times n\times n}}$ described above, the multi-dimensional probabilistic model can be established. To illustrate the proposed $\boldsymbol{MC^{gen}_{n\times n\times n}}$, an example of five superior solutions ($spsize=5$) containing four jobs ($n=4$) is used to instantiate it. In this case, the size of $\boldsymbol{SPop}(\boldsymbol{gen})$ is $spsize=5$ and $gen=1$. The selected superior solutions are $\boldsymbol{\pi^{1,1}_{Sbest}}=[\pi^{1,1}_{Sbest}(1),\pi^{1,1}_{Sbest}(2),\ \pi^{1,1}_{Sbest}(3),\pi^{1,1}_{Sbest}(4)]=[1,\ 2,\ 3,\ 4]$, $\boldsymbol{\pi^{1,2}_{Sbest}}=[\pi^{1,2}_{Sbest}(1),\pi^{1,2}_{Sbest}(2),\ \pi^{1,2}_{Sbest}(3),\pi^{1,2}_{Sbest}(4)]=[2,\ 3,\ 1,\ 4]$, $\boldsymbol{\pi^{1,3}_{Sbest}}=[\pi^{1,3}_{Sbest}(1),\pi^{1,3}_{Sbest}(2),\ \pi^{1,3}_{Sbest}(3),\pi^{1,3}_{Sbest}(4)]=[3,\ 2,\ 1,\ 4]$, $\boldsymbol{\pi^{1,4}_{Sbest}}=[\pi^{1,4}_{Sbest}(1),\pi^{1,4}_{Sbest}(2),\ \pi^{1,4}_{Sbest}(3),\pi^{1,4}_{Sbest}(4)]=[4,\ 3,\ 2,\ 1]$, $\boldsymbol{\pi^{1,5}_{Sbest}}=[\pi^{1,5}_{Sbest}(1),\pi^{1,5}_{Sbest}(2),\pi^{1,5}_{Sbest}(3),\pi^{1,5}_{Sbest}(4)]=[4,3,1,2]$, respectively. For the first position ($x=1$) of all individuals, it is clear that 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.*, $y=4,z=3$) appeared in these individuals from $\boldsymbol{\pi^{gen,1}_{Sbest}}$ to $\boldsymbol{\pi^{gen,5}_{Sbest}}$, is recorded in accordance with Eqs. (20)-(23) as follows:

$$MC^{1}_{4\times4\times4}(1,1,2)=\sum_{k=1}^{5}IF^{1,k}_{4\times4\times4}(1,1,2)=IF^{1,1}_{4\times4\times4}(1,1,2)+IF^{1,2}_{4\times4\times4}(1,1,2)+IF^{1,3}_{4\times4\times4}(1,1,2)+IF^{1,4}_{4\times4\times4}(1,1,2)+IF^{1,5}_{4\times4\times4}(1,1,2)=1+0+0+0+0=1,$$

$$MC^{1}_{4\times4\times4}(1,2,3)=\sum_{k=1}^{5}IF^{1,k}_{4\times4\times4}(1,2,3)=IF^{1,1}_{4\times4\times4}(1,2,3)+IF^{1,2}_{4\times4\times4}(1,2,3)+IF^{1,3}_{4\times4\times4}(1,2,3)+IF^{1,4}_{4\times4\times4}(1,2,3)+IF^{1,5}_{4\times4\times4}(1,2,3)=0+1+0+0+0=1,$$

$$MC^{1}_{4\times4\times4}(1,3,2)=\sum_{k=1}^{5}IF^{1,k}_{4\times4\times4}(1,3,2)=IF^{1,1}_{4\times4\times4}(1,3,2)+IF^{1,2}_{4\times4\times4}(1,3,2)+IF^{1,3}_{4\times4\times4}(1,3,2)+IF^{1,4}_{4\times4\times4}(1,3,2)+IF^{1,5}_{4\times4\times4}(1,3,2)=0+0+1+0+0=1,$$

$$MC^{1}_{4\times4\times4}(1,4,3)=\sum_{k=1}^{5}IF^{1,k}_{4\times4\times4}(1,4,3)=IF^{1,1}_{4\times4\times4}(1,4,3)+IF^{1,2}_{4\times4\times4}(1,4,3)+IF^{1,3}_{4\times4\times4}(1,4,3)+IF^{1,4}_{4\times4\times4}(1,4,3)+IF^{1,5}_{4\times4\times4}(1,4,3)=0+0+0+1+1=2.$$

The remaining elements in $\boldsymbol{MC^{gen}_{4\times4\times4}}(\boldsymbol{1})$ are set to zero. Likewise, for the second position ($x=2$) of all individuals, 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 also be recorded, respectively. Then we have.

$$MC^{1}_{4\times4\times4}(2,2,1)=\sum_{k=1}^{5}IF^{1,k}_{4\times4\times4}(2,2,1)=0+0+1+0+0=1,$$

$$MC^{1}_{4\times4\times4}(2,2,3)=\sum_{k=1}^{5}IF^{1,k}_{4\times4\times4}(2,2,3)=1+0+0+0+0=1,$$

$$MC^{1}_{4\times4\times4}(2,3,1)=\sum_{k=1}^{5}IF^{1,k}_{4\times4\times4}(2,3,1)=0+1+0+0+1=2,$$

$$MC^{1}_{4\times4\times4}(2,3,2)=\sum_{k=1}^{5}IF^{1,k}_{4\times4\times4}(2,3,2)=0+0+0+1+0=1.$$

The other cells of $\boldsymbol{MC^{gen}_{4\times4\times4}}(\boldsymbol{2})$ are set to zero. Since four job blocks [1, 2], [1, 4],[2, 1], and [3, 4] are located in the third position ($x=3$), the characteristic information can also be saved, and we have $MC^{1}_{4\times4\times4}(3,1,2)=1$, $MC^{1}_{4\times4\times4}(3,1,4)=2$, $MC^{1}_{4\times4\times4}(3,2,1)=1$, and $MC^{1}_{4\times4\times4}(3,3,4)=1$, respectively. It is worth noting that the information about job blocks in the last position of the job sequence is already included in the penultimate position. Therefore, the values of all the cells in $\boldsymbol{MC^{gen}_{4\times4\times4}}(\boldsymbol{4})$ are set to zero. It is indicated that the job blocks at different positions in superior solutions may be precisely learnt and entirely preserved in accordance with the position-based submatrices in $\boldsymbol{MC^{gen}_{4\times4\times4}}$. However, in the aforementioned case, the critical characteristic information about the similar blocks from $\boldsymbol{\pi^{gen,1}_{Sbest}}$ to $\boldsymbol{\pi^{gen,5}_{Sbest}}$, *i.e.*,[1, 2],[2, 3], and[3, 2], is kept exclusively in the subscripts (1,2), (2,3), and (3, 2) via utilizing the two-dimensional probabilistic model (Pan & Ruiz, 2012). Indeed, for each selected elite solution, the important patterns of the positions in which similar blocks or job blocks are located have been completely lost and fused. Then, the two-dimensional probabilistic model's sampling strategy cannot precisely predict the most proper position to place these valued similar blocks. To compensate the limitations of two-dimensional EDAs, we design the matrix cube to learn the structural information relating to the order relationships of jobs and the position of similar blocks through the use of different layers of $\boldsymbol{MC^{gen}_{n\times n\times n}}$. To be specific, the block $[2,3]$ in $\boldsymbol{\pi^{gen,1}_{Sbest}}=[1,2,3,4]$ can be reserved in $\boldsymbol{MC^{1}_{4\times4\times4}}(\boldsymbol{2})$ (*i.e.*, the second layer of $\boldsymbol{MC^{1}_{4\times4\times4}}$), but the identical block [2,3]), but the identical block $[2,3]$ in $\boldsymbol{\pi^{gen,2}_{Sbest}}=[2,3,1,4]$ can be recorded in $\boldsymbol{MC^{1}_{4\times4\times4}}(\boldsymbol{1})$ (*i.e.*, the first layer of $\boldsymbol{MC^{1}_{4\times4\times4}}$), but the identical block $[2,3]$), respectively. It is clear that $\boldsymbol{MC^{gen}_{n\times n\times n}}$ facilitates the accurate identification and differentiation of various interesting similar blocks located at different positions in job sequences. That is, the distribution characteristics of these similar blocks in the feasible solution space can be precisely described by a multi-dimensional probabilistic model, which may be capable of effectively directing the search toward more promising regions and preventing promising patterns from being destroyed or improperly fused.

4.2.1.2. Probabilistic model. Probabilistic models are essential for the successful application of EDA since they enable for the accurate estimation of the distribution of promising patterns of superior solutions to

the problem under consideration. In other words, it is highly advantageous to enhance the algorithm's performance if the proposed probabilistic model correctly learns the structural features from the selected superior solutions. It should be noted that the proposed matrix cube $\boldsymbol{MC}^{gen}_{n\times n\times n}$ can completely capture the structural features of some superior solutions, *i.e.*, the ordinal of jobs and the dependency of jobs, during the iterative process. In contrast to existing two-dimensional probabilistic models (Jarboui, et al., 2009; Pan & Ruiz, 2012; Wang & Wang, 2016; Wang, et al., 2013), a matrix-cube-based multi-dimensional probabilistic model is devised by taking advantage of promising patterns derived from superior solutions. The critical characteristic information implicit in different solution sequences can be effectively extracted and appropriately accumulated, while valuable information can be interacted with and integrated via an incremental learning mechanism. Offspring individuals can be generated directly by sampling from the proposed probabilistic model. Let $\boldsymbol{PM}^{gen}_{n\times n\times n}$ be the multi-dimensional probabilistic model. Detailed definitions are described in Eqs. (24)-(25).

$$\begin{aligned}&\boldsymbol{PM}^{gen}_{n\times n\times n}(\boldsymbol{x},\boldsymbol{y})=\\&\left[PM^{gen}_{n\times n\times n}(x,y,1),PM^{gen}_{n\times n\times n}(x,y,2),...,PM^{gen}_{n\times n\times n}(x,y,n)\right]_{1\times n},\\&x=1,2,...,n-1;y=1,2,...,n,\end{aligned}\tag{24}$$

$$\begin{aligned}\boldsymbol{PM}^{gen}_{n\times n\times n}(\boldsymbol{x})&=\begin{bmatrix}\boldsymbol{PM}^{gen}_{n\times n\times n}(\boldsymbol{x},\boldsymbol{1})\\\vdots\\\boldsymbol{PM}^{gen}_{n\times n\times n}(\boldsymbol{x},\boldsymbol{n})\end{bmatrix}_{n\times 1}\\&=\begin{bmatrix}PM^{gen}_{n\times n\times n}(x,1,1)&\cdots&PM^{gen}_{n\times n\times n}(x,1,n)\\\vdots&\ddots&\vdots\\PM^{gen}_{n\times n\times n}(x,n,1)&\cdots&PM^{gen}_{n\times n\times n}(x,n,n)\end{bmatrix}.\end{aligned}\tag{25}$$

Each element $PM^{gen}_{n\times n\times n}(x,y,z)$ in $\boldsymbol{PM}^{gen}_{n\times n\times n}$ corresponds to a probability value for the occurrence of the job block $[y,z]$ at the xth position in the genth iteration, referring to the job block's relevance. To clearly describe the multi-dimensional probabilistic model, let $\boldsymbol{S}^{gen}_{\boldsymbol{MC}}(\boldsymbol{x})$ and $\boldsymbol{S}^{gen}_{\boldsymbol{PM}}(\boldsymbol{x})$ represent the summation function of the xth layer of $\boldsymbol{MC}^{gen}_{n\times n\times n}$ and $\boldsymbol{PM}^{gen}_{n\times n\times n}$, respectively, where $\boldsymbol{S}^{gen}_{\boldsymbol{MC}}(\boldsymbol{x})=\sum_{y=1}^{n}\sum_{z=1}^{n}\boldsymbol{MC}^{gen}_{n\times n\times n}(\boldsymbol{x},y,z)$ and $\boldsymbol{S}^{gen}_{\boldsymbol{PM}}(\boldsymbol{x})=\sum_{y=1}^{n}\sum_{z=1}^{n}PM^{gen}_{n\times n\times n}(x,y,z)$. Thus, by utilizing both a matrix cube and a multi-dimensional probabilistic model, the incremental learning update mechanism can be stated in Eq. (26).

$$\begin{aligned}&\boldsymbol{PM}^{gen+1}_{n\times n\times n}(\boldsymbol{x})=\\&(1-r)\times\boldsymbol{PM}^{gen}_{n\times n\times n}(\boldsymbol{x})+r\times\left(\boldsymbol{MC}^{gen}_{n\times n\times n}(\boldsymbol{x})/\boldsymbol{S}^{gen}_{\boldsymbol{MC}}(\boldsymbol{x})\right),\\&x=1,2,...,n-1.\end{aligned}\tag{26}$$

Note that the parameter $r\in[0,1]$ in Eq. (26), represents the learning rate. If $r=1$, the multi-dimensional probabilistic model is updated only by using the matrix cube; otherwise, it is updated by using historical evolutionary information. The proposed updating strategy can adequately accumulate characteristic information about promising patterns of superior solutions. That is, the incremental learning mechanism can take into account not only the current distribution characteristics of similar blocks, but also make use of the previously obtained useful historical information, resulting in a suitable trade-off in terms of learning rate r. Notice that the normalization for each layer of $\boldsymbol{PM}^{gen}_{n\times n\times n}$ in Step 3 should be performed before to sampling, *i.e.*, $\boldsymbol{S}^{gen}_{\boldsymbol{MC}}(\boldsymbol{x})=1, gen>1, x=1,...,n-1$. Moreover, the features of the first job block in the selected superior subpopulation have a considerable effect on the performance of the developed algorithm for dealing with the considered problem. If this feature is not well handled, the algorithm's superiority will be diminished. Therefore, the steps for developing and updating a probability model with the aforementioned characteristics are detailed as follows.

Step 1: Initialize $\boldsymbol{PM}^{\boldsymbol{0}}_{\boldsymbol{n\times n\times n}}$. Set $PM^{0}_{n\times n\times n}(x,y,z)=\begin{cases}0, & x=1;\ y,z=1,...,n,\\1/n^2, & x=2,3,...,n-1;\ y,z=1,...,n.\end{cases}$

Step 2: Obtain $\boldsymbol{MC}^{\boldsymbol{1}}_{\boldsymbol{n\times n\times n}}$ by the initial population $\boldsymbol{Pop}(\boldsymbol{0})$, then compute

$$PM^{1}_{n\times n\times n}(x,y,z)=\begin{cases}\dfrac{MC^{1}_{n\times n\times n}(x,y,z)}{S^{1}_{MC}(x)},\\x=1;\ y,z=1,...,n,\\\dfrac{PM^{0}_{n\times n\times n}(x,y,z)+MC^{1}_{n\times n\times n}(x,y,z)}{S^{0}_{PM}(x)+S^{1}_{MC}(x)},\\x=2,3,...,n-1;\ y,z=1,...,n.\end{cases}$$

Step 3: Set $gen=2$. Compute $\boldsymbol{MC}^{\boldsymbol{gen}}_{\boldsymbol{n\times n\times n}}$ and $\boldsymbol{PM}^{\boldsymbol{gen}}_{\boldsymbol{n\times n\times n}}(\boldsymbol{x})$ using Eq. (26).

$$\begin{aligned}&PM^{gen}_{n\times n\times n}(x,y,z)=\\&(1-r)\times PM^{gen-1}_{n\times n\times n}(x,y,z)+r\times\left(\frac{MC^{gen}_{n\times n\times n}(x,y,z)}{S^{gen}_{MC}(x)}\right),\\&x=1,2,...,n-1;\ y,z=1,2,...,n.\end{aligned}$$

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

Notably, for $x=2,3,...,n-1$, all values in $\boldsymbol{PM}^{\boldsymbol{0}}_{\boldsymbol{n\times n\times n}}$ is initialized to $1/n^2$, ensuring that the entire solution space is uniformly sampled. In addition, when $x=1$, the cell values in the first layer of $\boldsymbol{PM}^{\boldsymbol{0}}_{\boldsymbol{n\times n\times n}}$ are set to 0, instead of $1/n^2$, which can highlight potential patterns of job blocks at the first position of all individuals in the superior subpopulation and effectively increase guidance to promising regions during the initial phase. To illustrate, an example of establishing a matrix-cube-based multi-dimensional probabilistic model is provided below, with the learning rate set to $r=0.3$.

(1) Firstly, $\boldsymbol{MC}^{\boldsymbol{1}}_{\boldsymbol{n\times n\times n}}$ is computed by counting the initial subpopulation via Eqs. (20)-(21). For the first layer of $\boldsymbol{PM}^{\boldsymbol{1}}_{\boldsymbol{n\times n\times n}}$, *i.e.*, $\boldsymbol{PM}^{\boldsymbol{1}}_{\boldsymbol{n\times n\times n}}(\boldsymbol{1})$, the probability values can be calculated as follows: $PM^{1}_{4\times4\times4}(1,1,2)=PM^{1}_{4\times4\times4}(1,2,3)=PM^{1}_{4\times4\times4}(1,3,1)=(0+1)/5=0.2$, $PM^{1}_{4\times4\times4}(1,3,1)=(0+2)/5=0.4$. The values of the other cells are set to zero.

(2) Secondly, the probability values of the second layer of $\boldsymbol{PM}^{\boldsymbol{1}}_{\boldsymbol{n\times n\times n}}$ can be computed as follows:$PM^{1}_{4\times4\times4}(2,2,1)=PM^{1}_{4\times4\times4}(2,2,3)=PM^{1}_{4\times4\times4}(2,3,2)=(1/16+1)/(1+5)=0.177$, $PM^{1}_{4\times4\times4}(2,3,1)=(1/16+2)/(1+5)=0.344$. Then, all other cell values are set to $(1/16+0)/(1+5)=0.010$.

(3) Thirdly, the probability values of the third layer of $\boldsymbol{PM}^{\boldsymbol{1}}_{\boldsymbol{n\times n\times n}}$ can be computed as follows: $PM^{1}_{4\times4\times4}(3,1,2)=PM^{1}_{4\times4\times4}(3,2,1)=PM^{1}_{4\times4\times4}(3,3,4)=(1/16+1)/(1+5)=0.177$ and $PM^{1}_{4\times4\times4}(3,1,4)=(1/16+2)/(1+5)=0.344$. Similarly, the values of the other cells are equal to 0.01.

(4) Finally, $\boldsymbol{PM}^{\boldsymbol{gen}}_{\boldsymbol{n\times n\times n}}$ $(gen>1)$ is updated by using the incremental learning mechanism proposed in Step 3. Considering five superior solutions, each of which contains four jobs, Fig. 2 illustrates the updating process of a multi-dimensional probabilistic model.

4.2.2. New population generation

According to the above subsection, it is clear that proper probabilistic models may be employed to effectively extract excellent features from the superior solutions. The sampling strategy also has an effect on the search behavior of EDA-based algorithms, since such sampling strategy determines how to guide the population's evolutionary direction in the search space. Therefore, it is important to set up suitable sampling strategies for sampling from the multi-dimensional

probabilistic model to generate a new population.

Let $\boldsymbol{\pi}^{gen,k} = [\pi^{gen,k}(1), \pi^{gen,k}(2), \ldots, \pi^{gen,k}(n)]$ denote the kth individual in $\boldsymbol{Pop}(\boldsymbol{gen})$, $k = 1, 2, \ldots, popsize$. Due to the fact that the multi-dimensional probabilistic model records the probability information of both the order of jobs and the distribution of blocks, the probability value of each block or similar block$[\pi^{gen,k}(i-1), \pi^{gen,k}(i)]$, $(i = 2, \ldots, n)$ in the job sequence $\boldsymbol{\pi}^{gen,k}$ is stored in the $(i-1)$th layer of $\boldsymbol{PM}^{gen}_{n\times n\times n}$, *i.e.*,$\boldsymbol{PM}^{gen}_{n\times n\times n}(\boldsymbol{i-1})$. The selection of job $\pi^{gen,k}(i)$ at the ith position is dependent upon the appearance of job $\pi^{gen,k}(i-1)$ at the $(i-1)$th position in $\boldsymbol{\pi}^{gen,k}$. Let $SelectJob(\boldsymbol{PM}^{gen}_{n\times n\times n}, \boldsymbol{\pi}^{gen,k}, i)$ be a selection function that is utilized to determine the candidate job π_c at the ith position of $\boldsymbol{\pi}^{gen,k}$ $(i > 1)$. That is, the selection of the candidate job π_c at the ith position in the $\boldsymbol{\pi}^{gen,k}$ should sample from the $(\pi^{gen,k}(i-1))$th row of the $(i-1)$th layer in $\boldsymbol{PM}^{gen}_{n\times n\times n}$, *i.e.*, $\boldsymbol{PM}^{gen}_{n\times n\times n}(\boldsymbol{i-1}, \boldsymbol{\pi}^{gen,k}(\boldsymbol{i-1}))$. To eliminate duplication to guarantee the generation of feasible solutions, the probability values of the π_cth column from the ith layer to the $(n-1)$th layer in $\boldsymbol{PM}^{gen}_{n\times n\times n}$ should be set as 0 and all elements in $\boldsymbol{PM}^{gen}_{n\times n\times n}$ need to be renormalized if the job π_c is already scheduled in the ith position of $\boldsymbol{\pi}^{gen,k}$. The procedure of $SelectJob(\boldsymbol{PM}^{gen}_{n\times n\times n}, \boldsymbol{\pi}^{gen,k}, i)$ is described in Algorithm 4.In order to precisely learn the promising patterns of jobs located at the first position of high-quality individuals in the population, and to compress the front delay as much as possible, a first position-based selection strategy (FPBSS) is presented in Algorithm 5 for determining the first job in the $\boldsymbol{\pi}^{gen,k}$. According to Algorithms 4 and 5, the new population generation method is provided in Algorithm 6. As seen from Algorithm 6, if $i = 1$, the job π_c is selected by $FPBSS(\boldsymbol{PM}^{gen}_{n\times n\times n}, \boldsymbol{\pi}^{gen,k})$, otherwise, the job block $[\pi^{gen+1,k}(i), \pi^{gen+1,k}(i+1)]$ at the ith position of $\boldsymbol{\pi}^{gen,k}$ is selected by $SelectJob(\boldsymbol{PM}^{gen}_{n\times n\times n}, \boldsymbol{\pi}^{gen,k}, i)$. Thereafter, job blocks at different positions in the solution sequence are determined, and these job blocks are connected according to the sequential relationship among jobs to produce new feasible solutions.

Algorithm 4: $SelectJob(PM^{gen}_{n\times n\times n}, \boldsymbol{\pi}^{gen,k}, i)$

Input: $\mathbf{PM^{gen}_{n\times n\times n}}$, $\boldsymbol{\pi}^{\mathbf{gen,k}}$ and i.

Output: the candidate job π_c.

Produce a random number r, *i.e.*, $r \in \left[0, \sum_{h=1}^{n} PM^{gen}_{n\times n\times n}(i-1, \pi^{gen,k}(i-1), h)\right)$.
if $r \in \left[0, PM^{gen}_{n\times n\times n}(i-1, \pi^{gen,k}(i-1), 1)\right)$ **then**
    $\pi_c \leftarrow 1$.
else
    for $t = 1$ *to* $n-1$ **do**
        if $r \in \left[\sum_{h=1}^{t} PM^{gen}_{n\times n\times n}(i-1, \pi^{gen,k}(i-1), h), \sum_{h=1}^{t+1} PM^{gen}_{n\times n\times n}(i-1, \pi^{gen,k}(i-1), h)\right)$ **then**
            $\pi_c \leftarrow t+1$, *break.*
        end if
    end for
end if
for $t = i$ *to* $n-1$ **do** *//avoid repeated selection of jobs.*
    for $j = 1$ *to* $n-1$ **do**
        $PM^{gen}_{n\times n\times n}(t, j, \pi_c) = 0$.
    end for
end for
return π_c.

Algorithm 5: $FPBSS(MC^{gen}_{n\times n\times n}, \boldsymbol{\pi}^{gen,k})$

Input: $\mathbf{PM}^{\mathbf{gen}}_{\mathbf{n\times n\times n}}$, $\boldsymbol{\pi}^{\mathbf{gen,k}}$.
Output: The first job $\pi^{gen,k}(1)$

Produce a random number r, *i.e.*, $r \in [0,1)$.
for $t = 1$ *to* $n-1$ **do**
  if $r \in \left[\sum_{h=1}^{t}\sum_{z=1}^{n}(MC^{gen}_{n\times n\times n}(1,h,z)/S^{gen}_{MC}(1)), \sum_{h=1}^{t+1}\sum_{z=1}^{n}(MC^{gen}_{n\times n\times n}(1,h,z)/S^{gen}_{MC}(1))\right)$ **then**
    $\pi^{gen,k}(1) = t+1$, *break*;
  end if
end for
for $t = 1$ *to* $n-1$ **do** *//avoid repeated selection of jobs*
  for $j = 1$ to $n-1$ **do**
    $MC^{gen}_{n\times n\times n}(t, j, \pi^{gen,k}(1)) = 0$.
  end for
end for

Algorithm 6: New Population Generation

Input: $\mathbf{PM}^{\mathbf{gen}}_{\mathbf{n\times n\times n}}$, $\mathbf{MC}^{\mathbf{gen}}_{\mathbf{n\times n\times n}}$, $\boldsymbol{\pi}^{\mathbf{gen,k}}$.
Output: **Pop(gen + 1)**

for $k = 1$ *to* *popsize* **do**
  for $i = 1$ *to* n **do**
    if $i = 1$ **then**
      $\pi_c \leftarrow FPBSS(\mathbf{MC}^{\mathbf{gen}}_{\mathbf{n\times n\times n}}, \boldsymbol{\pi}^{\mathbf{gen,k}})$.
    else
      $\pi_c \leftarrow SelectJob(\mathbf{PM}^{\mathbf{gen}}_{\mathbf{n\times n\times n}}, \boldsymbol{\pi}^{\mathbf{gen,k}}, i)$.
    end if
    $\pi^{gen,k}(i) = \pi_c$.
  end for
  $\mathbf{Pop(gen+1)} = \mathbf{Pop(gen+1)} \cup \boldsymbol{\pi}^{\mathbf{gen,k}}$, $\boldsymbol{\pi}^{\mathbf{gen,k}} \leftarrow \varnothing$.
end for
return **Pop(gen + 1)**.

4.2.3. Diversity controlling mechanism

As stated in Section 4.1, although the initial population is of high quality and well distribution, the diversity of the population may decline as the evolutionary process progresses. In order to maintain a reasonable balance between global exploration and local exploitation, a diversity control mechanism is provided to effectively prevent the proposed algorithm from prematurely converging to local optima. Let P_{div} be defined as the diversity index and δ as the diversity threshold. P_{div} can be used to measure the similarity between individuals in a population. If the value of P_{div} is less than δ, the top 20% of high-quality individuals in the population are retained and the remaining 80% are reinitialized. Specifically, 20% of the remaining 80% of individuals are formed by employing two constructive heuristics, *i.e.*, PFT_NEH(x) and PFZ_RZ(x), described in Subsections 4.1.1 and 4.1.2, while the others are randomly generated. The corresponding calculation steps are given below.

Step 1: According to Eqs. (20)-(23), $\boldsymbol{MC}^{\boldsymbol{gen}}_{\boldsymbol{n\times n\times n}}$ in the *gen*th generation is determined.

Step 2: Count the number of elements greater than 0 in the ith layer of $\boldsymbol{MC}^{\boldsymbol{gen}}_{\boldsymbol{n\times n\times n}}$, denoted as α_i.

Step 3: The value of diversity index P_{div} is calculated by using Eq. (27).

$$P_{div} = \frac{1}{n-1}\sum_{i=1}^{n-1}\frac{\alpha_i - 1}{\min(n^2 - n - 1, spsize - 1)} \tag{27}$$

Notice that the population size or subpopulation size often exceeds one. According to Eq. (27), the range of P_{div} is $[0,1]$. If the P_{div} value is closer to one, the better the diversity of the population. Conversely, the closer the value of P_{div} is to zero, the more similar the individuals in the

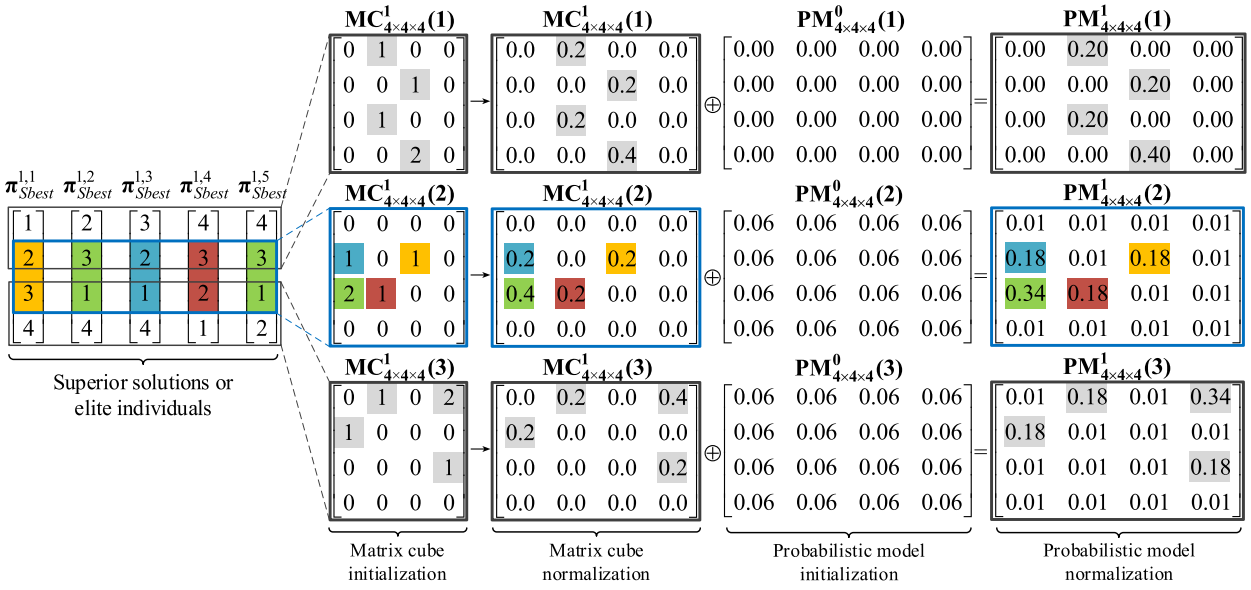

Fig. 2. Illustration of updating process of multi-dimensional probabilistic model.

population are. That is, the population has poor diversity if the individuals in the population have similar structural features or patterns. In order to reduce computational effort, the value of diversity index P_{div} is calculated every ten generations, and the diversity threshold δ is tuned in detail in the next section (see Subsection 5.2). To facilitate intuitive comprehension, the example of $\boldsymbol{MC}^1_{n\times n\times n}$ depicted in Fig. 2 is utilized to illustrate the diversity mechanism, while still considering those five high-quality individuals. It is evident from the preceding stages that $\alpha_1 = 4$, $\alpha_2 = 4$, $\alpha_3 = 4$, and the diversity index $P_{div} = (3/4 + 3/4 + 3/4)/3 = 3/4 = 0.75$ can be computed by using Eq. (27).

4.3. Local search controlled by multi-dimensional probabilistic model

4.3.1. Fast Insert-based neighbor evaluation method

It is well known that the insertion neighborhood structure is one of the most effective neighborhood structures for the permutation-based models of BFSPs (Pan & Wang, 2012; Schiavinotto & Stutzle, 2007;

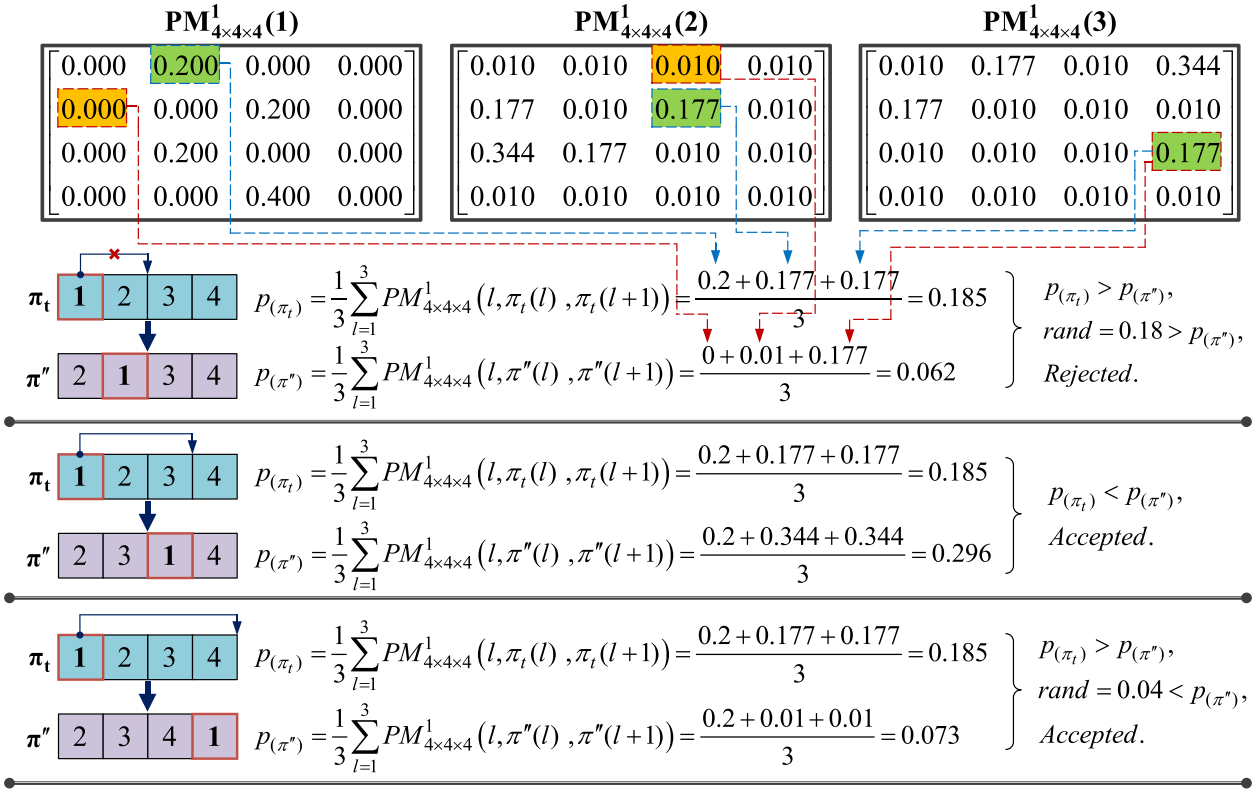

Fig. 3. An example of using the probability model to determine whether to evaluate π_t in $NS_{\text{PM\_boost}}$.

Tasgetiren, et al., 2017; Wang, et al., 2010). Therefore, it makes sense to select the insertion neighborhood to design neighborhood searches in the next Subsections 4.3.2 and 4.3.3. Since the number of neighbor solutions in the insertion neighborhood is $(n-1)^2$, the time complexity of such neighborhood search is $O(mn^3)$ by using forward and backward calculations given in Section 3. Following the reversibility of the permutation-based model for the BFSP_SDST stated in Section 3, a fast *Insert*-based neighbor evaluation method is devised by using a bidirectional calculation method. The devised fast evaluation method is beneficial for enhancing the efficiency of evaluating *Insert*-based neighbors. Its procedure is provided in Algorithm 7.

The pivotal point of the proposed fast *Insert*-based neighbor evaluation method is to remarkably reduce the complexity by appropriately adding the space storage. Specifically, $d_{\pi(i),j}$ and $f_{\pi(i),j}$ can be calculated and conserved at the beginning in Algorithm 7 (see Lines 1–2), and their numerical values can be treated as constants while evaluating *Insert*-based neighbors (see Lines 5–17). Although Algorithm 7 applies to the basic insertion neighborhood (see Lines 4 and 15), it can be easily extended to block-based insertion neighborhoods, *i.e.*, picking multiple jobs to form job blocks and performing neighborhood search based on those blocks. Therefore, both the job-based insertion neighborhood in Subsection 4.3.2 (see Lines 2–27 in Algorithm 8) and the block-based insertion neighborhood in Subsection 4.3.3 (see Lines 4–7 in Algorithm 10) can utilize the above-mentioned fast evaluation method to quickly calculate neighbors.

Algorithm 7: The fast Insert-based neighbor evaluation method

Input: p, q, and $\boldsymbol{\pi}$.

Output: $\boldsymbol{\pi}''$ and $C_{\max}(\boldsymbol{\pi}'')$.

1: Calculate departure time $d_{\pi(i),j}$ of job $\pi(i)$ by Eqs. (1)-(5), $i=1,2,...,n; j=1,2,...,m$.
2: Calculate duration time $f_{\pi(i),j}$ of job $\pi(i)$ by Eqs. (7)-(11), $i=n,n-1,...,1; j=m,m-1,...,1$.
3: **for** $p=1$ *to* n **do**
4: $\quad\boldsymbol{\pi}' \leftarrow$ Remove job $\pi(p)$ at the pth position from $\boldsymbol{\pi}$.
5: $\quad$**if** $i<p$ **then** *//calculate departure time.*
6: $\quad\quad d'_{\pi'(i),j}=d_{\pi(i),j}, i=1,2,...,n-1; j=1,2,...,m$.
7: $\quad$**else**
8: $\quad\quad$ Compute departure time $d'_{\pi'(i),j}$ by Eqs. (1)-(4).
9: $\quad$**end if**
10: $\quad$**if** $i>p$ **then** *//calculate duration time.*
11: $\quad\quad f'_{\pi'(i),j}=f_{\pi(i+1),j}, i=n-1,n-2,...,1; j=m,m-1,...,1$.
12: $\quad$**else**
13: $\quad\quad$ Compute duration time $f'_{\pi(i),j}$ by Eqs. (6)-(10).
14: $\quad$**end if**
15: $\quad\boldsymbol{\pi}'' \leftarrow$ Reinsert job $\pi(p)$ into the qth position of $\boldsymbol{\pi}'$, $q \in \{1,2,...,n\} \wedge q \notin \{p-1,p\}$.
16: $\quad$Compute departure time $d''_{\pi''(q),j}$ by the obtained $d'_{\pi'(q-1),j}$, $j=1,2,...,m$.
17: $\quad$Compute $C_{\max}(\boldsymbol{\pi}'')$ of new solution $\boldsymbol{\pi}''$ as follows:
$\quad\quad C_{\max}(\boldsymbol{\pi}'')=\max_{j=1,2,...,m}\{d''_{\pi''(q),j}+S_{\pi''(q),\pi'(q),j}+f'_{\pi'(q),j}\}$.
18: **end for**

4.3.2. Neighborhood search boosted by probabilistic model (NS_{PM_boost})

Since the multi-dimensional probabilistic model contains complete information of both the order relations and the block distributions of excellent individuals (see Subsection 4.2.1), it can be utilized to boost the neighborhood search. Thus, the *Insert*-based neighborhood search boosted by the probabilistic model, denoted as $NS_{\text{PM\_boost}}$, is proposed to perform fast exploitation.

The procedure of the $NS_{\text{PM\_boost}}$ is provided in Algorithm 8, where $\boldsymbol{\pi}_{best}$ is the best solution obtained so far. In Algorithm 8, when performing the $NS_{\text{PM\_boost}}$, the corresponding conditional probability is calculated via the probabilistic model (see Lines 7 and 14). If the conditional probability value acquired satisfies the predefined condition, the insertion operation is executed, and then the corresponding new neighbor is evaluated. Otherwise, a certain probability value is randomly generated to determine whether to perform the insertion operation (see Lines 9 and 16). It should be noted that the $NS_{\text{PM\_boost}}$ can reasonably utilize the structural patterns of excellent individuals to adjust the search scope in the *Insert*-based neighborhood, thereby avoiding the evaluations of potentially poor neighbors and improving the search efficiency. In order to facilitate a better understanding, Fig. 3 gives an example of using the probability model to determine whether to perform the insertion operation on the current neighbor $\boldsymbol{\pi}_t = [1, 2, 3, 4]$ in the $NS_{\text{PM\_boost}}$.

Algorithm 8: $PM\_NeighborSearch(\boldsymbol{\pi}_{best}, PM^{gen}_{n\times n\times n})$

Input: $\boldsymbol{\pi}_{\mathbf{best}}$, $\mathbf{PM}^{\mathbf{gen}}_{\mathbf{n\times n\times n}}$.
Output: The updated $\boldsymbol{\pi}_{\mathbf{best}}$ and $C_{\max}(\boldsymbol{\pi}_{\mathbf{best}})$.

1: $\boldsymbol{\pi}_{\mathbf{t}} = \boldsymbol{\pi}_{\mathbf{best}}$, $flag = true$.
2: **for** $k = 1$ *to* n **do**
3: $\quad\boldsymbol{\pi}' \leftarrow$ Remove job $\pi_t(k)$ at the kth position from $\boldsymbol{\pi}_{\mathbf{t}}$.
4: $\quad$**for** $i = 1$ *to* n **do**
5: $\quad\quad\boldsymbol{\pi}'' \leftarrow$ Reinsert the job $\pi_t(k)$ into the ith slot of $\boldsymbol{\pi}'$.
6: $\quad\quad$**if** $i < k$ **then** *//forward insertion*
7: $\quad\quad\quad$**if** $\sum_{l=i}^{k} PM^{gen}_{n\times n\times n}\left(l, \pi''(l), \pi''(l+1)\right) > \sum_{l=i}^{k} PM^{gen}_{n\times n\times n}\left(l, \pi_t(l), \pi_t(l+1)\right)$ **then**
8: $\quad\quad\quad\quad\boldsymbol{\pi}_{\mathbf{t}} \leftarrow \boldsymbol{\pi}''$.
9: $\quad\quad\quad$**else if** $rand() < 1/(k-i+1)\sum_{l=i}^{k} PM^{gen}_{n\times n\times n}\left(l, \pi''(l), \pi''(l+1)\right)$ **then**
10: $\quad\quad\quad\quad\boldsymbol{\pi}_{\mathbf{t}} \leftarrow \boldsymbol{\pi}''$. *//accept the solution by probability*
11: $\quad\quad\quad$**else** $flag = false$.
12: $\quad\quad\quad$**end if**
13: $\quad\quad$**else** *//backward insertion*
14: $\quad\quad\quad$**if** $\sum_{l=k}^{i} PM^{gen}_{n\times n\times n}\left(l, \pi''(l), \pi''(l+1)\right) > \sum_{l=k}^{i} PM^{gen}_{n\times n\times n}\left(l, \pi_t(l), \pi_t(l+1)\right)$ **then**
15: $\quad\quad\quad\quad\boldsymbol{\pi}_{\mathbf{t}} \leftarrow \boldsymbol{\pi}''$.
16: $\quad\quad\quad$**else if** $rand() < 1/(i-k+1)\sum_{l=k}^{i} PM^{gen}_{n\times n\times n}\left(l, \pi''(l), \pi''(l+1)\right)$ **then**
17: $\quad\quad\quad\quad\boldsymbol{\pi}_{\mathbf{t}} \leftarrow \boldsymbol{\pi}''$. *//accept the solution by probability*
18: $\quad\quad\quad$**else** $flag = false$.
19: $\quad\quad\quad$**end if**
20: $\quad\quad$**end if**
21: $\quad\quad$**if** $flag = true$ **then**
22: $\quad\quad\quad$**if** $C_{\max}(\boldsymbol{\pi}_{\mathbf{t}}) < C_{\max}(\boldsymbol{\pi}_{\mathbf{best}})$ **then**
23: $\quad\quad\quad\quad\boldsymbol{\pi}_{\mathbf{best}} = \boldsymbol{\pi}_{\mathbf{t}}$, $C_{\max}(\boldsymbol{\pi}_{\mathbf{best}}) = C_{\max}(\boldsymbol{\pi}_{\mathbf{t}})$.
24: $\quad\quad\quad$**end if**
25: $\quad\quad$**end if**
26: $\quad$**end for**
27: **end for**
28: **return** $\boldsymbol{\pi}_{\mathbf{best}}$, $C_{\max}(\boldsymbol{\pi}_{\mathbf{best}})$.

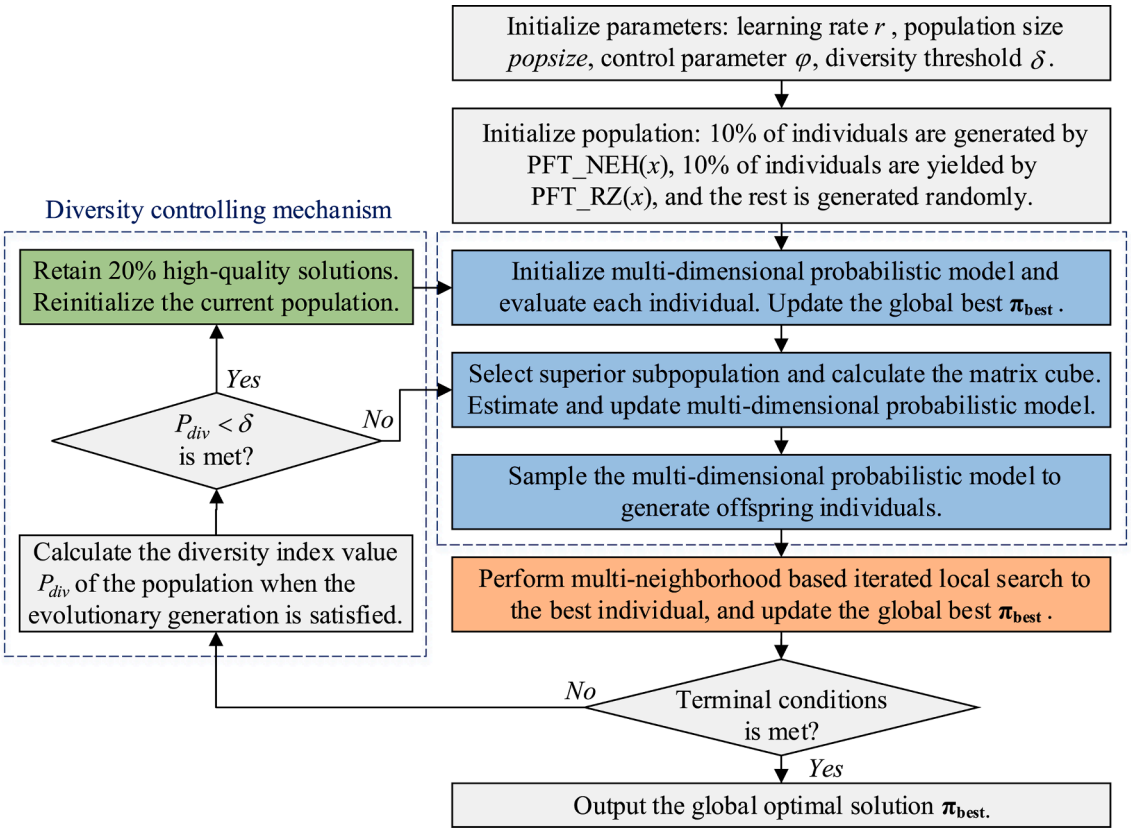

Fig. 4. The flow chart of the proposed MCEDA.

4.3.3. Neighborhood search guided by probabilistic model and reference sequence ($NS_{PM\_RS\_guide}$)

According to Subsection 4.2.1, the probability information related to the block distribution is saved in a series of position-based probability matrices, *i.e.*, $\boldsymbol{PM}^{gen}_{n\times n\times n}(\mathbf{1})$, $\boldsymbol{PM}^{gen}_{n\times n\times n}(\mathbf{2})$,…,$\boldsymbol{PM}^{gen}_{n\times n\times n}(\boldsymbol{n})$. Therefore, the *Insert*-based neighborhood search guided by the probabilistic model and the reference sequence, denoted as $NS_{PM\_RS\_guide}$, is designed to execute deep exploitation.

Let p_r be the correlation coefficient of jobs. p_r is calculated according to the following Eq. (28):

$$p_r = \frac{\sum_{x=1}^{n-1} PM^{gen}_{n\times n\times n}(x, \pi^{gen}(l), \pi^{gen}(l'))}{\sum_{x=1}^{n-1}\sum_{z=1}^{n-1} PM^{gen}_{n\times n\times n}(x, \pi^{gen}(l), z)} \tag{28}$$

where $l \neq l'$ and $l, l' = 1, 2, \ldots, n$. When $p_r \geqslant 0.2$, it indicates that two jobs $\pi^{gen,k}(i)$ and $\pi^{gen,k}(i')$ in $\boldsymbol{\pi}^{gen,k}$ are strongly correlated. Obviously, p_r represents the tightness of the connection between the job $\pi^{gen}(l)$ and the job $\pi^{gen}(l')$ in $\boldsymbol{\pi}^{gen}$. Firstly, a block construction strategy is presented in Algorithm 9, where the block $\boldsymbol{\pi}_{block}$ is extracted from the current individual or solution sequence $\boldsymbol{\pi}^{gen}$ via utilizing the reference sequence $\boldsymbol{\pi}_{best}$ (*i.e.*, the best solution obtained so far) and the probability model $\boldsymbol{PM}^{gen}_{n\times n\times n}$. Then, the procedure of the $NS_{PM\_RS\_guide}$ is outlined in Algorithm 10, where $\boldsymbol{\pi}_{left}$ is a partial sequence after removing $\boldsymbol{\pi}_{block}$ from $\boldsymbol{\pi}^{gen}_{best}$, and $Insert(\boldsymbol{\pi}_{left}, i, \boldsymbol{\pi}_{block})$ means that $\boldsymbol{\pi}_{block}$ is inserted in the ith position in $\boldsymbol{\pi}_{left}$.

From Algorithm 10, it can be seen that the core idea of the $NS_{PM\_RS\_guide}$ is to dynamically construct blocks with strong correlation and promising pattern in the process of neighborhood search (see Line 7), and then search the *Insert*-based neighborhood determined by each block (see Lines 11–17). Since the constructed blocks are different in most cases, the $NS_{PM\_RS\_guide}$ can perform rich searches in a variety of *Insert*-based neighborhoods, which is conducive to increasing the search depth. Moreover, the $NS_{PM\_RS\_guide}$ can ensure that the overall quality of neighbors in the constructed neighborhood is high, which helps to improve the search quality.

Algorithm 9: $Variable\_Block(k, pos, len, \boldsymbol{\pi}^{gen}, \boldsymbol{\pi}_{best}, PM^{gen}_{n\times n\times n})$

Input: k, pos, len, $\boldsymbol{\pi}^{\mathbf{gen}}$, $\boldsymbol{\pi}_{\mathbf{best}}$, and $\mathbf{PM}^{\mathbf{gen}}_{\mathbf{n\times n\times n}}$.
Output: len and $\boldsymbol{\pi}_{\mathbf{block}}$.

while $\pi^{gen}(k+len)=\pi_{best}(pos+len)$ **do**
  Calculate p_r for $\pi^{gen}(k+len-1)$ and $\pi^{gen}(k+len)$.
  if $(p_r \geq 0.2)$ *or* $((p_r < 0.2)$ *and* $(rand() < 0.35))$ **then**
    $\pi_{block}(len) \leftarrow \pi^{gen}(k+len)$.
  else
    break.
  end if
  $len = len + 1$.
end while
return len and $\boldsymbol{\pi}_{\mathbf{block}}$.

Algorithm 10: $Reference\_NeighborSearch(\boldsymbol{\pi}^{gen}, \boldsymbol{\pi}_{best}, PM^{gen}_{n\times n\times n})$

Input: $\boldsymbol{\pi}_{\mathbf{best}}$, $\boldsymbol{\pi}^{\mathbf{gen}}$ and $\mathbf{PM}^{\mathbf{gen}}_{\mathbf{n\times n\times n}}$.
Output: The updated $\boldsymbol{\pi}^{\mathbf{gen}}$ and $C_{\max}(\boldsymbol{\pi}^{\mathbf{gen}})$.

$N_c = 1$.
while $N_c \leq n$ **do**
  $k=1$, $len=1$, $pos=N_c$, $\boldsymbol{\pi}_{\mathbf{block}} \leftarrow \varnothing$.
  while $\pi^{gen}(k) \neq \pi_{best}(pos)$ **do**
    $k = k+1$.
  end while
  $\boldsymbol{\pi}_{\mathbf{block}} \leftarrow Variable\_Block(k, pos, len, \boldsymbol{\pi}^{\mathbf{gen}}, \boldsymbol{\pi}_{\mathbf{best}}, \mathbf{PM}^{\mathbf{gen}}_{\mathbf{n\times n\times n}})$. //Algorithm 9
  if $len = 1$ **then**
    $\pi_{block}(1) = \pi_{best}(pos)$.
  end if
  $\boldsymbol{\pi}_{\mathbf{left}} \leftarrow$ Remove job block $\boldsymbol{\pi}_{\mathbf{block}}$ from $\boldsymbol{\pi}^{\mathbf{gen}}$.
  for $i = 1$ *to* $(n - len + 1)$ **do**
    $\boldsymbol{\pi}_{\mathbf{new}} \leftarrow Insert(\boldsymbol{\pi}_{\mathbf{left}}, i, \boldsymbol{\pi}_{\mathbf{block}})$.
    if $C_{\max}(\boldsymbol{\pi}_{\mathbf{new}}) < C_{\max}(\boldsymbol{\pi}^{\mathbf{gen}})$ **then**
      $\boldsymbol{\pi}^{\mathbf{gen}} = \boldsymbol{\pi}_{\mathbf{new}}$, $C_{\max}(\boldsymbol{\pi}^{\mathbf{gen}}) = C_{\max}(\boldsymbol{\pi}_{\mathbf{new}})$, $N_c = 1$.
    end if
  end for
  $N_c = N_c + 1$.
end while
return $\boldsymbol{\pi}^{\mathbf{gen}}$, $C_{\max}(\boldsymbol{\pi}^{\mathbf{gen}})$.

4.3.4. Multi-neighborhood iterated local search

In the last decade, various local search methods based on the VNS framework have been proposed, among which iterated local search (ILS) proposed by Lourenco et al. (Lourenço, et al., 2010) is one of the most effective local search methods. The main idea of ILS is to first perturb the current best solution for preventing cycle search and jumping out of local optima, and then to undertake an iterative variable neighborhood search to find more satisfied solutions. Nowadays, ILS has been widely applied to solve a variety of scheduling problems. Therefore, a new multi-neighborhood ILS (MNILS) combing the $NS_{\text{PM\_boost}}$ (Algorithm 8) and the $NS_{\text{PM\_RS\_guide}}$ (Algorithm 10) is devised to perform deeper exploitation from promising regions obtained by the global search in Section 4.2.

The procedure of the MNILS is given in Algorithm 11, where $Interchange(\pi_{t1}^{gen}, u, v)$ means interchange the uth job and the vth job in π_{t1}^{gen}, π_{best}^{gen} is the current best individual or sequence at generation gen, and T is the temperature control parameter set to $T = \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}/5nm$ in the proposed MCEDA. From Algorithm 11, it can be known that the MNILS starts the exploitation from the promising regions (*i.e.*, π_{best} and π_{best}^{gen}), and iteratively executes the $NS_{\text{PM\_boost}}$ (see Lines 1 and 6) and the $NS_{\text{PM\_RS\_guide}}$ (see Lines 2 and 7) to guide the exploitation down to the optimal or near optimal solution. Moreover, the hybrid perturbation strategy combining *Interchange*-based moves (see Lines 4 and 5) and simulated annealing mechanism (see Line 13) is used to drive the local search to jump out of local optima.

4.4. The framework of MCEDA

In general, EDA reproduces offspring by sampling from a well-designed probabilistic model, which mainly consists of the following five steps (Pan & Ruiz, 2012): (*a*) generating initial population, (*b*) selecting elite individuals, (*c*) updating probability model with superior solutions, (*d*) sampling from the probability model to create a new population, and (*e*) repeating steps (*b*)-(*d*) until the termination condition is satisfied. After covering each component in detail in the preceding sections, Fig. 4 illustrates the MCEDA framework.

From Fig. 4, it can be seen that the proposed MCEDA consists of two main aspects, *i.e.*, a breadth global search and a depth local search. First, two effective constructive heuristics are applied to generate some high-quality initial individuals. Second, an effective EDA-based global search is adopted to estimate the distribution of excellent individuals or solutions, which is beneficial to quickly guide the exploration to discover the promising regions in solution space. Third, a multi-neighborhood iterated local search with a fast *Insert*-based neighbor evaluation method is devised to conduct in-depth exploitation in the promising regions found by global search. Since both global exploration and local exploitation are well stressed, it is expected that the proposed MCEDA can achieve good performance in solving BFSP_SDST.

4.5. Computational complexity analysis

According to Fig. 4, the computational complexity (CC) of the pri-

Algorithm 11: Multi-neighborhood iterated local search (MNILS)

Input: π_{best}, π^{gen} and $\mathbf{PM}_{\mathbf{n\times n\times n}}^{\text{gen}}$.
Output: π_{best} and $C_{\max}(\pi_{\text{best}})$.

1: $\pi_{\text{best}} \leftarrow PM\_NeighborSearch(\pi^{\text{gen}}, \pi_{\text{best}}, \mathbf{PM}_{\mathbf{n\times n\times n}}^{\text{gen}})$. //Algorithm 8
2: $\pi_{\text{t1}}^{\text{gen}} \leftarrow Reference\_NeighborSearch(\pi^{\text{gen}}, \pi_{\text{best}}, \mathbf{PM}_{\mathbf{n\times n\times n}}^{\text{gen}})$. //Algorithm 10
3: **while** the termination condition not met **do**
4: Randomly generate u and v and meet $|u-v| > \lceil n/3 \rceil$, Perform $\lceil n/10 \rceil$ times interchange operation on $\pi_{\text{t1}}^{\text{gen}}$.
5: $\pi_{\text{t2}}^{\text{gen}} \leftarrow Interchange(\pi_{\text{t1}}^{\text{gen}}, u, v)$. //*perturbation phase*
6: $\pi_{\text{best}} \leftarrow PM\_NeighborSearch(\pi_{\text{t2}}^{\text{gen}}, \pi_{\text{best}}, \mathbf{PM}_{\mathbf{n\times n\times n}}^{\text{gen}})$. //Algorithm 8
7: $\pi_{\text{t3}}^{\text{gen}} \leftarrow Reference\_NeighborSearch(\pi_{\text{t2}}^{\text{gen}}, \pi_{\text{best}}, \mathbf{PM}_{\mathbf{n\times n\times n}}^{\text{gen}})$. //Algorithm 10
8: **if** $C_{\max}(\pi_{\text{t3}}^{\text{gen}}) < C_{\max}(\pi_{\text{t1}}^{\text{gen}})$ **then**
9: $\pi_{\text{t1}}^{\text{gen}} = \pi_{\text{t3}}^{\text{gen}}$.
10: **if** $C_{\max}(\pi_{\text{t1}}^{\text{gen}}) < C_{\max}(\pi_{\text{best}})$ **then**
11: $\pi_{\text{best}} = \pi_{\text{t1}}^{\text{gen}}$.
12: **end if**
13: **else if** $rand() \le \exp\{-(C_{\max}(\pi_{\text{t3}}^{\text{gen}}) - C_{\max}(\pi_{\text{t1}}^{\text{gen}}))/T\}$ **then**
14: $\pi_{\text{t1}}^{\text{gen}} = \pi_{\text{t3}}^{\text{gen}}$. //*accept the solution by probability*
15: **end if**
16: **end while**
17: **return** π_{best} and $C_{\max}(\pi_{\text{best}})$.

mary parts of the proposed MCEDA is analyzed as follows. At the initialization phase, since the CC of determining makespan by the forward calculation in Eqs. (1)-(5) or by the backward calculation in Eqs. (7)-(12) is $O(mn)$, the CC of evaluating the population is $O(popsize \times mn)$. In Algorithm 2, the CCs of calculating $I(i)$ and σ_i are $O(mn)$, and the CCs of Line 2, Line 5, Lines 7–10, and Lines 3–12 are $O(n\log n)$, $O(mn^2)$, $O(mn^3)$, and $O(xmn^3)$, respectively. Hence, the CC of Algorithm 2 is $O(xmn^3)$. In Algorithm 3, the CC of computing $I_Z(i)$ is $O(mn)$, and the CCs of Lines 2–3, Line 6, Lines 7–13, and Lines 4–15 are $O(n\log n)$, $O(mn^2)$, $O(mn^3)$, and $O(xmn^3)$, respectively. Hence, the CC of Algorithm 3 is $O(xmn^3)$. Since the fast *Insert*-based neighbor evaluation method given in Subsection 4.3.1 is used to reduce the CC of evaluating solution from $O(mn)$ to $O(m)$, the CCs of Algorithms 2 and 3 are also reduced from $O(xmn^3)$ to $O(xmn^2)$. In Subsection 4.2.1, the CC of calculating $\boldsymbol{MC}^{gen}_{n\times n\times n}$ is $O(n^3)$ when using Eqs. (20)-(23) and the CC of initializing $\boldsymbol{PM}^{0}_{n\times n\times n}$ is $O(n^3)$ when using Eqs. (24)-(26).

At the iterative process, since *spsize* superior solutions are selected from $\boldsymbol{Pop}(\boldsymbol{gen})$ to calculate $\boldsymbol{MC}^{gen}_{n\times n\times n}$, the CC of sorting the population via quick sort method is $O(popsize \times \log popsize)$. In Algorithm 4, the CC of Line 1 is $O(n)$ and that of Lines 2–10 is also $O(n)$. So, the CC of Algorithm 4 is $O(n)$. In Algorithm 5, the CC of Lines 2–6 is $O(n)$. Thus, the CC of Algorithm 5 is $O(n)$. In Algorithm 6, the CCs of Lines 2–9 and Lines 1–11 are $O(n^2)$ and $O(popsize \times n^2)$, respectively. So, the CC of generating a new population by sampling $\boldsymbol{PM}^{gen}_{n\times n\times n}$ in Subsection 4.2.2 is $O(popsize \times n^2)$. Note that the CC of updating $\boldsymbol{PM}^{gen}_{n\times n\times n}$ is also $O(n^3)$ according to Eq. (26). Moreover, in Subsection 4.2.3, the CCs of calculating diversity index value P_{div} and that of reinitializing part of the individuals in the population are nearly $O(n)$ and $O(popsize \times n^2)$, respectively. In Algorithm 8, the CCs of Lines 7–12, Lines 14–19, Lines 4–26, and Lines 2–27 are nearly $O(n)$, $O(n)$, $O(mn^2)$, and $O(mn^3)$, respectively. In Algorithm 10, the CCs of Line 7, Lines 12–17, and Lines 2–19 are nearly $O(len)$, $O(mn^2)$, and $O(mn^3)$. In Algorithm 11, the CCs of Line 1, Line 2, and Lines 3–16 are all approximately equal to $O(mn^3)$. Since the fast *Insert*-based neighbor evaluation method is used to calculate neighbor solutions and the probabilistic model is employed to guide neighborhood searches, the CC of conducting MNILS in Algorithm 11 is estimated to be $O(mn\widetilde{\log}n)$, where $O(\widetilde{\log}n)$ is less than linear time $O(n)$.

Let T_{CC} be the total CC of MCEDA, and K_1^{gen} (K_2^{gen}) the repeat times of Algorithms 8 (Algorithms 10) at generation *gen* for a given instance. Then, denote $K_1 = \sum_{gen=1}^{maxgen} K_1^{gen}$ and $K_2 = \sum_{gen=1}^{maxgen} K_2^{gen}$. Since K_1^{gen} and K_2^{gen} are no less than one, we have $K_1, K_2 \geqslant maxgen$. In general, n is larger than m and *len*. According to the above analysis, T_{CC} can be expressed as.

$$\begin{aligned} T_{CC} &= O(maxgen \times (xmn^2 + mn^2 + n^3 + popsize \times n^2) \\ &\quad + K_1 \times mn\widetilde{\log}n + K_2 \times mn\widetilde{\log}n) \\ &= O\left(maxgen \times \left(n^3 + popsize \times n^2\right) + \overline{K} \times mn\widetilde{\log}n\right), \end{aligned} \tag{29}$$

where $\overline{K}$ is the average repeat times of executing Algorithms 8 and 10. From Eq. (29), it can be observed that the CC of MCEDA is acceptable because the highest degree in $(maxgen \times (n^3 + popsize \times n^2) + \overline{K} \times mn\widetilde{\log}n)$ is three.

5. Experimental results and statistical analysis

This section implements the extensive experiments to demonstrate the effectiveness and efficiency of the proposed MCEDA. Firstly, the experimental setup is briefly described in Section 5.1, including the testing instances, performance metrics, and experimental environment. Then, the effects of MCEDA's parameters are discussed in Section 5.2. Afterwards, the superiority of multi-dimensional probabilistic model and the advantages of improvement strategies are investigated in Section 5.3 and Section 5.4, respectively. Finally, computational comparisons and statistical analysis of MCEDA against several state-of-the-art algorithms are conducted and discussed.

5.1. Experimental setup

In order to investigate the performance of the proposed MCEDA, a set of well-known benchmark datasets provided by Ruiz, et al. (2005) for PFSP_SDST are employed as test sets, which are available at https://soa.iti.es/. These test sets contain a total of 480 instances with different sizes, which can be divided into four subsets according to different setup times, namely SSD-10, SSD-50, SSD-100, and SSD-125. Each subset consists of 120 different instances, ranging from 20 jobs and 5 machines to 500 jobs and 20 machines. The processing time of each job is randomly generated in the uniformly distributed interval $[1, 99]$. The setup times in each test subset SSD-K (K = 10, 50, 100 and 125) are randomly generated in the uniformly distributed interval $[1, K-1]$. The comparison algorithms are all conducted in the same programming environment and computer configuration. All algorithms are coded in Pascal language, compiled by Embarcadero Rad Studio (XE8), and executed independently on a PC equipped with Inter(R) Core(TM) i7-8700 M @ 3.2 GHz processor and 32 GB of RAM memory under Windows 7 OS. It should be noted that all algorithms have the same termination condition, *i.e.*, the maximum elapsed CPU time of $60nm$ milliseconds. Moreover, to fairly derive reliable computation results in the same time, each algorithm for each specific instance is performed 30 times independently. Therefore, a total number of 14,400 results are available for each algorithm, and the computational comparisons are completely fair and comparable. In order to evaluate the performance of the algorithms, the average relative percent deviation (ARPD) is used to measure the average relative quality of the experimental results, as stated by Eq. (30):

$$ARPD = \frac{1}{R}\sum_{i=1}^{R}\left(\frac{C_i - C_{opt}}{C_{opt}}\right) \times 100\% \tag{30}$$

where R is the number of runs. C_i is the makespan obtained by a specific algorithm in the ith experiment for a given instance. C_{opt} is the optimal makespan for that instance. Since few algorithms are devised to solve the problem under consideration, the minimum makespan found by all algorithms is selected as C_{opt}. For the calibration of the algorithm parameters, C_{opt} is the best makespan found by all configurations for the calibration instance. It is obvious that the smaller the value of *ARPD*, the better the performance of the algorithm. In the statistical table of the experimental results, the best values obtained are highlighted in bold font, the second-best values are indicated in bold and underlined font, and the third-best values are marked in italic and underlined font.

5.2. Parameter calibration

Parameter calibration has an important impact on the efficacy and efficiency of HIOAs. As stated in Section 4, the proposed MCEDA contains four controllable parameters, *i.e.*, population size (*popsize*), proportion of superior solutions (φ), learning rate (r), and diversity threshold (δ). In order to calibrate these parameters, the Design of Experiments (DOE) approach (Montgomery, 2008) is employed to provide proper parameters of MCEDA. To further investigate the sensitivity and interaction of parameters, all of the obtained experiment results are analyzed by the multi-factor Analysis of Variance (ANOVA) technique, which has been widely used in the scheduling literature (Shao, et al., 2018a, 2018b). According to the research in recent years (Shao, et al., 2018b), if the algorithm's parameters are calibrated by using the same instances (see Section 5.1) that will later be used for comparison, the calibrated parameters may over fit (Shao, et al., 2018b). Thus, the additional subsets (*i.e.*, ASSD-10, ASSD-50, ASSD-100 and ASSD-125) are generated for parameter calibration. The instances in each ASSD-K (K = 10, 50, 100 and 125) are generated in the same way as those in the corresponding SSD-K in Section 5.1, and each ASSD-K is half the size of the SSD-K. That is, there are a total of 240 instances. Moreover, since the range of parameter values is more flexible, it is required to restrict the

Table 3
The level of each parameter for MCEDA.

Parameter	Factor level			
	1	2	3	4
popsize	50	100	150	200
φ	0.1	0.2	0.3	0.4
r	0.1	0.2	0.3	0.4
δ	0.2	0.3	0.4	0.5

Table 4
Results of ANOVA for MCEDA's parameters.

Source	Sum of squares	Degrees of freedom	Mean square	*F*-ratio	*p*-value
Main effects					
popsize	0.016	3	0.005	537.00	**0.000**
φ	0.010	3	0.003	346.13	**0.000**
r	0.016	3	0.005	538.97	**0.000**
δ	0.008	3	0.003	272.97	**0.000**
Interactions					
*popsize**φ	0.033	15	0.002	215.487	**0.000**
*popsize**r	0.008	12	0.001	68.555	**0.000**
*popsize**δ	0.010	12	0.001	86.743	**0.000**
φ*r	0.000	9	0.000	0.039	1.000
φ*δ	0.000	9	0.000	0.590	0.804
r*δ	0.000	9	0.000	0.108	0.999
Residual	0.002	189	0.000		
Total	0.053	255			

selection range of parameters. The reasonable range for each parameter is determined according to the previous literature (Pan & Ruiz, 2012; Zhang, et al., 2021; Zhang, et al., 2022) and our preliminary experiments. Following that, the multiple potential levels (values) for each factor (parameter) are determined by trial and error.

The levels of each parameter are listed in Table 3. The full factorial experimental design is conducted for the proposed MCEDA with $4 \times 4 \times 4 \times 4 = 256$ distinct configurations. MCEDA is repeated 30 times with a running time of $60nm$ milliseconds on each instance. As a result, a total of $256 \times 30 \times 240 = 1843200$ results are obtained. In consequence, if the test program runs as a whole single process program. it needs at least 280 CPU days to obtain the entire experimental results. Fortunately, due to the multi-core architecture in our personal computers, the test program was divided into different sub-programs, which were arranged to run on different cores. So, it actually took about 12.5 days to complete the calibration.

The parameter is regarded as the controller factor, and the average *ARPD* value is regarded as the response variable (RV). Obviously, the lower the RV value is, the better the performance is. Moreover, three major hypotheses (*i.e.*, normality, homogeneity of variance, and independence of residuals) are checked before ANOVA is conducted. The checked results reveal that no significant deviations are found, so these hypotheses can be accepted. Note that the *F*-ratio is a strong signal of significance when the *p*-value is less than the confidence level. The larger the *F*-ratio is, the greater the effect of the factor on the RV is. The ANOVA results are reported in Table 4. The main effects plot for all parameters is shown in Fig. 5.

It is clearly observed from Table 4 that four parameters *popsize*, φ, r and δ are statistically significant since their *p*-values are smaller than $\alpha = 0.05$ (α denotes the confidence level). The parameter *popsize* achieves the largest *F*-ratio, indicating that the population size has the most significant effect on the performance of the proposed MCEDA. As can be seen in Fig. 5, the choice of $popsize = 100$ yields the best result, while $popsize = 200$ obtains the worst result. It suggests that a medium-scale population is advantageous to maintain a proper search scope in solution space and ensure a certain search efficiency. The second largest *F*-ratio value corresponds to the factor φ. As also can be seen in Fig. 5, the value $\varphi = 0.2$ can achieve the best performance, while $\varphi = 0.1$ and $\varphi = 0.4$ yield worse results. It is obvious that the proportion of superior solutions has a significant effect on the probabilistic model's ability of accumulating the valuable information of promising patterns in high-quality subpopulation. The third significant factor is parameter r. It is clear from Fig. 5 that too small or too large learning rate r may degrade algorithm performance, and $r = 0.3$ is a suitable choice. If the value of r is set too high, the algorithm may converge prematurely, otherwise it may lead to slow convergence. Although the diversity threshold δ has the least impact on the algorithm's effectiveness, a lack of population diversity directly results in search stagnation. So, an appropriate diversity threshold still favors MCEDA in suitably switching between exploration and exploitation. Fig. 5 reveals that the proper value of δ is 0.3.

Although the main effects in Fig. 5 show the best choice of each single parameter, the analysis on single parameter is incomplete if there are significant interactions between parameters (Tasgetiren, et al., 2017). Thus, the two-level interactions between the involved parameters are also investigated, and the relevant results are reported in Table 4. It is observed from Table 4 that the interactions of three parameter pairs (*i.e.*, *popsize**φ, *popsize**r, and *popsize**δ) are statistically significant since their *p*-values are less than 0.05. The interaction effect plots of these parameter pairs are depicted in Fig. 6. From Fig. 6, it can be seen that all the interactions of *popsize**φ, *popsize**r, and *popsize**δ are weak and coincide with the conclusions drawn from Fig. 5. Based on the above analyses, the parameters of MCEDA are set as: $popsize = 100$, $\varphi = 0.2$, $r = 0.3$, $\delta = 0.3$.

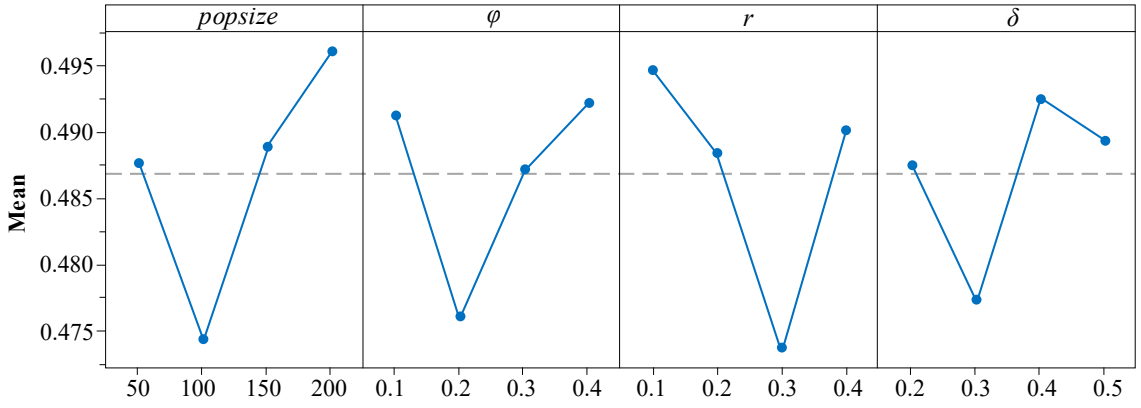

Fig. 5. Main effect plots of parameters.

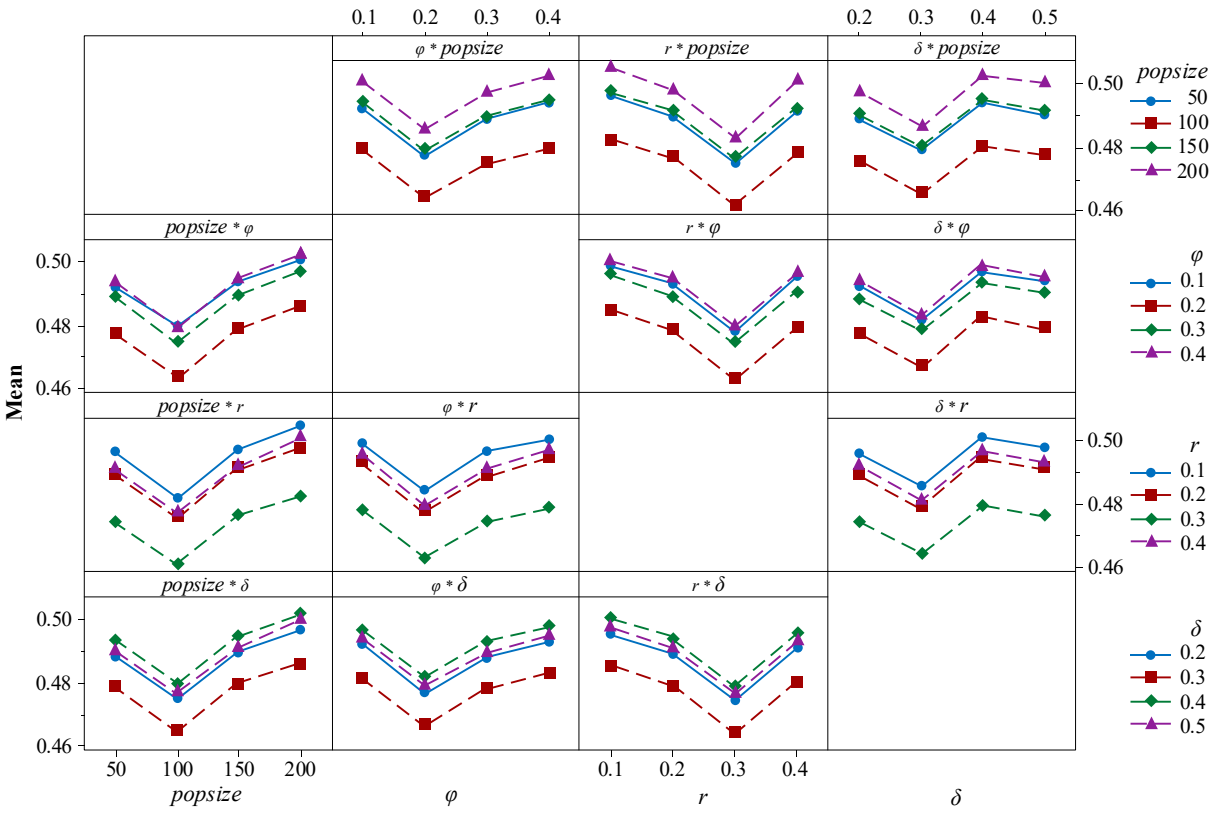

Fig. 6. Interaction effect plots of parameter pairs.

5.3. Performance analysis of improvement strategies

As stated in Section 4, there are four important improvement strategies contributing to improving the performance of our presented MCEDA: (1) the problem's characteristics based two constructive heuristics developed in Subsections 4.1.1 and 4.1.2; (2) the diversity controlling mechanism described in Subsection 4.2.3; (3) the fast *Insert*-based neighbor evaluation method provided in Subsection 4.3.1; and (4) the multi-neighborhood local search controlled by multi-dimensional probabilistic model referred in Subsections 4.3.2–4.3.4. In order to analyze the performance of these strategies, in this section, six variants of MCEDA are implemented to investigate and validate their contributions. Firstly, to evaluate the effectiveness of the initialization method, we utilize the random initialization method to replace the original one, yielding a variant named MCEDA$_{v1}$. In MCEDA$_{v1}$, the initial population is randomly generated without using constructive heuristics or scheduling rules. Secondly, to validate the proposed diversity controlling mechanism, the diversity controlling mechanism is removed from MCEDA and a variant algorithm named MCEDA$_{v2}$ is developed, in which the population is never reinitialized throughout the execution of MCEDA$_{v2}$. Thirdly, to determine if the proposed fast evaluation method promotes the efficiency of local search, we adjust the neighbor calculation method in local search to produce a variant that does not employ the fast *Insert*-based neighbor evaluation method, designated as MCEDA$_{v3}$. Finally, to examine the effectiveness of local search controlled by the multi-dimensional probabilistic model, we implement three variants of MCEDA, including MCEDA without $NS_{\text{PM\_boost}}$ (denoted as MCEDA$_{v4}$), MCEDA without $NS_{\text{PM\_RS\_guide}}$ (denoted as MCEDA$_{v5}$), and MCEDA without MNILS (denoted as MCEDA$_{v6}$). Note that for the first two variants, *i.e.*, MCEDA$_{v4}$ and MCEDA$_{v5}$, MNILS is implemented based only on a single neighborhood search strategy, *i.e.*, $NS_{\text{PM\_boost}}$ or $NS_{\text{PM\_RS\_guide}}$. These two variants are adopted to certify that the proposed two neighborhood search strategies are essential for local search. The last one is used to verify the vital role of the devised MNILS for the proposed MCEDA.

To sum up, a total of six MCEDA's variants are created to demonstrate the effectiveness of these improvement strategies. The controlled experiments are conducted in such a way that each variant modifies a single component of the complete MCEDA. To guarantee a fair comparison, the probabilistic model update mechanism and sampling strategy remain the same in MCEDA and its variants, and the parameters of the above algorithms are also the same. The MCEDA and all variants adopt the same $60nm$ millisecond elapsed CPU time, and they are tested by running 30 times independently on each instance. The benchmark instances introduced in Section 5.1 are employed as the testbed. The statistical results obtained by computational comparisons are reported in Table 5, grouped by each scenario and per number of jobs.

As shown in Table 5, the proposed MCEDA outperforms the other variants over a variety of scale instances. The results obtained by MCEDA are remarkably better in terms of both the average relative percent deviation (*ARPD*) and the standard deviation (*SD*), indicating that these improvement strategies contribute considerably to improving the performance of MCEDA. Specifically, MCEDA yields much better *ARPD* values than MCEDA$_{v1}$ for different scales and scenarios, implying that both constructive heuristics affect algorithm's performance, especially for the large-scale instances. The two constructive heuristics (*i.e.*, PFT_NEH(x) and PFZ_RZ(x)) utilize problem properties to produce partially promising solutions, which can provide better starting points for subsequent searches and notably narrow the search scope. That is, MCEDA is afforded more opportunities to find promising search regions within the reduced search space. Moreover, the results of MCEDA$_{v2}$ is slightly weaker than MCEDA, indicating that the proposed diversity mechanism not only ensures the vitality of the search to avoid stagnation, but also preserves the population diversity and evolutionary information. As can be observed from Table 5, MCEDA$_{v3}$ is somewhat inferior to MCEDA in all scenarios, which demonstrates the effectiveness

Table 5
Comparison results of MCEDA with its six variants.

Scenario	*n*	MCEDA$_{v1}$		MCEDA$_{v2}$		MCEDA$_{v3}$		MCEDA$_{v4}$		MCEDA$_{v5}$		MCEDA$_{v6}$		MCEDA	
		ARPD	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*
SSD-0	20	0.027	0.296	**0.017**	**0.286**	*0.023*	*0.293*	0.193	0.332	0.182	0.361	0.334	0.557	**0.013**	**0.284**
	50	*1.154*	0.288	**0.843**	**0.263**	1.224	*0.282*	1.773	0.341	1.569	0.355	2.532	0.584	**0.357**	**0.257**
	100	*1.108*	0.282	**0.885**	**0.257**	1.182	*0.271*	1.554	0.336	1.375	0.343	2.375	0.611	**0.383**	**0.248**
	200	**0.579**	*0.264*	*0.712*	**0.253**	0.761	*0.264*	1.141	0.363	0.973	0.376	1.794	0.573	**0.362**	**0.245**
	500	**0.475**	*0.323*	*0.555*	**0.295**	0.637	**0.295**	1.773	0.425	0.825	0.443	1.483	0.607	**0.188**	**0.283**
Average		*0.669*	0.291	**0.602**	**0.271**	0.765	*0.281*	1.287	0.359	0.985	0.376	1.704	0.586	**0.261**	**0.263**
SSD-10	20	*0.021*	0.291	**0.015**	**0.281**	*0.021*	*0.286*	0.186	0.343	0.176	0.363	0.323	0.561	**0.011**	**0.279**
	50	*1.027*	0.298	**0.736**	**0.268**	1.207	*0.291*	1.652	0.347	1.464	0.367	2.434	0.578	**0.243**	**0.265**
	100	*1.054*	0.282	**0.827**	**0.262**	1.118	*0.268*	1.478	0.344	1.262	0.355	2.324	0.597	**0.335**	**0.257**
	200	**0.533**	0.279	*0.652*	**0.253**	0.667	*0.272*	1.034	0.371	0.918	0.368	1.732	0.566	**0.328**	**0.246**
	500	**0.436**	0.323	*0.517*	**0.284**	0.541	*0.314*	1.766	0.432	0.775	0.437	1.426	0.612	**0.173**	**0.258**
Average		*0.614*	0.295	**0.549**	**0.270**	0.711	*0.286*	1.223	0.367	0.919	0.378	1.648	0.583	**0.218**	**0.261**
SSD-50	20	0.026	*0.288*	**0.018**	**0.283**	*0.024*	**0.282**	0.193	0.345	0.188	0.364	0.338	0.553	**0.014**	**0.271**
	50	*1.112*	0.296	**0.823**	**0.264**	1.218	*0.286*	1.742	0.332	1.543	0.373	2.524	0.581	**0.345**	**0.254**
	100	*1.027*	0.278	**0.746**	**0.258**	1.043	*0.272*	1.424	0.347	1.215	0.356	2.257	0.594	**0.287**	**0.243**
	200	**0.436**	0.267	*0.612*	**0.255**	0.634	*0.261*	1.007	0.355	0.826	0.372	1.671	0.568	**0.243**	**0.246**
	500	**0.413**	0.324	*0.523*	**0.276**	0.535	*0.295*	1.731	0.428	0.796	0.445	1.433	0.611	**0.186**	**0.257**
Average		*0.603*	0.291	**0.544**	**0.267**	0.691	*0.279*	1.219	0.361	0.914	0.382	1.645	0.581	**0.215**	**0.254**
SSD-100	20	0.022	0.295	**0.015**	**0.287**	*0.021*	*0.289*	0.186	0.361	0.174	0.355	0.331	0.546	**0.013**	**0.271**
	50	*1.024*	0.277	**0.752**	**0.262**	1.163	*0.268*	1.632	0.345	1.438	0.364	2.423	0.576	**0.287**	**0.258**
	100	*1.085*	*0.273*	**0.836**	**0.263**	0.985	*0.273*	1.439	0.353	1.312	0.343	2.354	0.591	**0.322**	**0.255**
	200	*0.417*	0.267	*0.554*	**0.252**	0.572	*0.262*	0.976	0.365	0.753	0.368	1.548	0.563	**0.167**	**0.227**
	500	*0.337*	0.326	*0.473*	**0.281**	0.506	*0.295*	1.711	0.416	0.762	0.437	1.412	0.603	**0.143**	**0.263**
Average		**0.577**	0.288	**0.526**	**0.269**	0.649	*0.277*	1.189	0.368	0.888	0.373	1.614	0.576	**0.186**	**0.255**
SSD-125	20	0.024	0.292	**0.017**	**0.281**	*0.019*	*0.285*	0.182	0.356	0.167	0.352	0.328	0.552	**0.011**	**0.268**
	50	*1.017*	0.288	**0.782**	**0.257**	1.183	*0.267*	1.651	0.348	1.426	0.356	2.513	0.571	**0.283**	**0.252**
	100	*1.021*	0.279	**0.743**	**0.251**	0.894	*0.272*	1.374	0.344	1.221	0.338	2.221	0.587	**0.245**	**0.246**
	200	**0.483**	0.265	*0.615*	**0.245**	0.632	*0.257*	0.979	0.357	0.864	0.362	1.624	0.553	**0.223**	**0.241**
	500	**0.412**	0.323	*0.531*	**0.273**	0.546	*0.303*	1.761	0.422	0.812	0.426	1.457	0.592	**0.182**	**0.254**
Average		*0.591*	0.289	**0.538**	**0.261**	0.655	*0.277*	1.189	0.365	0.898	0.367	1.629	0.571	**0.189**	**0.252**
Tot. average		*0.611*	0.291	**0.552**	**0.268**	0.694	*0.280*	1.222	0.364	0.921	0.375	1.648	0.579	**0.214**	**0.257**

of the fast *Insert*-based neighbor evaluation method. Indeed, the proposed fast evaluation method facilitates the efficiency of evaluating *Insert*-based neighbor solutions and reduces the computational cost, thereby allowing more iterations and raising the chances of discovering more promising solutions with less computational effort.

From Table 5, it can be seen that the total average values of MCEDA$_{v4}$ (1.222), MCEDA$_{v5}$ (0.921) and MCEDA$_{v6}$ (1.648) are inferior to MCEDA (0.214). As regards MCEDA$_{v6}$, it achieves the worst results, remarkably lags behind other competitors, clearly revealing that the integration of both $NS_{\text{PM\_boost}}$ and $NS_{\text{PM\_RS\_guide}}$ in the multi-neighborhood iterated local search effectively enhances the searchability. Since both neighborhood search strategies, *i.e.*, $NS_{\text{PM\_boost}}$ and $NS_{\text{PM\_RS\_guide}}$, can utilize valuable probability information of promising patterns from superior solutions to drive neighborhood search, MNILS can fully exploit local areas in depth by cyclically switching between neighborhood search strategies through the framework of ILS. If it is eliminated, the capacity for local exploitation would be greatly diminished. Furthermore, the *SD* values of MCEDA are smaller than those of its variants, *i.e.*, MCEDA produces more stable results across various scale instances, indicating that MCEDA has good robustness and stability. As a consequence of such comparison, MCEDA has a stronger and superior search power, demonstrating the advantages of all well-designed improvement strategies.

Although the statistical results in Table 5 illustrate the superiority of incorporating improvement strategies, ANOVA is still used to further confirm the significance of the observed differences. The results of the ANOVA are reported in Fig. 7, which depicts the interaction between algorithms and scenarios with 95% Tukey's Honest Significant Difference (HSD) confidence intervals. Note that the overlapping intervals among algorithms imply that there are statistically insignificant differences in their performances. As shown in Fig. 7, MCEDA is significantly superior to the other six variants due to the absence of overlapping intervals, confirming the above conclusion that these improvement strategies have a great potential to boost the performance of MCEDA.

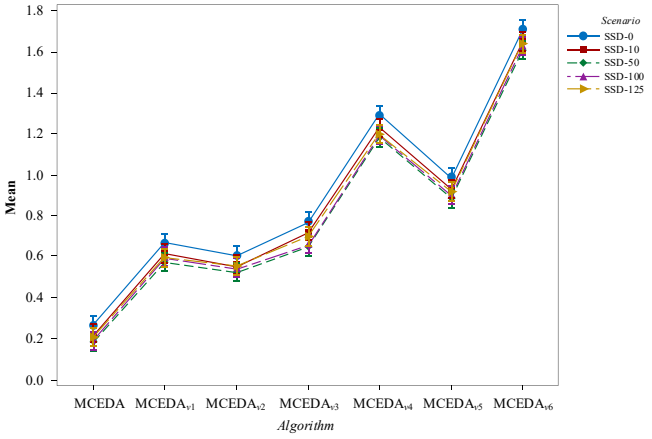

Fig. 7. Interaction plot with 95% Tukey's HSD confidence interval between algorithm and scenario.

Table 6
Comparison results of EDA's global exploration under different scenarios.

n,m	SSD-0				SSD-10				SSD-50				SSD-100				SSD-125			
	$JEDA_{nls}$	$EEDA_{nls}$	$P\text{-}EDA_{nls}$	$MCEDA_{nls}$	$JEDA_{nls}$	$EEDA_{nls}$	$P\text{-}EDA_{nls}$	$MCEDA_{nls}$	$JEDA_{nls}$	$EEDA_{nls}$	$P\text{-}EDA_{nls}$	$MCEDA_{nls}$	$JEDA_{nls}$	$EEDA_{nls}$	$P\text{-}EDA_{nls}$	$MCEDA_{nls}$	$JEDA_{nls}$	$EEDA_{nls}$	$P\text{-}EDA_{nls}$	$MCEDA_{nls}$
20,5	0.24	*0.13*	**0.05**	**0.03**	0.31	*0.11*	**0.04**	**0.02**	0.31	*0.15*	**0.04**	**0.02**	0.36	*0.17*	**0.05**	**0.03**	0.41	*0.19*	**0.05**	**0.03**
20,10	0.32	*0.18*	**0.08**	**0.04**	0.34	*0.14*	**0.05**	**0.03**	0.39	*0.21*	**0.05**	**0.03**	0.45	*0.24*	**0.04**	**0.02**	0.48	*0.25*	**0.06**	**0.03**
20,20	0.28	*0.15*	**0.05**	**0.03**	0.39	*0.19*	**0.08**	**0.04**	0.45	*0.26*	**0.08**	**0.03**	0.57	*0.29*	**0.06**	**0.04**	0.35	*0.23*	**0.06**	**0.04**
50,5	1.81	*1.54*	**1.23**	**1.08**	1.72	*1.38*	**1.07**	**0.72**	1.83	*1.47*	**1.31**	**0.95**	2.14	*1.65*	**1.38**	**1.15**	2.21	*1.83*	**1.54**	**1.26**
50,10	1.95	*1.62*	**1.32**	**1.15**	2.07	*1.53*	**1.23**	**0.91**	2.15	*1.54*	**1.22**	**0.83**	2.19	*1.77*	**1.45**	**1.24**	2.15	*1.79*	**1.47**	**1.14**
50,20	2.12	*1.68*	**1.39**	**1.18**	2.15	*1.46*	**1.14**	**0.77**	1.87	*1.47*	**1.16**	**0.76**	2.08	*1.61*	**1.37**	**1.06**	2.24	*1.69*	**1.35**	**1.05**
100,5	2.23	*1.87*	**1.63**	**1.45**	1.91	*1.54*	**1.29**	**1.06**	1.95	*1.62*	**1.43**	**1.18**	2.17	*1.71*	**1.43**	**1.37**	2.17	*1.75*	**1.44**	**1.28**
100,10	2.18	*1.73*	**1.35**	**1.24**	2.25	*1.63*	**1.38**	**1.12**	2.03	*1.75*	**1.51**	**1.22**	2.14	*1.67*	**1.36**	**1.25**	2.34	*1.97*	**1.63**	**1.43**
100,20	2.31	*1.91*	**1.56**	**1.44**	2.07	*1.51*	**1.22**	**1.04**	2.12	*1.78*	**1.57**	**1.29**	2.25	*1.78*	**1.47**	**1.36**	2.26	*1.89*	**1.57**	**1.36**
200,10	2.36	*1.98*	**1.72**	**1.53**	2.24	*1.67*	**1.43**	**1.17**	1.86	*1.51*	**1.34**	**0.95**	1.83	*1.39*	**1.18**	**0.89**	2.15	*1.73*	**1.34**	**0.93**
200,20	2.13	*1.83*	**1.65**	**1.36**	2.01	*1.58*	**1.32**	**1.09**	1.91	*1.59*	**1.45**	**1.09**	1.91	*1.56*	**1.25**	**1.08**	1.97	*1.77*	**1.45**	**1.25**
500,20	1.14	*1.08*	**0.84**	**0.42**	1.17	*1.09*	**0.89**	**0.55**	1.28	*0.97*	**0.72**	**0.47**	1.04	*0.88*	**0.69**	**0.54**	1.11	*0.98*	**0.69**	**0.42**
Average	1.59	*1.31*	**1.07**	**0.91**	1.55	*1.15*	**0.93**	**0.71**	1.51	*1.19*	**0.99**	**0.74**	1.59	*1.23*	**0.98**	**0.84**	1.65	*1.34*	**1.05**	**0.85**

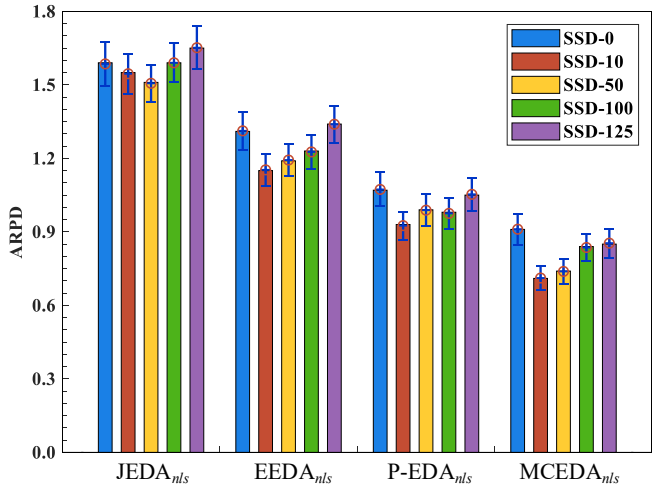

Fig. 8. Comparisons of EDA's global performance.

5.4. Comparisons of MCEDA and other two-dimensional EDAs

Since most well-performing EDA-based algorithms use two-dimensional probabilistic models to guide the global search direction, it is critical to conduct a comprehensive investigation of the global search performance of EDA-based algorithms. To verify the proposed MCEDA's superiority over existing two-dimensional probabilistic model-based EDAs for BFSP_SDST, we compared it with three recently proposed two-dimensional probabilistic model-based EDAs, including the state-of-the-art EDA (JEDA) (Jarboui, et al., 2009), the effective EDA (EEDA) (Wang, et al., 2013), and the modified JEDA (P-EDA) (Pan & Ruiz, 2012). To eliminate the effect of local search on the global performance of these EDAs, only the framework of global search for all EDA-based algorithms is used to perform global exploration without local search. These variants are abbreviated as $MCEDA_{nls}$, $JEDA_{nls}$, $EEDA_{nls}$, and $P\text{-}EDA_{nls}$. The parameters of these EDA-based algorithms are set to the same values as in the original literature. The experimental results under five different scenarios are summarized in Table 6.

As can be observed from Table 6, $MCEDA_{nls}$ achieves the best results in almost all instances compared to the existing effective two-dimensional probabilistic model-based EDAs. Specifically, the global performance of $P\text{-}EDA_{nls}$ is significantly better than that of $JEDA_{nls}$ and $EEDA_{nls}$, which indicates that $P\text{-}EDA_{nls}$ can attain better search performance by using two two-dimensional matrices to preserve information of both the order of jobs and the number of similar blocks. However, the proposed $MCEDA_{nls}$ notably outperforms all the existing EDA-based algorithms in terms of *ARPD* values for five scenarios, SSD-0, SSD-10, SSD-50, SSD-100, and SSD-125. For different setup time scenarios, the histogram including means with 95% Tukey's HSD confidence interval is illustrated in Fig. 8. It is clear that, as compared to other EDA-based algorithms, $MCEDA_{nls}$ can yield significantly lower *ARPD* and relatively smaller *SD* with considerable advantages. These findings demonstrate the benefits of the matrix-cube-based probabilistic model in improving the performance of MCEDA. The main reason is explained by the fact that the three-dimensional probabilistic model employed in $MCEDA_{nls}$ is capable of not only learning valuable information about the order of jobs that existed in superior solutions, but also accurately and reasonably recording the relative position of each similar block, which is difficult to do with two-dimensional probabilistic models. So, for the two-dimensional probabilistic model-based EDAs, similar blocks cannot be placed in the correct positions to produce new individuals during the sampling process, resulting in relatively poor search capability of these comparison algorithms. According to the above experiments and analysis, it can be concluded that the proposed matrix-cube-based probabilistic model plays an important role in MCEDA. Also, it may be

worthwhile to consider embedding the developed multi-dimensional probabilistic model into other HIOAs for solving BFSPs in future research.

5.5. Comparisons of MCEDA and the state-of-the-art methods

To evaluate the effectiveness and efficiency of the proposed MCEDA, this section aims to compare the performance of MCEDA against several state-of-the-art algorithms available in the literature. As described in Section 2, it should be noted that few algorithms are directly designed to address the BFSP_SDST; accordingly, any algorithms that attempt addressing the BFSP_SDST and its relevant problems are considered for computational comparison. For various types of FSPs and BFSPs with makespan criterion, the efficacy of MCEDA with various high-performance algorithms is comprehensively compared according to the same benchmark test sets as SSD-10, SSD-50, SSD-100, and SSD-125 introduced in Section 5.1. These algorithms may be classified into three categories. The first group has a single algorithm, *i.e.*, DWWO (Shao, et al., 2018b), which is developed to solve the BFSP_SDST; The second group contains fourteen algorithms, including HDDE (Wang, et al., 2010), hmgHS (Wang, et al., 2011), TPA (Wang, et al., 2012), MA (Pan, et al., 2013), RAIS (Lin & Ying, 2013), SVNS_S and SVNS_D (Ribas, et al., 2013), HVNS (Moslehi & Khorasanian, 2014), DE_ABC (Han, et al., 2015), DE_PLS (Tasgetiren, et al., 2015), MFFO (Han, et al., 2016), IG_IJ and IG_RLS (Tasgetiren, et al., 2017), and P-EDA (Shao, et al., 2018a), where these algorithms are designed to address the BFSP; The third group contains six algorithms, including HGA (Ruiz, et al., 2005), PACA (Gajpal, et al., 2006), IG_RS (Ruiz & Stutzle, 2008), AHA_1 and AHA_3 (Li & Zhang, 2012), EMBO (Sioud & Gagne, 2018), all of which are developed to deal with the PFSP. Additionally, since the proposed MCEDA is designed based on the multi-dimensional probabilistic model, it is necessary to further conduct a comparison between MCEDA and an effective two-dimensional probabilistic-model-based EDA, *i.e.*, EEDA (Wang, et al., 2013). The total of twenty two typical algorithms mentioned above are the most effective algorithms available for dealing with BFSP, PFSP and their extensions. Among these algorithms, DWWO, HDDE, HMGHS, MA, RAIS, DE_ABC, DE_PLS, MFFO, P-EDA, HGA, PACA, AHA1, AHA3, EMBO, and EEDA all fall within the category of population-based HIOAs. All algorithms are re-implemented strictly in accordance with the original literature, with appropriate adjustments to adapt the BFSP SDST with makespan criterion. Meanwhile, the fast *Insert*-based neighbor evaluation method described in Subsection 4.3.1 is incorporated into these re-implemented algorithms to expedite search efficiency. The parameters of each algorithm are derived from the suggested settings in the original literature, and the same calibration method as stated in Section 5.2 is employed to recalibrate the relevant parameters. The parameter values for all algorithms are reported in Table 7.

Table 7
Parameters of the compared algorithms for BFSP_SDST.

Algorithm	Author(s)	Parameter setting
DWWO	Shao, et al. (2018b)	$popsize = 5$, $\lambda_{\min} = 1$, $\lambda_{\max} = 2$, $h_{\max} = 10$, $\omega = 40$.
HDDE	Wang, et al. (2010)	$popsize = 20$, $F = 0.2$, $CR = 0.2$, $P_l = 0.2$.
hmgHS	Wang, et al. (2011)	$MS = 5$, $P_{CR} = 0.95$, $P_{AR} = 0.95$.
TPA	Wang, et al. (2012)	$T_{init} = \sum_{i=1}^{n}\sum_{j=1}^{n} p_{i,j}/5nm$, $T_{final} = 1$, $N_{iter} = 100000$, $\alpha = 0.8$, $\beta = (T_{init} - T_{final})/((N_{iter} - 1) \times T_{init} \times T_{final})$.
MA	Pan, et al. (2013)	$popsize = 10$, $P_m = 0.8$, $P_c = 0.2$, $\lambda = 20$, $\gamma = 20$.
RAIS	Lin and Ying (2013)	$T_0 = 0.6 \times \sum_{i=1}^{n}\sum_{j=1}^{n} p_{i,j}/10n$, $D_{\text{threshold}} = 5$, $G_T = 4000$, $n_c = 6$, $\alpha = 0.97$, $MaxT = 60nm$.
SVNS_D	Ribas, et al. (2013)	$\alpha = 0.75$, $\beta = 0.5$, $d = 8$.
HVNS	Moslehi and Khorasanian (2014)	$T_{init} = \sum_{i=1}^{n}\sum_{j=1}^{n} p_{i,j}/5nm$, $T_{final} = 0.1 \times T_{init}$, $k_{\max} = 100000$, $\beta = (T_{init} - T_{final})/((N_{iter} - 1) \times T_{init} \times T_{final})$.
DE_ABC	Han, et al. (2015)	$popsize = 20$, $P_{mu} = 0.9$, $P_c = 0.1$, $P_{ls} = 0.2$.
DE_PLS	Tasgetiren, et al. (2015)	$popsize = 10$, $\delta = 20$, $P_c = 0.1$, $\beta = 0.0005$, $F = 0.1$, $T_{\max} = 60nm$.
MFFO	Han, et al. (2016)	$popsize = 20$, $\varphi = 0.75$, $pls = 0.6$, $T = 5$, $Time_{\max} = 60nm$.
IG_RLS	Tasgetiren, et al. (2017)	$ds = 8$, $\tau P = 0.5$, $jP = 0.001$, $T_{\max} = 60nm$.
P-EDA	Shao, et al. (2018a)	$popsize = 50$, $\lambda = 0.3$, $t = 60nm$.
HGA	Ruiz, et al. (2005)	$popsize = 50$, $P_c = 0.1$, $P_m = 0.005$, $G_r = 25$.
PACA	Gajpal, et al. (2006)	$\tau_{ip} = 1/M_{best}$, $\rho = 0.75$.
IG_RS	Ruiz and Stutzle (2008)	$Temperature = T \times \sum_{i=1}^{n}\sum_{j=1}^{n} p_{i,j}/10nm$, $T = 0.5$, $d = 4$.
AHA3	Li and Zhang (2012)	$popsize = 50$,$\alpha = 20$, $\beta = 20$, $\rho = 1$, $\eta_i = 0.7$, $P_c = 0.6$, $P_m = 0.02$, $P_l = 0.1$.
EMBO	Sioud and Gagne (2018)	$popsize = 9$, $m = 100$, $k = 5$, $x = 1$, $age = 100$, $q_0 = 0.7$, $l = 10$.
EEDA	Wang, et al. (2013)	$popsize = 150$, $\eta = 10$, $\alpha = 0.1$.

Table 8
Statistical results of MCEDA compared with 22 different algorithms at scenario SSD-0.

n,*m*	PACA	RAIS	AHA_1	AHA_3	HGA	DE_ABC	DE_PLS	HDDE	MFFO	EMBO	hmgHS	SVNS_S	SVNS_D	P-EDA	EEDA	TPA	MA	HVNS	IG_RS	IG_RLS	IG_IJ	DWWO	MCEDA
20,5	0.24	0.35	0.15	0.14	0.33	0.08	0.03	0.03	0.05	0.04	0.05	0.05	0.04	0.04	0.02	0.02	*0.02*	0.03	0.02	0.02	**0.02**	0.02	**0.00**
20,10	0.27	0.43	0.19	0.17	0.39	0.07	**0.02**	*0.03*	0.07	0.07	0.04	0.05	0.06	0.06	0.04	0.04	0.04	0.04	0.03	0.03	0.04	0.04	**0.01**
20,20	0.18	0.37	0.17	0.14	0.42	0.04	0.03	0.03	0.04	0.05	0.03	0.03	0.03	0.05	0.03	0.04	*0.02*	0.03	0.02	**0.02**	0.03	0.03	**0.01**
50,5	2.53	1.85	1.72	1.65	1.71	1.44	1.17	1.51	1.29	1.22	1.07	0.90	0.95	0.96	0.92	0.71	0.68	0.55	0.64	**0.44**	*0.46*	0.49	**0.35**
50,10	2.46	2.33	1.81	1.72	1.69	1.51	1.28	1.52	1.44	1.27	1.14	0.98	1.05	0.99	1.02	0.76	0.65	0.67	0.59	**0.51**	*0.55*	0.62	**0.31**
50,20	2.63	2.56	1.91	1.77	1.64	1.58	1.36	1.57	1.56	1.31	1.18	1.04	1.12	0.96	1.08	0.75	0.64	0.53	0.45	**0.38**	*0.42*	0.52	**0.35**
100,5	2.43	1.94	2.13	1.88	1.82	1.71	1.43	1.76	1.66	1.45	1.45	1.18	1.26	1.14	1.04	0.81	0.85	0.58	0.73	**0.43**	*0.49*	0.61	**0.36**
100,10	2.79	2.17	2.06	1.85	1.94	1.76	1.54	1.71	1.72	1.48	1.24	1.11	1.19	1.07	1.13	0.77	0.74	0.59	0.64	**0.41**	*0.48*	0.57	**0.34**
100,20	2.76	2.54	2.17	1.95	2.16	1.73	1.52	1.84	1.67	1.59	1.48	1.41	1.44	1.22	1.24	0.89	0.76	0.66	0.67	**0.57**	*0.62*	0.71	**0.42**
200,10	2.65	2.11	2.22	1.98	1.81	1.82	1.58	1.89	1.76	1.65	1.54	1.47	1.52	1.26	1.22	0.96	0.67	0.71	0.69	**0.52**	*0.61*	0.65	**0.34**
200,20	2.38	2.23	2.02	1.81	1.72	1.74	1.52	1.64	1.67	1.53	1.35	1.29	1.34	1.18	1.18	0.79	0.69	0.62	0.63	**0.45**	*0.48*	0.61	**0.34**
500,20	0.96	0.92	1.27	1.09	0.63	0.71	0.48	0.56	0.69	0.75	0.43	0.47	0.53	0.58	0.63	0.47	0.35	0.39	0.32	**0.23**	*0.25*	0.27	**0.16**
Average	1.86	1.65	1.48	1.35	1.36	1.18	1.00	1.18	1.14	1.04	0.92	0.83	0.88	0.79	0.80	0.58	0.51	0.45	0.45	**0.34**	*0.37*	0.43	**0.25**

Table 9
Statistical results of MCEDA compared with 22 different algorithms at scenario SSD-10.

n,*m*	PACA	RAIS	AHA_1	AHA_3	HGA	DE_ABC	DE_PLS	HDDE	MFFO	EMBO	hmgHS	SVNS_S	SVNS_D	P-EDA	EEDA	TPA	MA	HVNS	IG_RS	IG_RLS	IG_IJ	DWWO	MCEDA
20,5	0.22	0.32	0.17	0.16	0.34	0.02	0.03	*0.02*	0.02	0.06	0.02	0.03	0.02	0.05	0.04	0.03	0.02	0.05	0.02	0.02	0.02	**0.01**	**0.00**
20,10	0.19	0.37	0.19	0.17	0.42	0.03	0.03	0.03	0.05	0.04	0.03	0.03	0.04	0.07	0.04	0.03	**0.02**	0.04	0.04	0.03	*0.02*	0.03	**0.00**
20,20	0.17	0.36	0.14	0.13	0.46	*0.02*	0.02	0.04	0.04	0.05	0.04	0.04	0.03	0.06	0.03	0.04	0.02	0.04	0.03	0.04	0.03	**0.02**	**0.01**
50,5	2.16	1.71	1.53	1.48	1.84	1.12	0.81	1.23	0.94	1.13	0.72	0.82	0.89	0.83	0.87	0.72	0.45	0.56	0.56	*0.39*	0.44	**0.34**	**0.27**
50,10	2.44	2.06	1.76	1.68	1.93	1.43	1.22	1.54	1.41	1.35	0.91	0.88	0.95	1.13	1.22	0.84	*0.48*	0.79	0.54	0.68	0.72	**0.37**	**0.22**
50,20	2.31	2.21	1.85	1.81	1.85	1.47	1.24	1.49	1.37	1.26	0.79	0.98	1.05	0.93	1.03	0.58	*0.39*	0.43	0.47	0.56	0.62	**0.33**	**0.18**
100,5	1.96	1.81	1.61	1.57	1.27	1.22	0.91	1.27	1.29	1.22	1.07	0.92	0.98	0.88	0.97	0.67	0.71	0.58	0.66	*0.38*	0.44	**0.36**	**0.33**
100,10	2.13	2.04	1.78	1.76	1.69	1.28	1.03	1.36	1.32	1.37	1.13	1.05	1.08	0.85	1.09	0.71	0.57	0.58	0.48	*0.42*	0.45	**0.39**	**0.26**
100,20	2.39	2.22	1.95	1.89	1.88	1.45	0.93	1.47	1.27	1.38	1.03	1.13	1.17	0.94	1.18	0.74	*0.52*	0.64	0.62	0.56	0.59	**0.47**	**0.33**
200,10	2.04	1.93	1.79	1.65	1.62	1.31	1.08	1.32	1.36	1.31	1.18	1.06	1.13	1.13	1.14	0.81	*0.48*	0.74	0.74	0.63	0.68	**0.38**	**0.26**
200,20	2.35	2.05	2.03	1.96	1.66	1.39	1.19	1.51	1.34	1.36	1.09	1.11	1.17	1.07	1.11	0.74	*0.45*	0.63	0.57	0.49	0.52	**0.41**	**0.33**
500,20	0.89	0.95	1.12	1.08	0.70	0.57	0.43	0.63	0.66	1.11	0.53	0.42	0.47	0.78	0.79	0.62	0.34	0.47	0.25	*0.25*	0.27	**0.24**	**0.15**
Average	1.60	1.50	1.33	1.28	1.30	0.94	0.75	0.99	0.92	0.97	0.71	0.71	0.75	0.73	0.79	0.55	0.37	0.46	0.42	*0.37*	0.40	**0.28**	**0.20**

Table 10
Statistical results of MCEDA compared with 22 different algorithms at scenario SSD-50.

n,*m*	PACA	RAIS	AHA_1	AHA_3	HGA	DE_ABC	DE_PLS	HDDE	MFFO	EMBO	hmgHS	SVNS_S	SVNS_D	P-EDA	EEDA	TPA	MA	HVNS	IG_RS	IG_RLS	IG_IJ	DWWO	MCEDA
20,5	0.39	0.32	0.23	0.22	0.49	0.04	0.03	0.04	0.04	0.07	0.02	0.03	0.04	0.07	0.07	0.06	0.02	0.04	0.03	0.02	**0.02**	*0.02*	**0.00**
20,10	0.57	0.43	0.27	0.24	0.72	**0.02**	0.03	0.03	0.04	0.06	0.03	0.04	0.05	0.09	0.05	0.08	0.03	0.04	0.03	*0.02*	0.03	0.03	**0.00**
20,20	0.45	0.36	0.22	0.19	0.61	0.04	0.03	0.04	*0.03*	0.06	0.03	0.03	0.03	0.11	0.09	0.05	0.03	0.03	0.03	0.03	0.04	**0.03**	**0.02**
50,5	2.12	1.87	1.52	1.45	1.93	1.26	0.94	1.42	1.12	1.24	0.97	1.04	1.17	0.93	0.99	0.72	0.52	0.66	0.62	*0.51*	0.55	**0.38**	**0.32**
50,10	2.22	2.13	1.63	1.55	1.82	1.17	0.88	1.49	1.06	1.45	0.84	0.97	1.09	1.23	0.89	0.99	0.45	0.85	0.71	*0.44*	0.49	**0.43**	**0.36**
50,20	2.13	2.39	1.61	1.49	1.77	1.39	1.03	1.45	1.29	1.34	0.78	1.12	1.28	0.96	1.02	0.84	*0.52*	0.75	0.66	0.56	0.64	**0.51**	**0.31**
100,5	1.88	1.84	1.45	1.41	1.54	1.48	1.20	1.44	1.36	1.25	1.17	1.17	1.31	1.16	0.98	0.87	0.66	0.76	0.45	*0.43*	0.45	**0.41**	**0.24**
100,10	1.84	2.03	1.47	1.44	1.69	1.48	1.22	1.42	1.43	1.47	1.22	1.36	1.45	0.96	1.04	0.83	0.68	0.78	0.72	*0.61*	0.66	**0.59**	**0.35**
100,20	2.13	2.36	1.63	1.58	1.79	1.52	1.19	1.52	1.47	1.51	1.28	1.45	1.56	1.27	1.18	1.03	0.72	0.91	0.81	*0.65*	0.69	**0.45**	**0.22**
200,10	1.84	1.82	1.48	1.43	1.56	1.55	1.09	1.47	1.29	1.29	0.97	1.42	1.51	0.97	1.11	0.86	*0.39*	0.66	0.49	0.45	0.53	**0.38**	**0.19**
200,20	1.87	1.94	1.52	1.47	1.53	1.49	1.17	1.42	1.35	1.34	1.06	1.24	1.37	1.15	0.88	0.95	0.43	0.87	0.42	*0.39*	0.45	**0.38**	**0.24**
500,20	1.27	0.97	0.94	0.91	0.84	0.72	0.49	0.71	0.74	1.13	0.49	0.47	0.57	1.03	0.86	0.82	0.34	0.63	**0.27**	*0.28*	0.32	0.34	**0.17**
Average	1.56	1.54	1.16	1.12	1.36	1.01	0.78	1.04	0.93	1.02	0.74	0.86	0.95	0.83	0.76	0.68	0.40	0.58	0.44	*0.37*	0.40	**0.33**	**0.20**

Table 11
Statistical results of MCEDA compared with 22 different algorithms at scenario SSD-100.

n,*m*	PACA	RAIS	AHA_1	AHA_3	HGA	DE_ABC	DE_PLS	HDDE	MFFO	EMBO	hmgHS	SVNS_S	SVNS_D	P-EDA	EEDA	TPA	MA	HVNS	IG_RS	IG_RLS	IG_IJ	DWWO	MCEDA
20,5	0.35	0.38	0.27	0.24	0.35	0.05	0.04	0.05	0.04	0.06	0.03	0.05	0.05	0.09	0.05	0.08	0.03	0.05	0.03	**0.02**	*0.03*	0.03	**0.00**
20,10	0.42	0.45	0.29	0.26	0.55	0.05	0.03	0.04	0.04	0.05	*0.02*	0.04	0.05	0.08	0.04	0.07	0.02	0.04	**0.02**	0.02	0.03	0.04	**0.02**
20,20	0.48	0.52	0.35	0.33	0.65	0.04	0.04	0.06	0.06	0.07	0.04	0.06	0.06	0.11	0.08	0.09	*0.03*	0.04	0.05	0.04	0.04	**0.03**	**0.01**
50,5	1.95	2.08	1.69	1.64	1.81	1.39	1.23	1.37	1.35	1.59	1.14	0.95	1.08	1.27	1.46	1.09	0.66	0.84	0.76	*0.65*	0.72	**0.53**	**0.32**
50,10	2.18	2.21	1.73	1.68	1.95	1.52	1.36	1.45	1.58	1.62	1.25	1.11	1.23	1.43	1.41	1.12	*0.54*	0.93	0.83	0.77	0.82	**0.48**	**0.21**
50,20	1.91	2.15	1.72	1.64	1.84	1.45	1.31	1.45	1.61	1.59	1.05	1.04	1.12	1.24	1.27	1.07	*0.47*	0.81	0.72	0.69	0.76	**0.45**	**0.24**
100,5	2.06	1.83	1.71	1.67	1.92	1.46	1.24	1.43	1.37	1.63	1.38	0.96	1.07	1.36	1.38	1.09	0.59	0.93	0.58	*0.57*	0.65	**0.57**	**0.26**
100,10	2.05	2.21	1.68	1.58	1.87	1.48	1.33	1.34	1.55	1.69	1.22	0.92	1.04	1.32	1.43	1.13	*0.63*	0.88	0.85	0.78	0.83	**0.61**	**0.31**
100,20	2.26	2.17	1.78	1.72	2.12	1.53	1.34	1.49	1.61	1.74	1.37	1.18	1.24	1.43	1.49	1.11	*0.57*	0.87	0.81	0.75	0.81	**0.56**	**0.32**
200,10	1.83	1.97	1.51	1.45	1.55	1.39	1.22	1.14	1.44	1.68	0.88	0.98	1.13	1.38	1.41	1.12	0.49	0.86	0.63	*0.47*	0.53	**0.46**	**0.17**
200,20	1.95	2.13	1.59	1.57	1.89	1.43	1.25	1.26	1.49	1.62	1.09	1.06	1.17	1.31	1.22	1.13	*0.43*	0.93	0.76	0.52	0.59	**0.42**	**0.13**
500,20	1.23	1.02	0.88	0.83	0.93	0.76	0.74	0.64	0.95	1.44	0.56	0.41	0.53	1.09	1.13	0.88	*0.35*	0.74	0.41	0.35	0.43	**0.34**	**0.12**
Average	1.56	1.59	1.27	1.22	1.45	1.05	0.93	0.98	1.09	1.23	0.84	0.73	0.81	1.01	1.03	0.83	*0.40*	0.66	0.54	0.47	0.52	**0.38**	**0.18**

Table 12
Statistical results of MCEDA compared with 22 different algorithms at scenario SSD-125.

n,*m*	PACA	RAIS	AHA_1	AHA_3	HGA	DE_ABC	DE_PLS	HDDE	MFFO	EMBO	hmgHS	SVNS_S	SVNS_D	P-EDA	EEDA	TPA	MA	HVNS	IG_RS	IG_RLS	IG_IJ	DWWO	MCEDA
20,5	0.32	0.41	0.29	0.26	0.44	0.06	0.04	0.05	0.05	0.05	0.03	0.06	0.05	0.11	0.25	0.06	0.04	0.05	**0.02**	0.04	0.05	*0.03*	**0.00**
20,10	0.41	0.47	0.37	0.32	0.57	0.06	0.03	0.06	0.04	0.06	0.04	0.05	0.05	0.09	0.23	0.07	0.04	0.05	*0.03*	0.03	0.04	**0.02**	**0.02**
20,20	0.27	0.33	0.22	0.18	0.35	0.05	0.03	0.05	0.04	0.05	0.04	0.03	0.04	0.13	0.18	0.05	*0.03*	0.04	**0.03**	0.04	0.05	0.04	**0.01**
50,5	2.03	2.24	1.71	1.66	1.89	1.38	1.25	1.55	1.42	1.62	1.27	1.19	1.21	1.34	1.23	0.93	0.65	0.75	0.69	*0.63*	0.70	**0.51**	**0.28**
50,10	1.89	2.32	1.64	1.59	1.73	1.28	1.24	1.56	1.37	1.62	1.14	1.09	1.13	1.43	1.32	0.84	*0.59*	0.73	0.77	0.71	0.75	**0.58**	**0.29**
50,20	1.78	2.26	1.44	1.39	1.61	1.24	1.35	1.37	1.39	1.53	1.04	0.92	0.99	1.39	1.41	0.91	0.54	0.72	0.63	*0.54*	0.62	**0.54**	**0.24**
100,5	1.86	2.15	1.62	1.54	1.73	1.34	1.33	1.46	1.43	1.57	1.28	1.22	1.28	1.37	1.40	0.95	0.79	0.74	0.56	*0.51*	0.59	**0.49**	**0.21**
100,10	1.92	2.36	1.72	1.67	1.84	1.32	1.29	1.61	1.55	1.49	1.42	1.12	1.21	1.33	1.42	1.14	0.86	0.83	0.74	*0.66*	0.71	**0.64**	**0.25**
100,20	2.06	2.25	1.74	1.69	2.17	1.41	1.31	1.56	1.59	1.58	1.35	1.27	1.32	1.43	1.51	1.27	*0.65*	0.91	0.88	0.72	0.77	**0.54**	**0.22**
200,10	1.74	2.07	1.41	1.34	1.19	1.22	1.19	1.31	1.35	1.55	0.95	0.84	0.93	1.41	1.53	1.09	0.58	0.75	0.57	*0.55*	0.59	**0.53**	**0.21**
200,20	1.86	1.94	1.53	1.45	1.77	1.27	1.21	1.42	1.53	1.46	1.23	1.02	1.09	1.36	1.25	1.25	*0.41*	0.83	0.64	0.56	0.64	**0.38**	**0.18**
500,20	1.02	1.12	0.78	0.71	0.89	0.76	0.71	0.68	0.94	1.12	0.46	0.53	0.56	0.95	0.96	0.76	*0.32*	0.56	0.43	0.34	0.38	**0.26**	**0.17**
Average	1.43	1.66	1.21	1.15	1.35	0.95	0.92	1.06	1.06	1.14	0.86	0.78	0.82	1.03	1.06	0.78	0.46	0.58	0.50	*0.44*	0.49	**0.38**	**0.18**

The comprehensive comparison results of MCEDA against 22 different algorithms under different scenarios are summarized in Tables 8-12. Notably, the setup time for all instances in SSD-0 is zero, implying that the BFSP SDST is reduced to the BFSP in this scenario. As seen in Table 8, all *ARPD* values obtained by MCEDA are the smallest when compared to the other algorithms. These numerical results reflect that the proposed MCEDA is capable of successfully solving BFSP. Moreover, it can be observed from Tables 9-12 that MCEDA yields the lowest *ARPD* values on the other four test sets, SSD-10, SSD-50, SSD-100, and SSD-125, indicating that the proposed MCEDA has a stronger search engine in terms of tackling the problem under consideration. Furthermore, MCEDA achieves the lowest overall average values of *ARPD* on 12 instances spanning a range of scales and scenarios, demonstrating that MCEDA outperforms the other algorithms in an average sense.

To further investigate the performance differences amongst algorithms, all results are also analyzed by means of multifactor ANOVA. The ANOVA is applied to determine whether or not there are indeed statistically significant differences in the *ARPD* values acquired by all of the compared algorithms. It is worth mentioning that in statistical trials with 95% confidence level ($\alpha = 0.05$), three main hypotheses are checked, including normality, homoscedasticity, and independence of residuals. From the analysis of the residuals resulting from the experimental results, all hypotheses are readily satisfied. Figs. 9-13 display the mean plots with 95% Tukey's HSD confidence intervals and the corresponding box plots for all test results obtained by MCEDA and 22 algorithms at different scales and scenarios, respectively. It is remarkable that the presence of overlapping confidence intervals between any two algorithms signifies that the observed differences are not statistically significant, signaling that there is no significant difference in the performance of algorithms. As can be seen in Figs. 9-13, there is no overlap between MCEDA and the other compared algorithms in different scenarios, indicating that the results achieved by MCEDA are statistically significant different from those acquired by other algorithms. In other words, from a statistical point of view, MCEDA has significant advantages in solving both BFSP and BFSP_SDST.

Moreover, when compared to several two-dimensional probabilistic model-based EDAs, MCEDA statistically significantly outperforms EEDA and P-EDA across all test sets, which highlights the fact that the developed multi-dimensional probabilistic model has obvious advantages in learning and utilizing promising patterns of superior solutions. It exhibits its prowess in terms of exploration and exploitation. Furthermore, the recently proposed DWWO and a series of enhanced or hybrid IG versions, including IG_RS, IG_RLS, and IG_IJ, are also competitive in all scenarios. It is obvious that the confidence intervals of DWWO and several extended IG are almost completely overlapped on SSD-0 and partially overlapped on SSD-10, SSD-50, SSD-100, and SSD-150, revealing that these algorithms perform similarly in these scenarios. In this sense, the results obtained by DWWO are slightly better than IG_RLS, except on the SSD-0, due to its high-performing local search mechanism, while the proposed MCEDA beats all its competitors by a considerable margin on small to large-scale instances. Also, as illustrated in the box plots in Figs. 9-13, MCEDA remarkably outperforms other state-of-the-art algorithms. In view of the abovementioned observations, not only the scheduling solutions provided by MCEDA are of higher quality, but the numerical results also have a narrower fluctuation range, implying that the proposed MCEDA is competitive and stable.

Additionally, according to the above analysis, it can be seen that there are significant differences in the performance of the compared algorithms for different scale instances and scenarios. To further investigate these behaviors, the interaction plots with 95% Tukey's HSD confidence interval between algorithm and *n*, between algorithm and *m*, and between algorithm and scenario, are provided in Figs. 14-16. As revealed in these figures, all algorithms are sensitive to the number of jobs, machines, and scenarios. However, as for the proposed MCEDA, it is depicted in Fig. 16 that the scales and scenarios have slight effect on it.

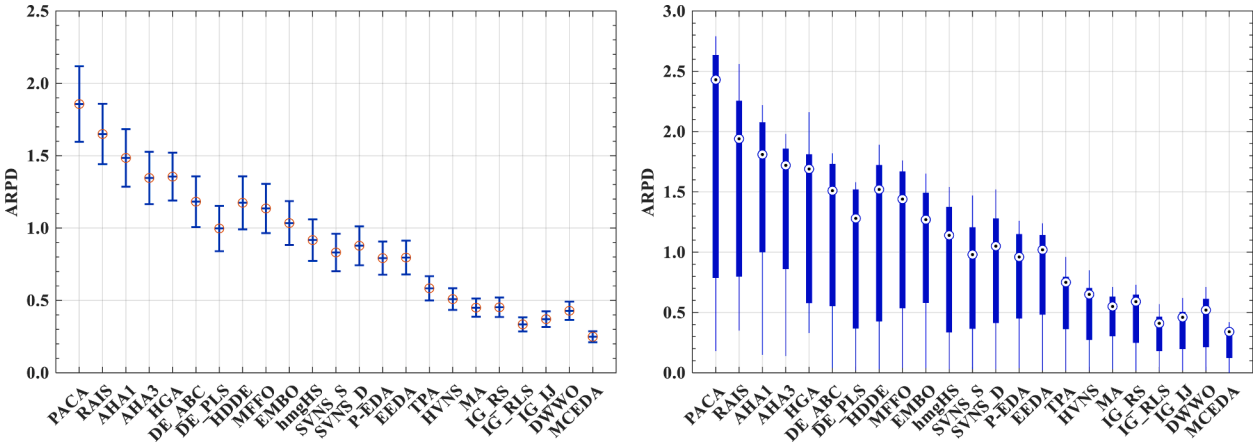

Fig. 9. Means plots with 95% Tukey's HSD confidence interval and box plots for MCEDA compared with 22 different algorithms (SSD-0).

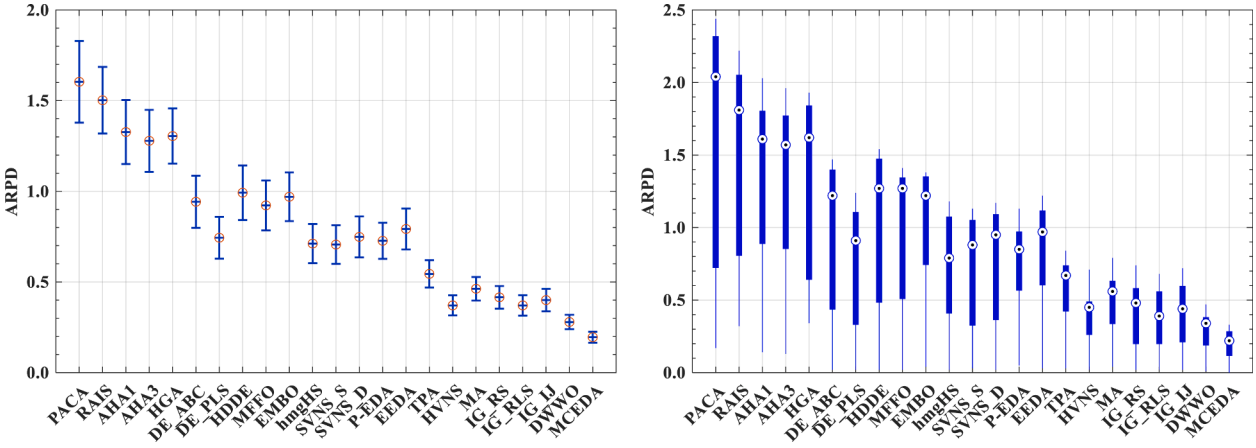

Fig. 10. Means plots with 95% Tukey's HSD confidence interval and box plots for MCEDA compared with 22 different algorithms (SSD-10).

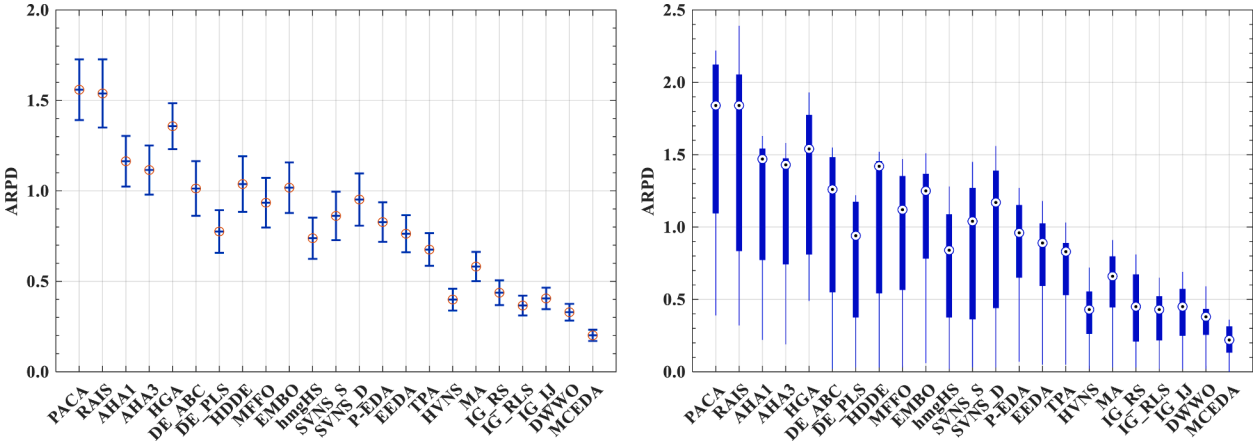

Fig. 11. Means plots with 95% Tukey's HSD confidence interval and box plots for MCEDA compared with 22 different algorithms (SSD-50).

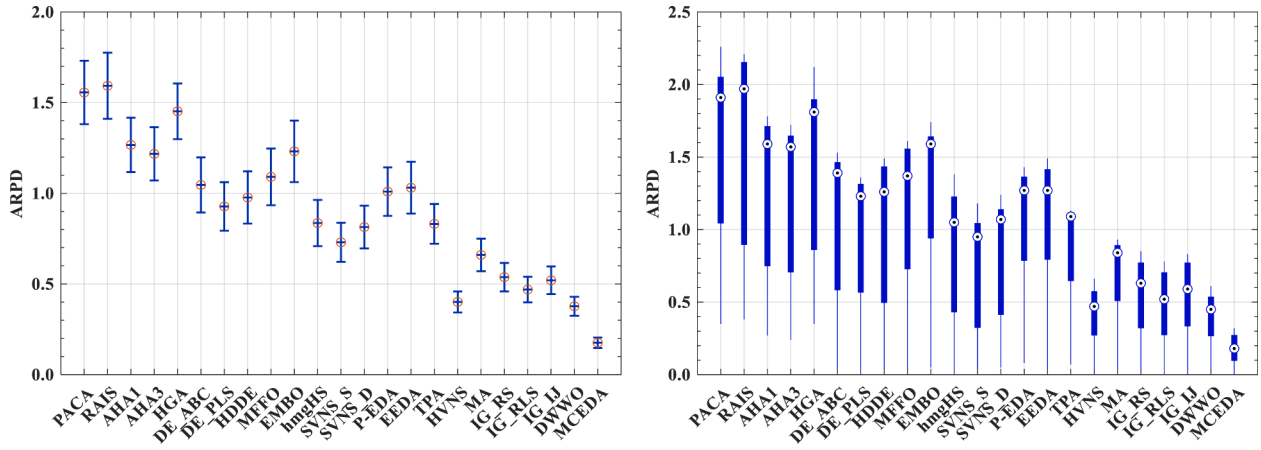

Fig. 12. Means plots with 95% Tukey's HSD confidence interval and box plots for MCEDA compared with 22 different algorithms (SSD-100).

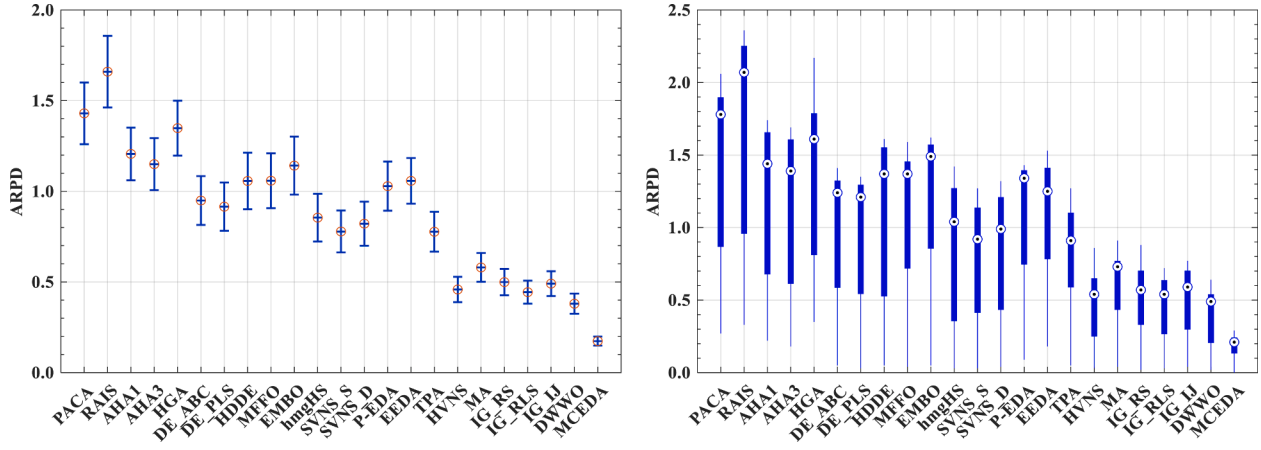

Fig. 13. Means plots with 95% Tukey's HSD confidence interval and box plots for MCEDA compared with 22 different algorithms (SSD-125).

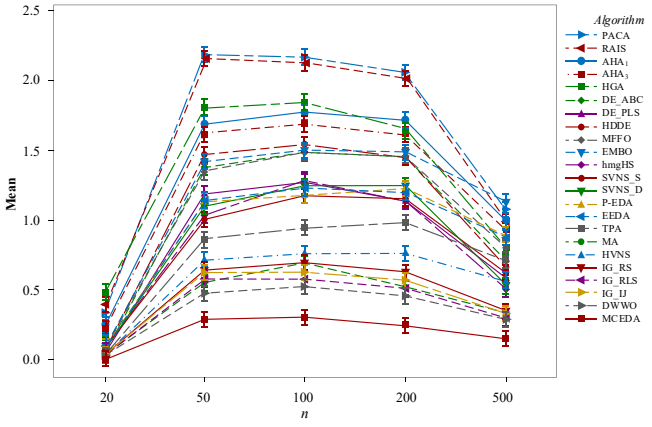

Fig. 14. Interaction plot with 95% Tukey's HSD confidence interval between algorithm and *n*.

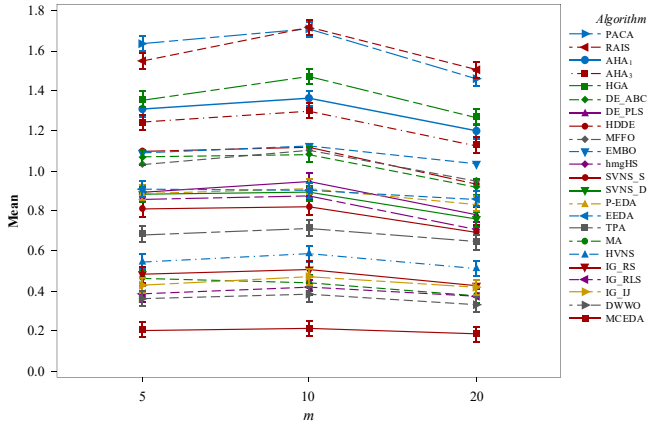

Fig. 15. Interaction plot with 95% Tukey's HSD confidence interval between algorithm and *m*.

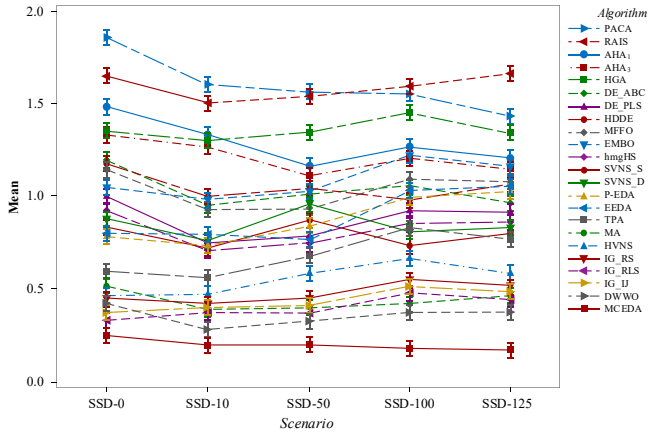

Fig. 16. Interaction plot with 95% Tukey's HSD confidence interval between algorithm and scenario.

Meanwhile, it is obvious that the proposed MCEDA performs superiorly and stably under different scale instances and scenarios, particularly when addressing the large-scale instances with $n \geqslant 100$, which suggests that MCEDA has superiority and stability.

Furthermore, to further verify the statistical validity of the numerical results obtained by these algorithms, the parametric Duncan's multiple range test (DMRT) is employed, which is a post hoc test for detecting the specific differences between pairs of means. Here, DMRT is used to categorize all comparing algorithms into distinct levels. Table 13 summarizes the results of DMRT under different scenarios with a confidence level of $\alpha = 0.05$. As shown in Table 13, all algorithms are graded into twelve levels. MCEDA ranks the first level (*i.e.*, A) for all scenarios, *i.e.*, SSD-0, SSD-10, SSD-50, SSD-100, and SSD-125. There are no other algorithms except MCEDA in the level A, demonstrating that the differences between MCEDA with other compared algorithms are statistically significant and underlining the fact that MCEDA delivers the best performance among all compared algorithms. Meanwhile, DWWO, MA, IG_RLS, IG_IJ, IG_RS, and HVNS are grouped together at the second level, indicating that they have similar performance, and the remaining comparison algorithms perform worse. The DMRT further confirms the competitiveness of MCEDA.

To sum up, according to the above comparative results and statistical analysis, it can be credibly concluded that MCEDA is an extremely effective and efficient algorithm for BFSP_SDST aiming at minimizing makespan. The superiority of MCEDA is mainly attributed to the following aspects: (1) the designed fast *Insert*-based neighbor evaluation method reduces the computational cost and speeds up the search process; (2) the devised two effective constructive heuristics provide diversity and quality for the initial population; (3) the presented matrix-cube-based multi-dimensional probabilistic model accurately estimates the distribution of promising patterns of superior solutions in the solution space; (4) the proposed multi-dimensional probabilistic model based local intensification performs detailed and in-depth multi-neighborhood local search; and (5) the developed diversity controlling mechanism maintains the search vitality, eliminates the search stagnation and prevents falling into local optima. In summary, MCEDA is capable of successfully solving the BFSP SDST and BFSP. Additionally, Table 14 reports the best results so far found by MCEDA for solving the BFSP for 120 instances of various scales in the Tailliard test set SSD-0.

6. Conclusion and future work

In this paper, a matrix-cube-based estimation of distribution algorithm (MCEDA) is proposed to solve a kind of important scheduling problem, *i.e.*, the blocking flow-shop scheduling problem with sequence-dependent setup times (BFSP_SDST). To the best of our knowledge, this is the first report on the application of EDA to the BFSP problems.

From the extensive test results, it can be concluded that the use of deep and fast local search is recommended. For non-convex optimization problems such as production scheduling, the explicit relationship between its intrinsic geometric structure and optimal solution is still an open problem. At present, it is impossible to directly obtain the quantitative relationship between them. In consequence, it is impossible to design a polynomial time algorithm that can obtain the optimal solution according to this relationship. Due to the objective existence of this unsolved open problem, how to execute local search as deeply as possible from the promising regions determined by global search, so as to obtain the high-quality near optimal solutions, is still the key to the design of high-performance hybrid intelligent scheduling algorithm.

Under this background, MCEDA utilizes the probability model that retains the block patterns of excellent solutions to dynamically generate rich promising neighborhoods in its local search process, which can ensure that its local search continues to search downward before reaching the local minimum solution common to all neighborhoods, so that it can have better performance. Meanwhile, MCEDA also utilizes the proposed fast neighbor evaluation methods to further improve the efficiency of its local search.

We have also shown that both the initial population and the structure of probability model have some effects on the performance of MCEDA. In particular, it is recommended to use the PFT_NEH(x) heuristic and the PFZ_RZ(x) heuristic to generate some excellent initial individuals, and employ a three-dimensional probabilistic model to reasonably guide the direction of global search.

There are two valuable directions for future research. Firstly, it

Table 13
Results of Duncan's multiple range test ($\alpha = 0.05$).

Rank	SSD-0	SSD-10	SSD-50	SSD-100	SSD-125
A	**{MCEDA}**	**{MCEDA}**	**{MCEDA}**	**{MCEDA}**	**{MCEDA}**
B	{IG_RLS, IG_IJ, DWWO, HVNS, IG_RS, MA, TPA, P-EDA, EEDA}	{DWWO, MA, IG_RLS, IG_IJ, IG_RS, HVNS, TPA}	{DWWO, IG_RLS, MA, IG_IJ, IG_RS, HVNS}	{DWWO, MA, IG_RLS, IG_IJ, IG_RS, HVNS}	{DWWO, IG_RLS, MA, IG_IJ, IG_RS, HVNS}
C	{SVNS_S, SVNS_D}	{SVNS_S, hmgHS, P-EDA, DE_PLS, SVNS_D}	{TPA, hmgHS, EEDA, DE_PLS}	{SVNS_S, SVNS_D, TPA, hmgHS}	{TPA, SVNS_S, SVNS_D, hmgHS}
D	{hmgHS}	{EEDA}	{P-EDA}	{DE_PLS}	{DE_PLS}
E	{DE_PLS}	{MFFO, DE_ABC}	{SVNS_S}	{HDDE, P-EDA}	{DE_ABC}
F	{EMBO}	{EMBO, HDDE}	{MFFO, SVNS_D, DE_ABC, EMBO, HDDE}	{EEDA, DE_ABC}	{P-EDA, HDDE, EEDA, MFFO}
G	{MFFO}	{AHA_3, HGA, AHA_1}	{AHA_3}	{MFFO}	{EMBO, AHA_3, AHA_1}
H	{HDDE, DE_ABC, AHA_3, HGA}	{RAIS, PACA}	{AHA_1}	{AHA_3, EMBO}	{HGA}
I	{AHA_1}		{HGA}	{AHA_1}	{PACA}
J	{RAIS}		{RAIS}	{HGA}	{RAIS}
K	{PACA}		{PACA}	{PACA}	
L				{RAIS}	
F-ratio	6.752	7.294	6.921	6.322	6.000
p-value	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**

Table 14
Best solutions for BFSP with makespan criterion.

Instance	Best	MCEDA	Instance	Best	MCEDA	Instance	Best	MCEDA
Ta1	1374	1374	Ta41	3611	3617	Ta81	7712	7712
Ta2	1408	1408	Ta42	3470	3470	Ta82	7744	7744
Ta3	1280	1280	Ta43	3466	3466	Ta83	7723	7723
Ta4	1448	1448	Ta44	3650	3650	Ta84	7743	7743
Ta5	1341	1341	Ta45	3582	3596	Ta85	7730	7734
Ta6	1363	1363	Ta46	3571	3571	Ta86	7779	7779
Ta7	1381	1381	Ta47	3667	3667	Ta87	7870	7870
Ta8	1379	1379	Ta48	3554	3554	Ta88	7898	7921
Ta9	1373	1373	Ta49	3508	3515	Ta89	7818	7818
Ta10	1283	1283	Ta50	3608	3608	Ta90	7856	7856
Ta11	1698	1698	Ta51	4479	4479	Ta91	13,149	13,149
Ta12	1833	1833	Ta52	4262	4262	Ta92	13,100	13,100
Ta13	1659	1659	Ta53	4261	4261	Ta93	13,204	13,204
Ta14	1535	1535	Ta54	4338	4345	Ta94	13,125	13,125
Ta15	1617	1617	Ta55	4249	4252	Ta95	13,150	13,150
Ta16	1590	1590	Ta56	4271	4274	Ta96	12,922	12,922
Ta17	1622	1622	Ta57	4291	4291	Ta97	13,431	13,445
Ta18	1731	1731	Ta58	4298	4298	Ta98	13,299	13,299
Ta19	1747	1747	Ta59	4304	4304	Ta99	13,105	13,105
Ta20	1782	1782	Ta60	4399	4399	Ta100	13,201	13,201
Ta21	2436	2436	Ta61	6070	6070	Ta101	14,192	14,237
Ta22	2234	2234	Ta62	5943	5943	Ta102	14,749	14,749
Ta23	2479	2479	Ta63	5851	5851	Ta103	14,874	14,874
Ta24	2348	2348	Ta64	5656	5667	Ta104	14,808	14,808
Ta25	2435	2435	Ta65	5901	5908	Ta105	14,628	14,628
Ta26	2383	2383	Ta66	5759	5764	Ta106	14,765	14,793
Ta27	2390	2390	Ta67	5920	5922	Ta107	14,787	14,787
Ta28	2328	2328	Ta68	5809	5809	Ta108	14,836	14,845
Ta29	2363	2363	Ta69	6035	6035	Ta109	14,711	14,711
Ta30	2323	2323	Ta70	6059	6059	Ta110	14,750	14,758
Ta31	2980	2980	Ta71	6916	6916	Ta111	35,513	35,524
Ta32	3182	3182	Ta72	6669	6671	Ta112	35,805	35,805
Ta33	2995	2995	Ta73	6797	6797	Ta113	35,479	35,479
Ta34	3116	3116	Ta74	7039	7039	Ta114	35,030	35,124
Ta35	3139	3139	Ta75	6733	6736	Ta115	35,487	35,487
Ta36	3158	3162	Ta76	6537	6537	Ta116	35,803	35,803
Ta37	3005	3008	Ta77	6707	6707	Ta117	35,451	35,451
Ta38	3042	3044	Ta78	6746	6746	Ta118	35,644	35,644
Ta39	2889	2889	Ta79	6928	6935	Ta119	35,421	35,421
Ta40	3097	3097	Ta80	6844	6851	Ta120	35,761	35,773

would be meaningful to design a probability model combined with reinforcement learning mechanism to further enhance the guidance ability of global search and the in-depth exploitation ability of local search. Secondly, the proposed MCEDA can be extended to address other important scheduling problems, such as the low-carbon production and transportation integrated scheduling problems.

CRediT authorship contribution statement

Zi-Qi Zhang: Investigation, Methodology, Software, Writing – original draft. **Bin Qian:** Methodology, Funding acquisition, Supervision, Writing – review & editing. **Rong Hu:** 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 Basic Research Key Project of Yunnan Province (202201AS070030), and the National Natural Science Foundation of China (62173169, 61963022, 61873328).

References

Ding, J. Y., Song, S. J., Gupta, J. N. D., Wang, C., Zhang, R., & Wu, C. (2016). New block properties for flowshop scheduling with blocking and their application in an iterated greedy algorithm. *International Journal of Production Research, 54*, 4759–4772.

Elmi, A., & Topaloglu, S. (2013). A scheduling problem in blocking hybrid flow shop robotic cells with multiple robots. *Computers & Operations Research, 40*, 2543–2555.

Faraji Amiri, M., & Behnamian, J. (2020). Multi-objective green flowshop scheduling problem under uncertainty: Estimation of distribution algorithm. *Journal of Cleaner Production, 251*, Article 119734.

Fernandez-Viagas, V., Leisten, R., & Framinan, J. M. (2016). A computational evaluation of constructive and improvement heuristics for the blocking flow shop to minimise total flowtime. *Expert Systems with Applications, 61*, 290–301.

Gajpal, Y., Rajendran, C., & Ziegler, H. (2006). An ant colony algorithm for scheduling in flowshops with sequence-dependent setup times of jobs. *International Journal of Advanced Manufacturing Technology, 30*, 416–424.

Gong, D. W., Han, Y. Y., & Sun, J. Y. (2018). A novel hybrid multi-objective artificial bee colony algorithm for blocking lot-streaming flow shop scheduling problems. *Knowledge-Based Systems, 148*, 115–130.

Gong, H., Tang, L. X., & Duin, C. W. (2010). A two-stage flow shop scheduling problem on a batching machine and a discrete machine with blocking and shared setup times. *Computers & Operations Research, 37*, 960–969.

Grabowski, J., & Pempera, J. (2007). The permutation flow shop problem with blocking. A tabu search approach. *Omega-International Journal of Management Science, 35*, 302–311.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. H. G. R. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. *Annals of Discrete Mathematics, 5*, 287–326.

Hall, N. G., & Sriskandarajah, C. (1996). A Survey of Machine Scheduling Problems with Blocking and No-Wait in Process. *Operations Research, 44*, 510-525.

Han, Y., Gong, D., Jin, Y., & Pan, Q. (2019). Evolutionary Multiobjective Blocking Lot-Streaming Flow Shop Scheduling With Machine Breakdowns. *IEEE Trans Cybern, 49*, 184–197.

Han, Y. Y., Gong, D. W., Li, J. Q., & Zhang, Y. (2016). Solving the blocking flow shop scheduling problem with makespan using a modified fruit fly optimisation algorithm. *International Journal of Production Research, 54*, 6782–6797.

Han, Y. Y., Gong, D. W., & Sun, X. Y. (2015). A discrete artificial bee colony algorithm incorporating differential evolution for the flow-shop scheduling problem with blocking. *Engineering Optimization, 47*, 927–946.
Han, Y. Y., Li, J. Q., Sang, H. Y., Liu, Y. P., Gao, K. Z., & Pan, Q. K. (2020). Discrete evolutionary multi-objective optimization for energy-efficient blocking flow shop scheduling with setup time. *Applied Soft Computing, 93*, Article 106343.
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.
Khorasanian, D., & Moslehi, G. (2012). An Iterated Greedy Algorithm for Solving the Blocking Flow Shop Scheduling Problem with Total Flow Time Criteria. *International Journal of Industrial Engineering, 23*, 301–308.
Koren, Y., Wang, W. C., & Gu, X. (2017). Value creation through design for scalability of reconfigurable manufacturing systems. *International Journal of Production Research, 55*, 1227–1242.
Larranga, P., & Lozano, J. A. (2001). *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. US: Springer.
Li, X. P., & Zhang, Y. (2012). Adaptive Hybrid Algorithms for the Sequence-Dependent Setup Time Permutation Flow Shop Scheduling Problem. *Ieee Transactions on Automation Science and Engineering, 9*, 578–595.
Lin, S. W., & Ying, K. C. (2013). Minimizing makespan in a blocking flowshop using a revised artificial immune system algorithm. *Omega-International Journal of Management Science, 41*, 383–389.
Lourenço, H. R., Martin, O. C., & Stützle, T. (2010). Iterated Local Search: Framework and Applications. *Handbook of Metaheurs, 146*, 363–397.
McCormick, S. T., Pinedo, M. L., Shenker, S., & Wolf, B. (1989). Sequencing in an Assembly Line with Blocking to Minimize Cycle Time. *Operations Research, 37*, 925–935.
Miyata, H. H., & Nagano, M. S. (2019). The blocking flow shop scheduling problem: A comprehensive and conceptual review. *Expert Systems with Applications, 137*, 130–156.
Montgomery, D. C. (2008). *Design and Analysis of Experiments*. John Wiley & Sons.
Moslehi, G., & Khorasanian, D. (2014). A hybrid variable neighborhood search algorithm for solving the limited-buffer permutation flow shop scheduling problem with the makespan criterion. *Computers & Operations Research, 52*, 260–268.
Nagano, M. S., Komesu, A. S., & Miyata, H. H. (2017). An evolutionary clustering search for the total tardiness blocking flow shop problem. *Journal of Intelligent Manufacturing, 30*, 1843–1857.
Nawaz, M., Enscore, E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega, 11*, 91–95.
Nouri, N., & Ladhari, T. (2018). Evolutionary multiobjective optimization for the multi-machine flow shop scheduling problem under blocking. *Annals of Operations Research, 267*, 413–430.
Pan, Q.-K., & Wang, L. (2012). Effective heuristics for the blocking flowshop scheduling problem with makespan minimization. *Omega, 40*, 218–229.
Pan, Q. K., Ling, W., Hong-yan, S., Jun-qing, L., & Min, L. (2013). A High Performing Memetic Algorithm for the Flowshop Scheduling Problem With Blocking. *Ieee Transactions on Automation Science and Engineering, 10*, 741–756.
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.
Pinedo, M. (2015). *Scheduling: Theory, Algorithms, and Systems* (fourth ed.). Springer Verlag.
Qian, B., Li, Z. C., & Hu, R. (2017). A copula-based hybrid estimation of distribution algorithm for m-machine reentrant permutation flow-shop scheduling problem. *Applied Soft Computing, 61*, 921–934.
Rajendran, C., & Ziegler, H. (1997). An efficient heuristic for scheduling in a flowshop to minimize total weighted flowtime of jobs. *European Journal of Operational Research, 103*, 129–138.
Riahi, V., Khorramizadeh, M., Newton, M. A. H., & Sattar, A. (2017). Scatter search for mixed blocking flowshop scheduling. *Expert Systems with Applications, 79*, 20–32.
Ribas, I., Companys, R., & Martorell, X. T. (2013). A competitive variable neighbourhood search algorithm for the blocking flow shop problem. *European J. of Industrial Engineering, 7*, 729–754.
Ribas, I., Companys, R., & Tort-Martorell, X. (2011). An iterated greedy algorithm for the flowshop scheduling problem with blocking. *Omega-International Journal of Management Science, 39*, 293–301.
Ribas, I., Companys, R., & Tort-Martorell, X. (2015). An efficient Discrete Artificial Bee Colony algorithm for the blocking flow shop problem with total flowtime minimization. *Expert Systems with Applications, 42*, 6155–6167.
Ribas, I., Companys, R., & Tort-Martorell, X. (2021). An iterated greedy algorithm for the parallel blocking flow shop scheduling problem and sequence-dependent setup times. *Expert Systems with Applications, 184*, Article 115535.
Ronconi, D. P. (2004). A note on constructive heuristics for the flowshop problem with blocking. *International Journal of Production Economics, 87*, 39–48.
Ronconi, D. P., & Henriques, L. R. S. (2009). Some heuristic algorithms for total tardiness minimization in a flowshop with blocking. *Omega-International Journal of Management Science, 37*, 272–281.
Ruiz, R., Maroto, C., & Alcaraz, J. (2005). Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics - Discrete optimization. *European Journal of Operational Research, 165*, 34–54.
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.
Schiavinotto, T., & Stutzle, T. (2007). A review of metrics on permutations for search landscape analysis. *Computers & Operations Research, 34*, 3143–3153.
Shao, Z., Pi, D., Shao, W., & Yuan, P. (2019). An efficient discrete invasive weed optimization for blocking flow-shop scheduling problem. *Engineering Applications of Artificial Intelligence, 78*, 124–141.
Shao, Z., Shao, W., & Pi, D. (2021). Effective constructive heuristic and iterated greedy algorithm for distributed mixed blocking permutation flow-shop scheduling problem. *Knowledge-Based Systems, 221*, Article 106959.
Shao, Z. S., Pi, D. C., & Shao, W. S. (2017). Self-adaptive discrete invasive weed optimization for the blocking flow-shop scheduling problem to minimize total tardiness. *Computers & Industrial Engineering, 111*, 331–351.
Shao, Z. S., Pi, D. C., & Shao, W. S. (2018a). Estimation of distribution algorithm with path relinking for the blocking flow-shop scheduling problem. *Engineering Optimization, 50*, 894–916.
Shao, Z. S., Pi, D. C., & Shao, W. S. (2018b). A novel discrete water wave optimization algorithm for blocking flow-shop scheduling problem with sequence-dependent setup times. *Swarm and Evolutionary Computation, 40*, 53–75.
Sioud, A., & Gagne, C. (2018). Enhanced migrating birds optimization algorithm for the permutation flow shop problem with sequence dependent setup times. *European Journal of Operational Research, 264*, 66–73.
Tasgetiren, M., Pan, Q. K., Kizilay, D., & Gao, K. Z. (2016). A Variable Block Insertion Heuristic for the Blocking Flowshop Scheduling Problem with Total Flowtime Criterion. *Algorithms, 9*, 71–95.
Tasgetiren, M. F., Kizilay, D., Pan, Q. K., & Suganthan, P. N. (2017). Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion. *Computers & Operations Research, 77*, 111–126.
Tasgetiren, M. F., Pan, Q., Kizilay, D., & Suer, G. (2015). A populated local search with differential evolution for blocking flowshop scheduling problem. In *In 2015 IEEE Congress on Evolutionary Computation (CEC)* (pp. 2789–2796).
Wang, C., Song, S. J., Gupta, J. N. D., & Wu, C. (2012). A three-phase algorithm for flowshop scheduling with blocking to minimize makespan. *Computers & Operations Research, 39*, 2880–2887.
Wang, L., Fang, C., Suganthan, P. N., & Liu, M. (2014). Solving system-level synthesis problem by a multi-objective estimation of distribution algorithm. *Expert Systems with Applications, 41*, 2496–2513.
Wang, L., Pan, Q. K., Suganthan, P. N., Wang, W. H., & Wang, Y. M. (2010). A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems. *Computers & Operations Research, 37*, 509–520.
Wang, L., Pan, Q. K., & Tasgetiren, M. F. (2011). A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem. *Computers & Industrial Engineering, 61*, 76–83.
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.
Wang, S. Y., Wang, L., Liu, M., & Xu, Y. (2013). An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem. *International Journal of Production Economics, 145*, 387–396.
Wu, C.-G., Wang, L., & Wang, J.-J. (2021). A path relinking enhanced estimation of distribution algorithm for direct acyclic graph task scheduling problem. *Knowledge-Based Systems, 228*, Article 107255.
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*, Article 100785.
Zhang, Z. Q., Hu, R., Qian, B., Jin, H. P., Wang, L., & Yang, J. B. (2022). A matrix cube-based estimation of distribution algorithm for the energy-efficient distributed assembly permutation flow-shop scheduling problem. *Expert Systems with Applications, 194*, Article 116484.
Zhao, F., Shao, D., Wang, L., Xu, T., Zhu, N., & Jonrinaldi.. (2022). An effective water wave optimization algorithm with problem-specific knowledge for the distributed assembly blocking flow-shop scheduling problem. *Knowledge-Based Systems, 243*, Article 108471.