Expert Systems With Applications 198 (2022) 116921

Contents lists available at ScienceDirect

Expert Systems With Applications

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

A discrete learning fruit fly algorithm based on knowledge for the distributed no-wait flow shop scheduling with due windows

Ningning Zhu [a], Fuqing Zhao [a,*], Ling Wang [b], Ruiqing Ding [a], Tianpeng Xu [a], Jonrinaldi [c]

[a] School of Computer and Communication Technology, Lanzhou University of Technology, Lanzhou 730050, China
[b] Department of Automation, Tsinghua University, Beijing 10084, China
[c] Department of Industrial Engineering, Universitas Andalas, Padang 25163, Indonesia

ARTICLE INFO

Keywords:
Distributed no-wait flow-shop
Due windows
Fruit fly optimization
Probability knowledge model
Variable neighborhood descend

ABSTRACT

The distributed no-wait flow shop scheduling problem with due windows (DNWFSPDW) is a novel and considerable model for modern production chain and large manufacturing industry. The object of total weighted earliness and tardiness ($TWET^{dw}$) is a common cost indicator in application. A discrete knowledge-guided learning fruit fly optimization algorithm (DKLFOA) is proposed in this study to minimize TWET in DNWFSPDW. A knowledge-based structural initialization method ($KNEH^{dw}$) is proposed to construct an effective initial solution. In the $KNEH^{dw}$, the property that the job has no waiting time between processing machines in the no-wait flow shop scheduling problem is abstracted as knowledge to instruct jobs to be placed in possible positions. The swarm center expands from a single individual to an elitist swarm in the vision search stage. A probability knowledge model is established based on the sequence relationship of jobs in the elite population. The feedback information in the iterative process using the probabilistic knowledge model leads the population to search in the direction with a high success rate. The inferior individuals are allocated to the corresponding elite individuals for the local search in the olfactory search stage. The knowledge of weight in due windows is utilized to avoid invalid search during the iteration process. The variable neighborhood descent (VND) strategy is adopted in the local search to enhance the accuracy of the proposed algorithm and jump out of the local optimal. The design of experimental method (DOE) is introduced to calibrate the parameters in the algorithm. The simulation results show that DKLFOA has advantages for solving DNWFSPDW problems comparing with the state-of-the-art algorithms.

1. Introduction

The distributed no-wait flow shop scheduling problem with due windows (DNWFSPDW) is a novel and considerable model for modern production chain and large manufacturing industry. Production scheduling, which is a significant problem in manufacturing production, allocates resources reasonably within a limited time to achieve benefits of the enterprise maximum (Springer, 2016; L Wang & Shen, 2007). As the reform and innovation of manufacturing technology, the methods of production scheduling are also updated. Production model has changed from single factory to multi-factory, and production structure shifts from centralized to decentralized. The novel multi-factory distributed manufacturing mode provides the possibility of cooperation between different factories and meets the requirement of different markets in different regions under the development tendency of economic globalization (Behnamian & Ghomi, 2016). The concept of distributed shop scheduling (DSSP) is proposed in (Toptal & Sabuncuoglu, 2010), and seven different properties are described to classify different types. Depending on the environment of machines, DSSP is divided into distributed flow shop scheduling (DFSP), distributed job shop scheduling (DJSP), distributed flexible job shop scheduling (DFJSP), and distributed assembly permutation flow shop scheduling (DAPFSP), etc.

The distributed no-wait flow shop scheduling problem (DNWFSP) (Zhao, He, et al., 2019) is the extension of DFSP, in which constraints are added to the jobs. Once a job starts to be processed, it is obliged to continue until the end of processing. No pause exists between each step of a job, that is, no waiting time is between two adjacent machines. DNWFSP is implemented in industry extensively, such as the steel industry, pharmaceutical industry and mining industry (Hall and Sriskandarajah, 1996), etc.

* Corresponding author.
E-mail address: Fzhao2000@hotmail.com (F. Zhao).

https://doi.org/10.1016/j.eswa.2022.116921
Received 20 October 2021; Received in revised form 11 January 2022; Accepted 14 March 2022
Available online 19 March 2022

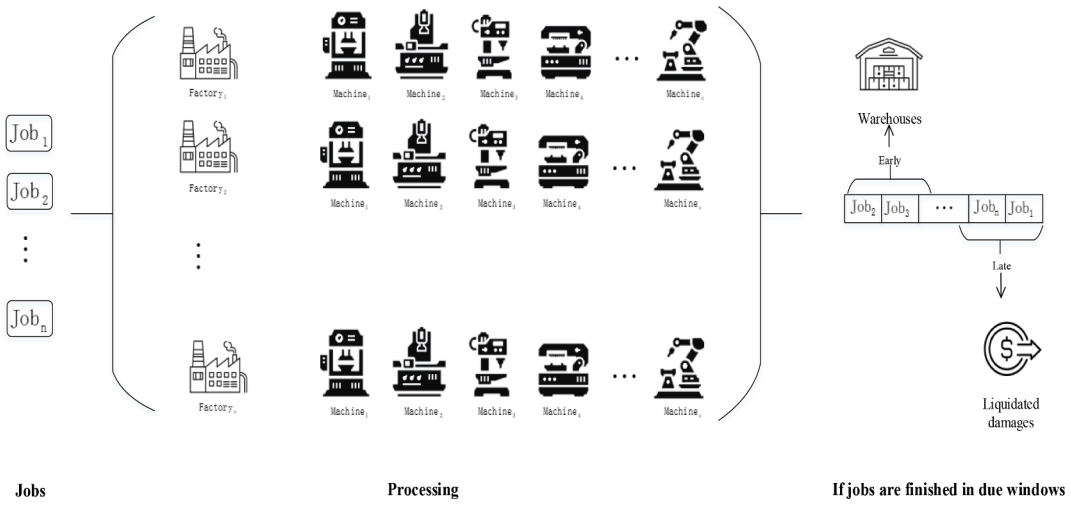


Fig. 1. The environment of the distributed processing with due windows.

In current research hotspots, the common optimization objectives include makespan (Mao et al., 2021), total flow time (Wang et al., 2010) and total tardiness time (Alidaee et al., 2021), etc. The target total weighted earliness and tardiness ($TWET^{dw}$), which is the cost of enterprises based on the finite due windows, is discussed in this study. The due windows contain the earliest due date and the latest due date. When a job is completed ahead of its earliest due date, the enterprise needs to build or lease warehouses to store the jobs, which leads to increased costs. If the job is completed later than the latest due date, the enterprise breaches the contract and should pay liquidated damages, adding to their costs (Jing et al., 2020). $TWET^{dw}$ has not been introduced into the DNWFSP problems so far. The diagram of distributed processing with the objective is shown in Fig. 1.

In some metal manufacturing and processing enterprises, the DNWFSPDW problem is widespread. For example, the coordinated scheduling problem of aluminum materials exists among multiple enterprises in the aluminum manufacturing system. The ore is smelted through electrolytic aluminum technology, then the aluminum alloy and other aluminum processing materials are produced through refining. Deeply-processed products are manufactured, such as automobile wheels. All steps in this process must be carried out at a certain temperature to ensure the processing requirements of aluminum. The temperature demand is a large proportion of energy-consuming part in the manufacturing system due to high melting point of aluminum. Once the processing of aluminum is started, it should be uninterrupted until the end to ensure that the temperature is high. The due windows are concerned in metal production system. The deeply-processed products of aluminum are relatively large in size, and they are generally stored in warehouses after processing, which increases the storage costs. When a contract is signed between a factory and a customer, the customer always specifies the due date or due windows. Overdue payment is subject to liquidated damages. Therefore, the optimization of DWFSPDW is an important scheduling problem, which is always considered in this type of metallurgical production system.

Fruit fly optimization algorithm (FOA) simulates the foraging behavior of fruit fly, which successfully forages with their keen sense of smell and vision when searching for food (Qin et al., 2022). FOA, which is a very simple method with few parameters and easy to implement, has a parallel search framework like other evolutionary algorithms. Certain heuristic and *meta*-heuristic algorithms are embedded in them, showing good performance for various scheduling problems (Zohali et al., 2019). FOA has been successfully applied to high dimensional continuous function optimization (Wang et al., 2013) and complex product scheduling problems in practice (Fu et al., 2021). The discrete learning fruit fly algorithm based on knowledge enhances the exploration and exploitation ability of the original algorithm (Zhao, et al., 2021).

There has been a great deal of research on makespan and total flow time. Some literatures have studied *TWET* in hybrid flow shop scheduling, and Jing has published a paper about *TWET* in distributed permutation flow shop scheduling problem. However, *TWET* in distributed no-wait flow shop scheduling is also a objective frequently encountered in practical production. Currently, there is no research on $TWET^{dw}$ optimization of the DNWFSP problem, but this research is of great significance for practical production. In addition, the research on FOA, which is used to optimize *TWET,* does not exist, and also the study on FOA for solving distributed no-wait flow shop scheduling is also blank. The objective $TWET^{dw}$ is introduced in the distributed no-wait flow shop scheduling with due windows (DNWFSPDW) in this paper. FOA with collaborative learning is utilized to solve the DNWFSPDW problem. The shortcoming of the original FOA is modified in the process of improving the algorithm, which avoids the single point search of the population. The algorithm balances the exploration and exploitation by expanding the single population center and combining with the efficient olfactory local search in the search process, effectively solving the DNWFSPDW. The contributions are described as follows.

- The objective is analyzed in the initial stage. The hidden knowledge in the objective function is combined with the exclusive knowledge in the no-wait problem. The constructive initialization method $KNEH^{dw}$ is proposed based on the two kinds of knowledge. The initial population center is constructed effectively.
- The population center is redefined as an elite population in the vision search stage. A probability model is constructed by collecting and summarizing the laws of previous generations and combining the information of due windows in the problem. The elitist population is updated through the feedback of probability information. The learning estimation of distribution algorithm (EDA) is predicted by collecting and analyzing sample points, and the global search is guided efficiently by the vision search.
- The knowledge of weight in due windows is utilized to avoid invalid search in the olfactory stage. The variable neighborhood descent strategy is used to guide the inferior individuals to local search around the elite individuals in the population center. Three kinds of neighborhood structures based on due window information are designed. The switching of neighborhood structures assists the population to escape from the local optimum.

The remainder of this paper is organized as follows. A literature review of related works is given in Section 2. The problem and objective function of this study are formulated in Section 3. The proposed

algorithm DKLFOA is presented in detail in Section 4. Experimental results and discussion are mentioned in Section 5. Finally, the conclusion and future work are summarized in Section 6.

2. Related works

The flow shop scheduling problem (FSP), which is the most common mode, has been applied in industry widely (D. Z. Zheng & Wang, 2003). In the flow shop scheduling problem, each job is processed in the fixed same procedure. The traditional FSP is a classical NP-hard problem (Lu et al., 2021). Many precise methods are utilized to solve this kind of problem, such as branch and bound Method, Lagrangian relaxation methods (Fisher, 1981), etc. Although the results of small-scale problems are accurate, it is difficult to accept temporal and spatial complexity especially in large scale problems and practical industrial production problems. In recent years, evolutionary algorithms have become hot pots in solving complex industrial optimization problems. For high dimensionality and complex engineering problems, mathematical models cannot be established and derivatives or integrals cannot be carried out. In this case, traditional precise methods are not suitable for application and are subject to many restrictions, evolutionary optimization algorithms guide the search process of feasible solutions by imitating a set of rules in nature (Eberhart, 2001). The efficiency of searching has improved greatly. For instance, a hybrid particle swarm optimization (PSO) (Qian et al., 2009) algorithm is proposed to solve the flow shop scheduling with limited buffers. A discrete harmony search (HS) algorithm (Gao et al., 2011) is studied to optimize the total flow-time in no-wait flow shop scheduling problem. For solving no-wait flow shop scheduling problem, a discrete artificial bee colony algorithm (DABC) (Li, Li, Gao, 2020; Li, Li, Gao, Zhang & Meng, 2020) cooperated with the variable neighborhood descent (VND) algorithm, is used in scout bee phase. To optimize the makespan, DABC is combined with knowledge of machine position (Li, Li, Gao, 2020; Li, Li, Gao, Zhang & Meng, 2020). A two-level encoding method is cooperated with hybrid neighborhood operator. Biogeography-based optimization algorithm (BBO) (Zhao, Qin, et al., 2019) with variable neighborhood search (VNS) strategy is used to solve no-wait flow shop scheduling problem. The lot-stream problem is addressed by the artificial bee colony algorithm, and an external archive of actions effectively guide the search for solutions in (Pan et al., 2011). A new priority rule is connected with the heuristic algorithm presented by Nawaz, Enscore and Ham (NEH), and the embedded water wave algorithm (WWO) (Zhao et al., 2020) to solve no-idle flow shop scheduling problem with total tardiness criterion. A distributed differential evolutionary (HMOEA/DE) algorithm is proposed in (Zhang et al., 2019) to optimize both the makespan and tardiness at the same time. Fernando Luis Rossi (Rossi & Nagano, 2020) analyzes the mixed no– idle flow shop scheduling problem by adding sequence-related settings and proposes a new heuristic, BS-ICH with a local acceleration method and a mixed integer linear programming (MILP) model for total tardiness minimum. Many literatures have studied FSP based on problem characteristics and experiments have proved that swarm intelligence optimization algorithm achieves better results than precise solution method, so it is more suitable for solving FSP problem. Different swarm intelligence optimization algorithms, such as PSO, DABC, BBO, etc, have taken advantage of different improvement methods by adding knowledge or acceleration strategy to solve the makespan, total flow time or total tardiness of NWFSP and NIFSP problems, and obvious effects have been achieved, which provides important reference ideas for subsequent research.

Based on FSP constraints, the distributed flow shop problem considers which factory the job is processed in, and the processed sequence in every factory simultaneously. Therefore, the search space is further enlarged, and the complexity is increased in DFSP. Obviously, DFSP is a NP-hard problem. The distributed permutation flow shop (DPFSP) scheduling problems are the most representative among numerous studies on DFSP. The classical permutation flow shop (PFSP) scheduling is considered in multi-factory in DPFSP. Six mixed integer linear programming models (MILP) of DPFSP are proposed by the research (Naderi & Ruiz, 2010). In the study (Quan Ke Pan et al., 2019), three constructive heuristics and four metaheuristics are proposed to optimize the makespan in the distributed permutation flow shop. The proposed knowledge-based scheduling method obtains the optimal solution quickly and effectively. A distributed knowledge-based cooperative algorithm is described in the research (Wang & Wang, 2018) to resolve energy efficiency scheduling in distributed permutation flow shop to balance the makespan and total energy consumption. Besides DPFSP, other distributed flow shop scheduling problems are also concerned. (Tasgetiren et al., 2019) proposes three metaheuristic algorithms and a speed scaling approach fit for various speed levels of the jobs to solve the Energy-efficient No-idle Flow-shop scheduling problem. Zhao et al. (2021) addresses a discrete Jaya algorithm with self-learning operation selection strategy and energy-saving strategy to minimize the total tardiness, total energy consumption and factory load balancing for the energy-efficient distributed no-idle flow shop scheduling problem in a heterogeneous factory system. (Chen et al., 2019) addresses the energy-efficient distributed no-idle permutation flow shop scheduling problem (EEDNIPFSP) to minimize the makespan and total energy consumption simultaneously, a collaborative optimization algorithm (COA) and some collaborative mechanisms are utilized for population initialization, and a speed adjusting strategy for the non-critical operations improves total energy consumption. For solving the distributed blocking flow shop scheduling problem, a distributed differential evolution (DE) (G. Zhang et al., 2018) is combined with elitist retain strategy to minimize the makespan. In the distributed no-idle flow shop problem studied by (Zhao, Zhang, Cao, 2020), Q-learning strategy is united in the structure of VNS, and water wave algorithm is utilized to optimize the maximum assembly completion time. To minimize the makespan in distributed limited-buffer flow shop scheduling, a metaheuristic based on DE, which is employed in (Zhang & Xing, 2019), has superior performance compared with existing heuristic algorithm. An adaptive cocktail decoding mechanism is combined with self-tuning iterative greedy algorithm (Ying et al., 2018) to solve the distributed hybrid flow shop scheduling problem. Four neighborhood search strategies, which are embedded in the distributed discrete artificial bee colony algorithm, have a higher performance in the distributed heterogeneous no-wait flow shop scheduling (Li, Li, Gao, 2020; Li, Li, Gao, Zhang & Meng, 2020) compared with current state-of-the-art algorithms. A shuffled frog-leaping algorithm (SFLA) (Cai et al., 2020) and a memeplex quality factor are used to optimize the two objectives of makespan and total tardiness in distributed hybrid flow shop scheduling problem. A Pareto-based estimation of distribution algorithm (PEDA) (Shao et al., 2019) is applied to solve the distributed no-wait flow shop scheduling problem with sequence-dependent setup time to optimize the makespan and the total weight tardiness at the same time. The literatures mentioned above adopt different optimization algorithms and various strategy mechanisms to optimize the makespan, total energy consumption or total tardiness according to the corresponding problem characteristics of the DFSP problem. The simultaneous optimization of the two objectives is completed in some literatures. Good experimental results are achieved.

In the current research, the objective $TWET^{dw}$ is commonly seen in hybrid shop scheduling problem. In the research (Khare & Agrawal, 2019), three kinds of metaheuristics hybrid squirrel search algorithm (HSSA), opposition-based whale optimization algorithm (OBWOA), and discrete grey wolf optimization (DGWO) are proposed. The proposed method has the best performance in scheduling hybrid flow shop with sequence-dependent comparing with other seven *meta*-heuristics algorithm. Pan (Pan et al., 2017) proposes an Iterated Greedy (IG) algorithm and iterative local search in hybrid flow shop scheduling problem. The employed optimal idle time insertion operation minimizes the target efficiently. A discrete colonial competitive algorithm is applied in hybrid flow shop scheduling to optimize earliness and quadratic tardiness penalties. A discrete artificial bee colony algorithm, associated with

local search to obtain superior performance, is proposed in (Pan, 2012) for minimizing $TWET^{dw}$ in lot flowshop problem. As a practical industry target, $TWET^{dw}$ is introduced in distributed permutation flow shop scheduling problem by Jing in the research (Jing et al., 2020), where five constructive methods and an improved IG algorithm are employed, and idle time insert operation is applied to further improve the objective. To optimize the objective *TWET* of the HFSP problem, HSSA, IG, BWOA and other swarm intelligence optimization algorithms have been improved on different levels, and obvious results have been achieved in experiments. However, compared with studies on makespan and total flow time, there are relatively few studies on the objective *TWET* except for *TWET* of the HFSP problems. Jing have solved $TWET^{dw}$ of the DPFSP problem through a improved IG algorithm. It provides ideas and inspiration for minimizing the objective $TWET^{dw}$ of the DNWFSP problem studied by this paper.

Fruit fly optimization algorithm (FOA), which is a typical evolutionary algorithm proposed by PAN (Pan, 2012) in 2012, has attracted extensive attention to address scheduling problem. A distributed FOA algorithm is incorporated with IG algorithm for flexible processing time in steelmaking casting systems of hybrid flow shop rescheduling problem in (Li et al., 2016). A multi-swarm fruit fly algorithm named hybrid enhanced discrete fruit fly optimization algorithm (HEDFOA) (Shao et al., 2019) is proposed by Shao for solving the block flow shop scheduling problem. The knowledge-based FOA (KBFOA) (Zheng & Wang, 2018) associated with new encoding and decoding methods is applied to the dual resource constrained flexible job-shop scheduling problem, and efficient results are obtained. Modified fruit fly optimization algorithm (MFFO) (Han et al., 2016) is utilized to minimize the makespan in the blocking flow shop scheduling problem combined with a speed-up neighbor structure. A two-stage adaptive FOA (TAFOA) (Zheng & Wang, 2016b) is devoted to minimize the makespan of unrelated parallel machine scheduling problem with additional resource constraints (UPMSP_RC), extensive numerical comparisons show the effectiveness of the TAFOA in solving the UPMSP_RC. Incorporating with knowledge-guided search, the experiments show that TAFOA is potential to optimize this problem. Collaborative multi-objective fruit fly optimization algorithm (CMFOA) (Zheng & Wang, 2016a) has an efficient performance in balancing the makespan and Total Energy Consumption (TEC) for resource-constrained green scheduling of unrelated parallel machine. A whale-inspired hunting strategy (WFOA) (Fan et al., 2020) replaces the random search plan of the original FOA, and the experiments have validated its effectiveness. A hierarchical guidance strategy, assisted fruit fly optimization algorithm with cooperative learning mechanism (HGCLFOA), is proposed by (Zhao, Ding, et al., 2021). The local hierarchical guidance strategy and the mixed Gaussian distribution estimation strategy are added respectively to the olfactory and visual search stages, the exploration and exploitation are balanced, and the experimental results demonstrate that HGCLFOA outperforms the classical FOA and the state-of-the-art variants of FOA. An improved FOA with bat sonar strategy (BSSFOA) proposed by Fan (Fan et al., 2021), is employed to enhance the exploration, and hybrid distribution combined Gaussian distribution with student distribution to improve the exploitation, and the results of a comprehensive set of benchmark functions with the continuous version and a discrete version of BSSFOA for a searching mechanism in the feature selection assess the ability of the BSSFOA to search the best performing features. The discrete FOA is united with stochastic simulation (Deng & Wang, 2017) to optimize the makespan and total tardiness at the same time. A high performance is achieved by the proposed algorithm in comparison of twenty-five instances in the environment of the stochastic multi-objective integrated disassembly-reprocessing-reassembly scheduling. The improved FOA algorithm adds special strategy or combines with different evolutionary algorithms for solving HFSP, BFSP and other scheduling problems. The experimental results have been verified by the research mentioned above. However, compared with extensive research results of makespan and total flow time, there are few research on *TWET*. The literatures on the $TWET^{dw}$ optimization of the DNWFSP problem does not exist at present, and the research on FOA, which is used to optimize *TWET* and solve distributed zero-wait flow shop scheduling problem is also blank. This paper proposes an improved FOA algorithm to solve $TWET^{dw}$ of the DNWFSP problem for the first time. This algorithm approaches knowledge to the whole process, including the initialization stage, the visual search stage with EDA mechanism, and olfactory search stage cooperated with VND strategy.

3. The description of problem

The notation of this research is presented as follows.

Notations	Illustration
n	The number of jobs
j	The index of jobs
f	The index of factory
m	The number of machine
i	The index of machine
$f^{'}$	The number of factory
J	The set of all jobs
M	The set of all machines
F	The set of all factories
$P_{j,i}$	The processing time of job j in machine i
$O_{j,\mathrm{m}}$	The operation of job j in machine m
d_j^-	The earliest due date of job j
d_j^+	The latest due date of job j
E_j	The earliness time of the job j
T_j	The tardiness time of the job j
w_j^E	The unit inventory cost of job j
w_j^T	The unit liquidated damage of job j
$D(j-1,j)$	The distance between job j and $j-1$ (the difference of completion time of job j and $j-1$)
$TWET^{dw}$	The Total Weighted Earliness and Tardiness
$X_{i,k,f}$	A binary variable which is 0 or 1
$C_{k,j,f}$	The completion time of the job j processed on the position k in machine f

3.1. Problem definition

The DNWFSPDW is described in detail as follows:

n different jobs $J=\{J_1,J_2,\cdots,J_n\}$ are processed in m machines $M=\{M_1,M_2,\cdots,M_m\}$. $f^{'}$ factories $F=\{f_1,f_2,\cdots,F_{f^{'}}\}$ share the processing tasks of the n jobs. Each job is only operated in one factory at most, and each factory has M machines. All jobs are independent. The processing times of each job on each machine are pre-deterministic non-negative. The setup times of machines and transportation times between operations are negligible. Each job must be completed without interruption once a machine starts to process it. Preemption is not allowed. After a job is assigned to a factory, all the processing operations of the job are carried out in the factory and it is no longer transferred to other factories. Each job can be assigned to any one of factories for processing. The processing times of a job at one specific machine are the same from factory to factory. The processed steps of each job on the machine are the same, and one machine only processes one job at a time. Once a job is operated, no intermittent exists between each step in the sequence of operation $[O_{j,1},O_{j,2},\cdots,O_{j,m}]$. Each job has different due windows, beyond which the cost will increase. Early complement causes incensement of inventory costs, while delayed completion leads to liquidated damage.

Objective: minimize.

$$TWET^{dw} \tag{1}$$

Subject to:

$$\sum_{k=1}^{n}\sum_{f=1}^{F} X_{i,k,f}=1, i\in\{1,2,3\cdots\mathrm{m}\} \tag{2}$$

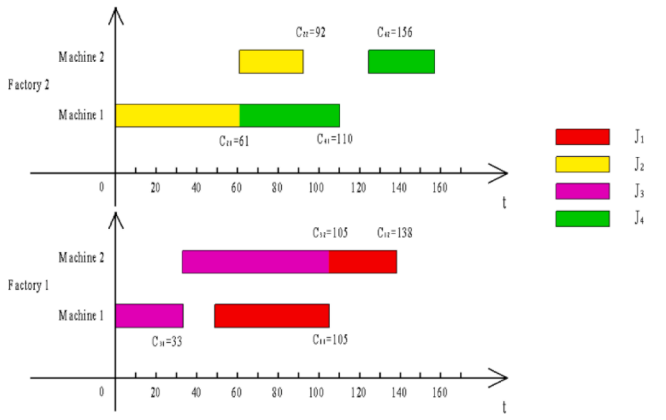


Fig. 2. Gantt chart of the example.

$$\sum_{i=1}^{n}\sum_{f=1}^{F} X_{i,k,f} = 1, k \in \{1,2,3\cdots m\} \tag{3}$$

$$C_{k,j,f} = C_{k,j-1,f} + \sum_{i=1}^{n} X_{i,k,f} \bullet \mathrm{P_{i,j}}, \mathrm{k} \in \{1,2,\cdots \mathrm{m}\}, \mathrm{j} \in \{2,\cdots \mathrm{n}\}, \mathrm{f} \in \{1,2,\cdots \mathrm{f'}\} \tag{4}$$

$$C_{k,j,f} \geq C_{k-1,j,f} + \sum_{i=1}^{n} X_{i,k,f} \bullet \mathrm{P_{i,j}}, \mathrm{k} \in \{2,3,\cdots \mathrm{m}\}, \mathrm{j} \in \{1,2,3\cdots \mathrm{n}\}, \mathrm{f} \in \{1,2,3\cdots \mathrm{f'}\} \tag{5}$$

$$E_j \geq 0, j \in \{1,2,3\cdots n\} \tag{6}$$

$$T_j \geq 0, j \in \{1,2,3\cdots n\} \tag{7}$$

$$C_{k,j,f} \geq 0, k \in \{1,2,\cdots m\}, j \in \{1,2,\cdots n\}, f \in \{1,2,\cdots f'\} \tag{8}$$

$$TWET^{dw} \geq 0, k \in \{1,2,\cdots m\}, f \in \{1,2,\cdots f'\} \tag{9}$$

$$X_{i,k,f} \in \{0,1\}, k \in \{1,2,\cdots m\}, j \in \{1,2,\cdots n\}, f \in \{1,2,\cdots f'\} \tag{10}$$

$$E_j = \max\left(d_j^- - C_{k,j,f}, 0\right), k \in \{1,2,\cdots m\}, j \in \{1,2,\cdots n\}, f \in \{1,2,\cdots f'\} \tag{11}$$

$$T_j = max\left(C_{k,j,f} - d_j^+, 0\right) k \in \{1,2,\cdots m\}, j \in \{1,2,\cdots n\}, f \in \{1,2,\cdots f'\} \tag{12}$$

$$TWET^{dw} = \sum_{j=1}^{n} (W_j^E \bullet \mathrm{E_j} + \mathrm{W_j^T} \bullet \mathrm{T_j}) \tag{13}$$

Constraint set (2) ensures that each job should be assigned to only one factory and one position at the assigned factory. Constraint set (3) states that n positions among all the possible nF must be occupied. Constraint set (4) illustrates the relationship between the completion time of each job on two sequential machines in the assigned factory, ensuring the no-wait constraint. Constraint set (5) illustrates each job is started only after the previous job assigned to the same machine at the same factory has been completed. Constraint set (6) guarantees the earliness time of the job j is non-negative. Constraint set (7) guarantees the tardiness time of the job j is non-negative. Constraint set (8) guarantees that the completion time of each job is non-negative. Constraint set (9) ensures that the total weighted earliness and tardiness with due windows for each job is non-negative. Constraint set (10) defines the binary decision variables which belongs to 0 or 1. Constraint set (11) states the premature time of the job. Constraint set (12) states the tardiness delivery time of job. Constraint set (13) guarantees the total cost that the enterprise needs to pay for these early or delayed jobs.

3.2. The calculation method of objective

Each job $j \in N$ has its own due windows $[d_j^-, d_j^+]$ according to section 5.1. d_j^- is the earliest delivery date. If the completion time C_j of job j happens before the earliest delivery date d_j^-, it is stored in a warehouse, so the enterprise needs to pay corresponding inventory costs for the job. The premature time of the job is $E_j = max(d_j^- - C_j, 0)$, and the unit inventory cost of the job is W_j^E, so, the total premature cost of job j is $W_j^E * E_j$. Similarly, d_j^+ is the latest delivery date of job j. If job j isn't processed on time, the enterprise needs to pay corresponding liquidated damages. The tardiness delivery time of job j is $T_j = max(C_j - d_j^+, 0)$, and the unit liquidated damage is W_j^T, so the corresponding liquidated damage that the enterprise has to pay is $W_j^T * T_j$. The total cost that the enterprise needs to pay for these early or delayed jobs is $TWET^{dw} = \sum_{j=1}^{n}(W_j^E * E_j + W_j^T * T_j)$ for the processing of the group of jobs. $TWET^{dw}$ is the common optimization goal in due window problem and the research objective in this study.

The overall calculation steps are shown in the Eqs. (11) - (13). A specific example of calculating $TWET^{dw}$ in DNWFSPDW is presented as follows Fig. 2. In 2th example, four jobs are allocated to two factories for processing, each with two machines. The due windows of four jobs are achieved by Eq. (15). The processed time of each job is listed in Eq. (16). Finally, the objective function value is calculated as Eq. (17).

$$\begin{pmatrix} w_j^E \\ w_j^T \end{pmatrix} = \begin{pmatrix} 4 & 2 & 5 & 1 \\ 2 & 3 & 1 & 4 \end{pmatrix} \tag{14}$$

$$\begin{pmatrix} d_j^- \\ d_j^+ \end{pmatrix} = \begin{pmatrix} 98 & 96 & 99 & 114 \\ 126 & 139 & 117 & 123 \end{pmatrix} \tag{15}$$

$$(P_{j,i})_{4\times 2} = \begin{pmatrix} 56 & 33 \\ 61 & 31 \\ 33 & 72 \\ 49 & 31 \end{pmatrix} \tag{16}$$

$$TWET^{dw} = 12 \times 2 + 4 \times 2 + 0 + 33 \times 4 = 164 \tag{17}$$

4. The proposed novel FOA for DNWFSPDW

4.1. The basic fruit fly optimization algorithm

The fruit fly is a common kind of insect in daily life. The food source is quickly found due to the super olfaction and vision for foraging behavior. The individual judges the food source by olfactory sense, and compares the smell concentration with other partners in the fruit fly swarm. They approach the individual who is nearest to the food by the visual sense. This process is iterated until the food is found.

The basic FOA steps are described as follows:

Step 1: Initialization.

Initialize all parameters, including population size, problem dimension, population location region, the maximum number of evaluations, and random location.

Step 2: Olfaction search stage.
Step 2.1 The direction and distance of fruit fly individual are given for foraging randomly.
Step 2.2 Calculate the distance to the origin since the location of the food source is unknown.
Step 2.3 Estimate the smell concentration.
Step 2.4 Substitute the smell concentration of the objective function to obtain the fitness value.

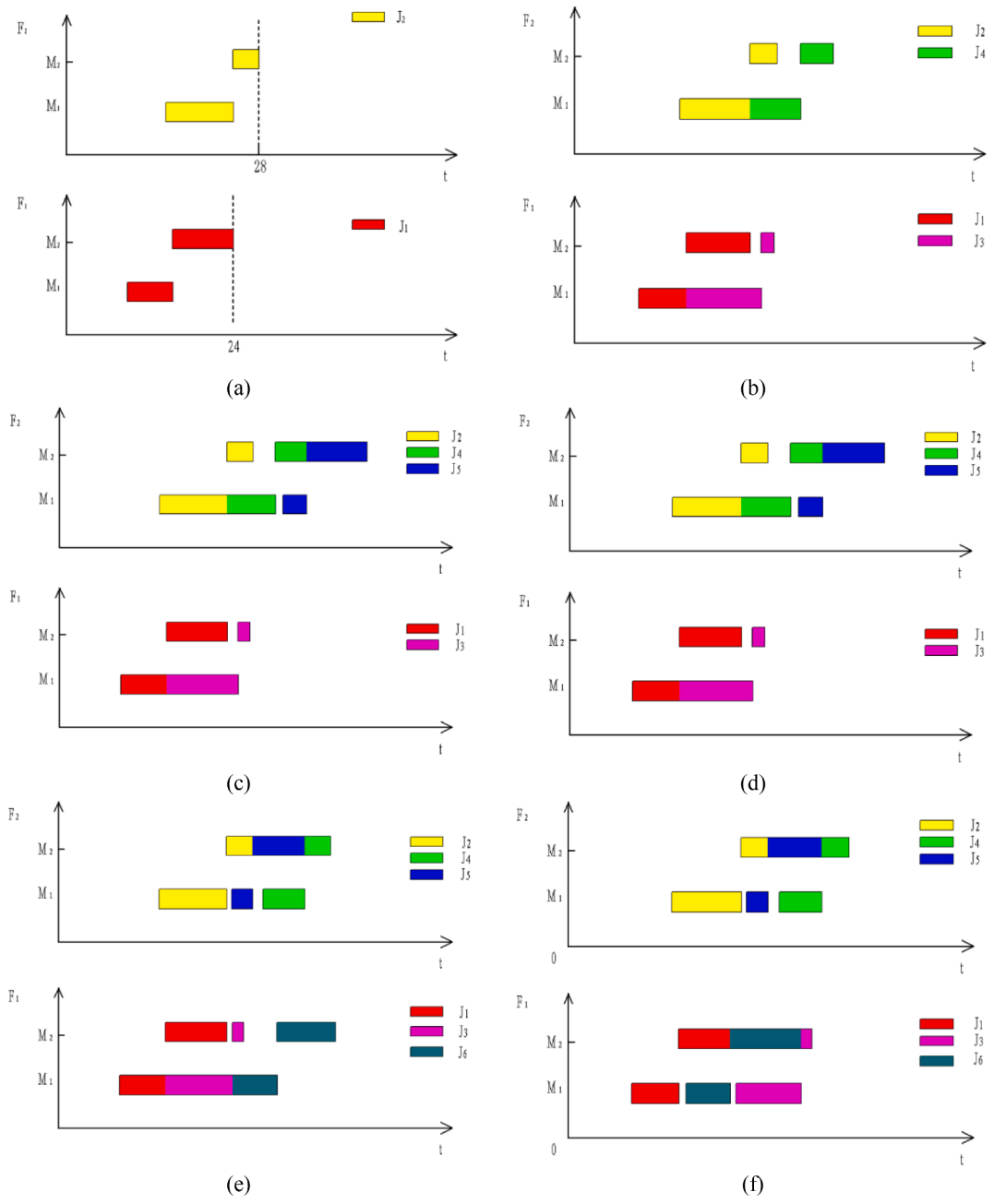


Fig. 3. The process of initialization.

Step 3: Vision search stage.

Record the best location of every generation and visually approach the current best location.

Step 4: Check termination condition.

Finish the loop and return the best solution if the termination condition is satisfied or converges to a fixed value. Otherwise, continue the iterative process and repeat steps 2 to 3.

4.2. The representation of solutions

A sequence of f is represented as a solution in each factory. The sequence of the factory k is represented by $\pi_k = \{\pi_{k,1}, \pi_{k,2}, \cdots, \pi_{k,n_k}\}$, where $k = 1, 2, ..., f$, $j = 1, 2, ..., n_k$, $\pi_{k,j} \in N$ represents the job j in π_k, n_k represents the total number of jobs assigned to factory k, and $\sum_{k=1}^{f} n_k = n$. A set of f sequences $\pi = \{\pi_1, \pi_2, ..., \pi_f\}$ represents a solution.

4.3. Initialization

A knowledge-based structural initialization method ($KNEH^{dw}$) is proposed according to the characteristics of no-wait flow shop scheduling problem and the composition of DNWFSPDWP objective function.

The due windows are generated by (Jing et al., 2020). Through experiments and analysis, the due windows are the scaling of C_{max} calculated by NEH adapted for DPFSP by Naderi and Ruiz (NEH2) (Deng & Wang, 2017). The due windows are narrow in the Taillard problem and most of jobs are difficult to get done on time. Only a small part of jobs is early completed, and most of jobs are late, which cause liquidated damages, at this time, it is necessary to arrange the job as close to its d_j^+ when the processing is completed. The problem characteristics of the no-

wait flow shop scheduling problem are that there is no waiting time between the processing of the jobs on the continuous machine. Therefore, once a job with long processing time is prioritized, almost all the completion of subsequent jobs will be delayed. On the basis of the original tardiness, the distance of the delayed delivery date d_j^+ of the job is extended again, which increases the cost of each job.

A completion time difference exists between the jobs. The characteristics of no-wait flow shop scheduling problem are abstracted as the knowledge to guide the job allocation to each factory. According to the previously inserted job, the jobs in the remaining sequence with the closest completion time difference to the last inserted job are preferentially processed. This method ensures that the current job is processed on time and reduces the influence on the subsequent delayed jobs.

The objective function $TWET^{dw}$ is combined with three parts, including the weight of early completion, late completion and the on-time completion. The objective function $TWET^{dw}$ equals to 0 when a job is finished in the due windows on time, and the processing cost is minimized to the greatest extent. Thus, the most important rule is to make the job satisfy the requirement of due windows as much as possible. The job with the earliest delivery date d_j^- is chosen as the first processed job unlike the Largest Processing Time rule (LPT). The processing is required to start from a specific time $(d_j^- - p_{j,1})$, not from zero, so that it can meet the requirements of the job in the due windows to complete the processing, and leave the maximum insertion space for the subsequent sequence.

According to the initialization method, a FOA population is generated, and the job sequence is updated based on the population in subsequent visual search and olfactory search. Other individuals in the fruit fly are randomly generated. An example is shown to describe the construction process by the Gant chart in Fig. 3. The pseudocode of $KNEH^{dw}$ is shown in Algorithm 1. The steps of the *KNEH* method are as follows.

Step1: The jobs with the earliest delivery date d_j^- are considered to give priority as the first processed job in each factory f_k. It is shown in the Fig. 3 (a).
Step2: The job closest to the last inserted job gets priority to be placed in the current factory. It is shown in the Fig. 3 $(b)\rightarrow(c)$.
Step3: The distance is adjusted between the job and the due window after the job is inserted. It is shown in the Fig. 3 $(d)\rightarrow(f)$.
Step4: Steps 2–4 are repeated until all the jobs are inserted to factories and the sequence τ is obtained.

Algorithm 1.

Algorithm 1. ***KNEH^{dw}***

1 $\tau_d \leftarrow$ *Sort the jobs according to* d_j^- *in ascending order*
2 ***for*** $j = 1$ ***to*** *n in* τ_d
3 ***if*** *job j is completed in its due windows (not conflict with other jobs)*
4 *Insert the job to* τ *in its due windows according*
5 ***else***
6 $j = j + 1$
7 ***endif***
8 ***endfor***
9 ***for*** $j = 1$ ***to*** $popsize - sizeof(\tau_d)$
10 *find the job with minimized distance to the last job of* τ
11 *Insert the job to the sequence* τ
12 ***endfor***
13 ***Output the initialized solution*** τ

4.4. The learning vision search based on experience accumulation

The population center is learned by all individuals during the vision search stage. The population center plays a crucial role in balancing the exploration and exploitation. An efficient EDA algorithm is employed in elitist individuals for enhancing the diversity of the population in the olfactory-based search stage to guide the visual search.

The distribution of individuals in the search space in EDA algorithm is predicted using the probability model by collecting and analyzing sample points (Liang et al., 2018). The experience knowledge is used to establish the probability model, which reflects the regulation of superior individuals in previous generations. In the process of solving combinatorial optimization problems, the relationship between elements in optimal solutions is analyzed, and the valid experience is kept by increasing the probability of specific elements in the solution space. EDA consists of three steps: sample point, the establishment of probability model and updating of solutions. The establishment of probability by the sampled superior individuals is the most significant in EDA algorithm.

The probability model is established by the superior individuals in the population center regarding with the characteristics of centralized guidance in FOA algorithm. The population search is carried out in the direction of better priority, which obviously improves the search efficiency compared with the original blind search.

The minimization of $TWET^{dw}$ for the DNWFSPDW is taken as the target of optimization in this study. Three kinds of knowledge, which are the unit weight W, the distance D between front and rear jobs and the distance d from the due windows, have influence on the objective functions in $TWET^{dw}$. A probability matrix Q is constructed to describe the combinatorial optimization problem due to the dispersion of the scheduling problems.

Algorithm 2.

Algorithm 2. ***Learning vision search***

1 $\tau_s \leftarrow$ *Sort the jobs according to the value of* $TWET^{dw}$ *in descending order*
2 ***for*** $j = 1$ ***to*** *halfofpopsize*
3 ***if*** *generation*==1
4 *Initialize the probability model according to Eqs.* (19) - (20)
5 *Update the probability matrix Q*
6 ***else***
7 *Update the probability matrix Q according to Eqs.* (21) - (22)
8 ***endif***
9 ***endfor***
10 ***for*** $j = 1$ ***to*** *halfofpopsize*
11 *The priority sequence* α *is generated according to probability matrix Q*
12 ***endfor***
13 ***for*** $j = 1$ ***to*** *n in sequence* α
14 *Insert the job to the last position in different factories*
15 *Insert the job to the best factory*
16 ***endfor***
17 ***Output*** *the Ne individuals*

$$Q = \begin{pmatrix} q_{11}(l) & q_{12}(l) & \cdots & q_{1n}(l) \\ q_{21}(l) & q_{22}(l) & \cdots & q_{2n}(l) \\ \vdots & \vdots & \vdots & \vdots \\ q_{n1}(l) & q_{n1}(l) & \cdots & q_{nn}(l) \end{pmatrix} \tag{18}$$

where $q_{ij}(l)$ is the probability that job J_j is arranged before the *ith* position in the *lth* generation. If the probability $q_{ij}(l)$ is large, job J_j has a high priority to be inserted into the sequence. The search experience is lacking during the initialization stage. Equal probability is assigned to all jobs. All the $q_{ij}(l)$ in matrix Q is defined as $q_{ij}(0) = 1/n$, which means each job is inserted in sequence with the same priority.

A machining sequence, which determines the processing priority, is generated by the probability matrix Q through probability sampling in the search space at each generation. In the process of updating, three factors are considered in each position, and knowledge is transformed to probability. The job is selected to be processed preferentially in this position as far as possible to reduce the impact on the subsequent jobs.

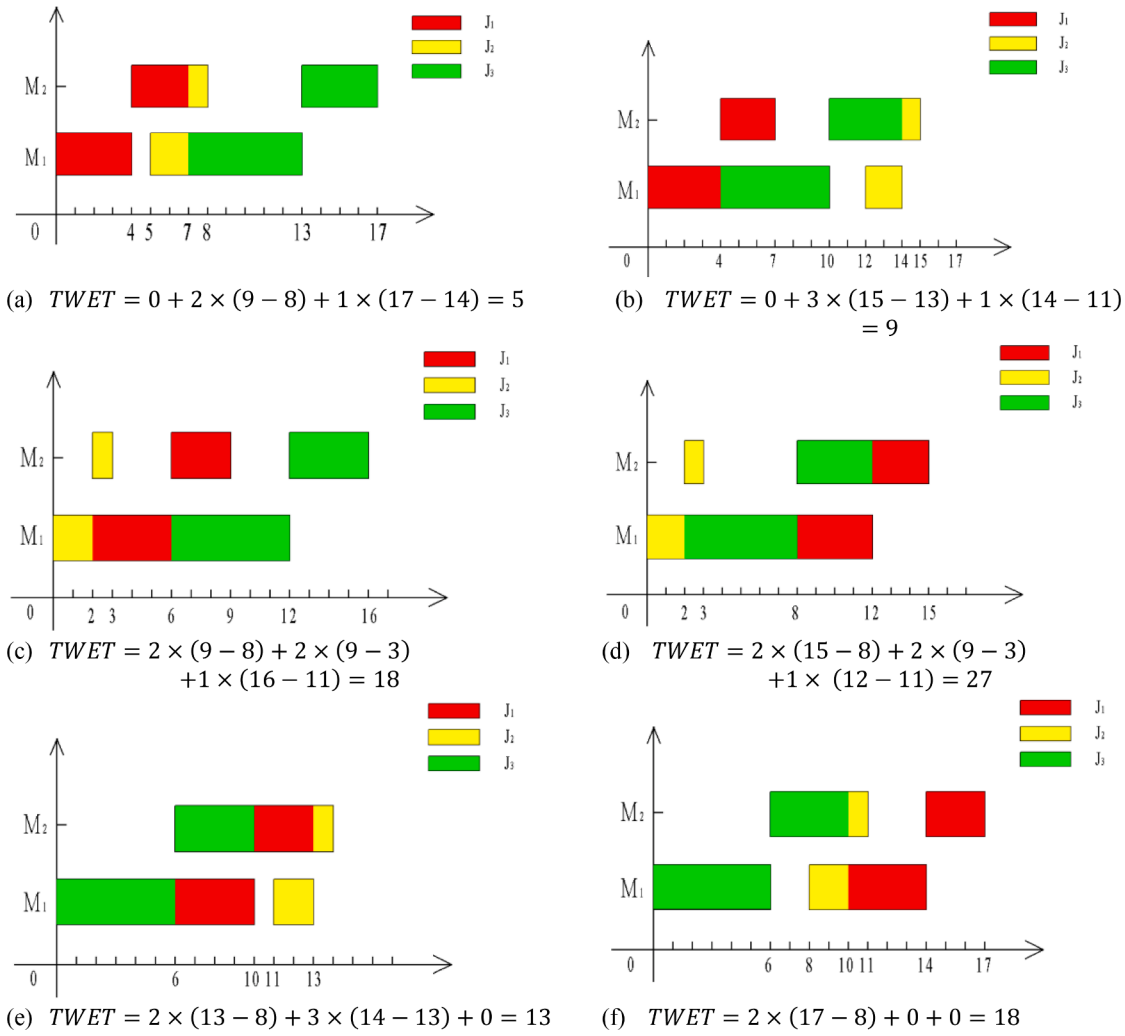


Fig. 4. The process of the learning vision search.

The shorter the distance from the previous job is, the less cost is increased by $TWET^{dw}$ for subsequent jobs. Thus, the job with small distance D from the previous job is processed preferentially. The unit weights are defined as the degree of punishment. The unit tardiness weight W^T plays an important role in the value of objective function $TWET^{dw}$ according to the characteristics of DNWFSPDW. The job with the big unit tardiness weight has the priority. The distance d from the due window is simplified as a measure of punctuality emphasizing the position of probability matrix Q. Job J_j in position i is selected with the probability $q_{ij}(l)$. $q_{ij}(l)$ is defined in detail as follows:

$$q_{ij}(l) = \begin{cases} q_{ij}(l), i = 1 \\ \dfrac{\delta(i,j)^* q_{ij}(l)^* D(i,j)^* W_j^T}{\sum_{a=1}^{n} \delta(i,a)^* D(i,a)^* W^T(a)^* q_{ia}(l)}, i > 1 \end{cases} \tag{19}$$

where $D(i,j)$ represents the distance of job j from the job in position $i-1$. $\delta(i,j)$ is the on-time enhancing factor defined as follows:

$$\delta(i,j) = \begin{cases} \mu, if J_j in position i is on time complement \\ 1, otherwise \end{cases} \tag{20}$$

where $\mu(\mu > 1)$ is utilized to enhance the degree of the probability. The importance of punctuality is emphasized by the enhancement factor, which guarantees that the job completed on time is given a higher priority. This is true of only a small part of the jobs in the whole sequence of DNWFSPDW. Overemphasis on the factor leads to the decrease of population diversity and the optimal solution isn't obtained. The small value of μ relaxes the constraint of punctuality. The specific value μ is adjusted in subsequent experiments. If job J_j has been assigned to the sequence, the probability of J_j in matrix Q is set to 0, and the probability of remaining jobs is normalized to 1 in the $i-th$ row.

When all the elements in matrix Q are set to 0, the whole jobs are inserted into the priority sequence α. A greedy assigned principle, in which the jobs are inserted to the factories with the earliest completion time in the order of sequence α, is applied. The delay of subsequent jobs is reduced and the cost of $TWET^{dw}$ is superior when completing the processing of jobs as early as possible according to the objective $TWET^{dw}$. Half of the size of population is selected as elite individuals in each generation according to the sampling points to guide the updating of the population.

The population approaches to the potential area step by step under the guide of probability model. The probability model is adjusted and updated through new superior individuals in each generation. Half of *popsize* elitist individuals *Ne* are selected through the sort of fitness to update the probability matrix Q. At the same time, a part of historical information, which makes full use of individual information in the population through the feedback d to guide the search, is retained. The detailed updating process is shown as follows.

$$q_{ij}(l+1) = (1-\theta) q_{ij}(l) + \frac{\theta}{i^* Ne} \sum_{k=1}^{Ne} S_{ij}^k \forall i,j \tag{21}$$

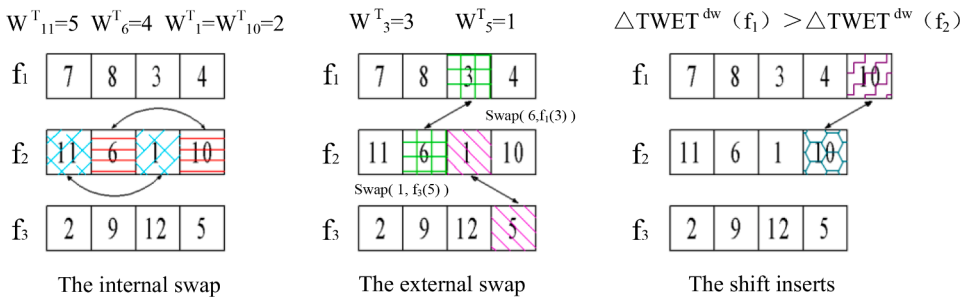


Fig. 5. Three neighborhood structures.

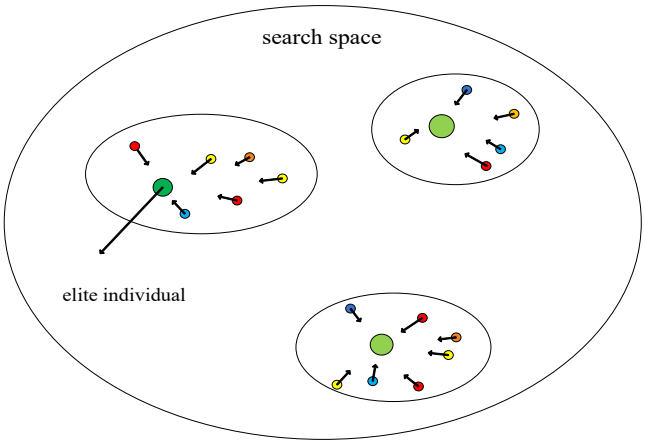


Fig. 6. The process of local search.

where $\theta \in (0,1)$ is the learning rate of historical information. S_{ij}^{k} is the enhancement factor in the *kth* elitist individuals solution in the superior population, which is defined as follows.

$$S_{ij}^{k} = \begin{cases} 1, if J_j isontimecomplement \\ 0, otherwise \end{cases} \tag{22}$$

Here is an example. The relevant data is defined as follows.

$$\begin{pmatrix} W_j^E \\ W_j^T \end{pmatrix} = \begin{pmatrix} 4 & 2 & 5 \\ 2 & 3 & 1 \end{pmatrix}$$

$$\begin{pmatrix} d_j^- \\ d_j^+ \end{pmatrix} = \begin{pmatrix} 2 & 9 & 5 \\ 8 & 13 & 11 \end{pmatrix}$$

The minimum of *TWET* exists in Fig. 4 (a) after calculation. The corresponding optimal sequence is $\{J_1, J_2, J_3\}$.

The probability model is initialized in the following steps.

Step 1: the Q table is initialized as follows.

$$\mathrm{Q} = \begin{pmatrix} 0.10 & 0.10 & 0.10 \\ 0.10 & 0.10 & 0.10 \\ 0.10 & 0.10 & 0.10 \end{pmatrix}$$

Step 2: the Q table is updated at the first generation according to Eq. (19).

$$i = 1 : q_{11} = q_{12} = q_{13} = 0.1$$

$$i > 1 : q_{21} = \frac{1.2 \times 0.1 \times 1 \times 2}{1 \times 6 \times 2 \times 0.1 + 1 \times 1 \times 3 \times 0.1 + 1 \times 7 \times 1 \times 0.1} = 0.11$$

$$q_{22} = \frac{1 \times 0.1 \times 1 \times 3}{1 \times 6 \times 2 \times 0.1 + 1 \times 1 \times 3 \times 0.1 + 1 \times 7 \times 1 \times 0.1} = 0.14$$

$$q_{23} = \frac{1 \times 0.1 \times 1 \times 3}{1 \times 6 \times 2 \times 0.1 + 1 \times 1 \times 3 \times 0.1 + 1 \times 7 \times 1 \times 0.1} = 0.14$$

$$q_{31} = \frac{1 \times 0.1 \times 6 \times 1}{1.2 \times 7 \times 2 \times 0.1 + 1.2 \times 1 \times 3 \times 0.1 + 1 \times 9 \times 1 \times 0.1} = 0.20$$

$$q_{32} = \frac{1 \times 0.1 \times 1 \times 3}{1.2 \times 7 \times 2 \times 0.1 + 1.2 \times 1 \times 3 \times 0.1 + 1 \times 9 \times 1 \times 0.1} = 0.10$$

$$q_{33} = \frac{1 \times 0.1 \times 9 \times 1}{1.2 \times 7 \times 2 \times 0.1 + 1.2 \times 1 \times 3 \times 0.1 + 1 \times 9 \times 1 \times 0.1} = 0.31$$

Step 3: The highest probabilities are saved. $q_{11} = 0.10$, $q_{22} = 0.14$, $q_{33} = 0.31$.

$$Q = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

The optimal sequence $\{J_1, J_2, J_3\}$ also is obtained.

4.5. The variable neighborhood olfactory search based on weight knowledge

Fruit fly individuals search around the population center to generate new individuals in the olfactory search stage, which is regarded as local search. Olfactory search is associated with the variable neighborhood search strategy. Three different neighborhood structures are proposed to improve the performance. The unit tardiness weight W^T is set as selection criteria to enhance the efficiency of local search according to the constraints of DNWFSPDW. The neighbor structures are formed through the three operations, which are the internal swap, external swap and shift insert Fig. 5. The optimization process is parallel across different factories, since the value of optimal solution is the sum of $TWET^{dw}$ values of all factories. The internal swap is the numerical optimization of a specific factory. The external swap is the simultaneous optimization of two factories. The shift insertion is utilized to insure the overall objective function optimization at the expense of objective function of another factory. In the olfactory search stage, jobs and factories are randomly selected and operated by the neighborhood methods to reduce computational complexity, since the scheduling problem is a large-scale complex problem with high dimension. The structures are described as follows.

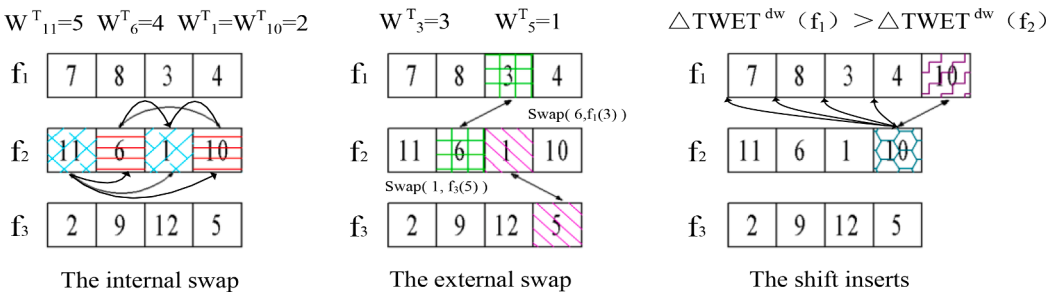


Fig. 7. The operation of local search.

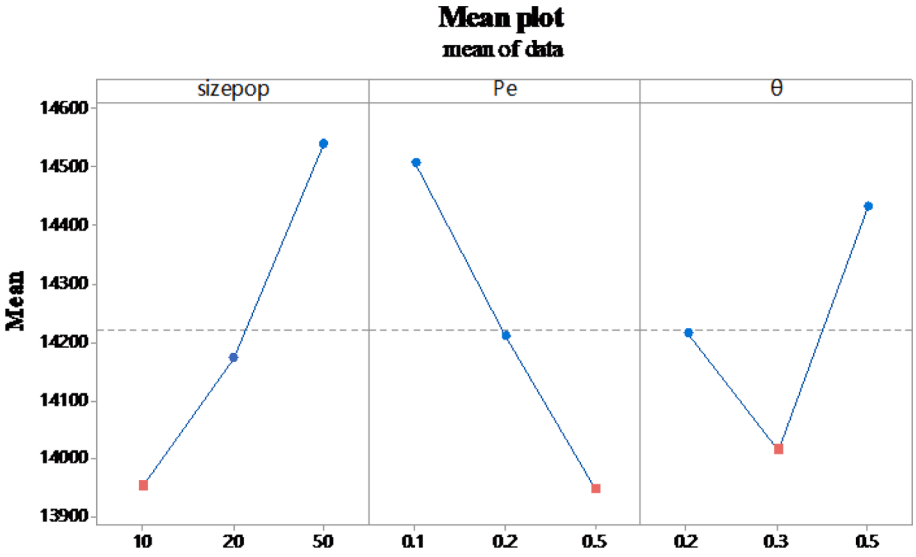


Fig. 8. The main effects plot.

1) S1(The internal swap): A factory is selected randomly and the jobs in the factory are sorted according to the unit tardiness weight W^T. The jobs with large W^T are selected to swap positions with others possessing small or equal W^T until the total $TWET^{dw}$ is improved.
2) S2(The external swap): Two factories are selected randomly, which is similar to the internal swap. A job is selected according to the sort of W^T from one factory and swapped with another job with smaller or equal W^T in another factory. The position with best fitness is kept.
3) S3(The shift insertion): A factory f_a is randomly selected and a job in the factory is extracted. The job is inserted into all positions in sequence of another factory f_b. When $\Delta TWET^{dw}(f_a) > \Delta TWET^{dw}(f_b)$, the operation is applied.

The inferior individuals learn around the nearby elite individuals, while they perform local search in the surrounding area of the elite individuals, as shown in Fig. 6. Thus, the olfactory search plays an

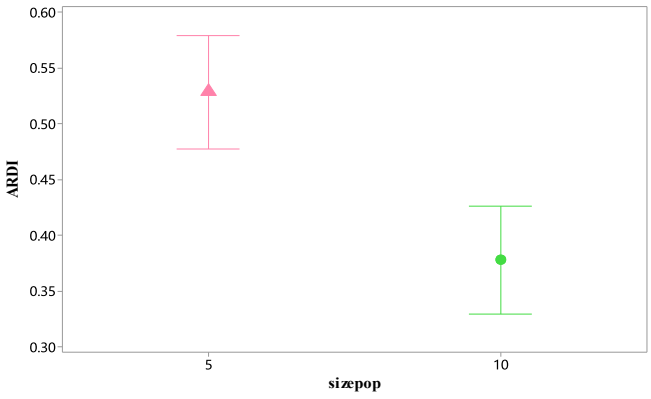


Fig. 9. Interval plot of parameter..*sizepop*

Table 1
ANOVA results for parameter settings of DKLFOA.

Source	Sum of squares	Degrees of freedom	Mean Square	*F −ratio*	*P −value*
sizepop	8.85E + 05	4	4.43E + 05	8.93	**0.04461**
Pe	9.68E + 05	5	9.68E + 05	9.81	**0.04167**
θ	4.61E + 05	2	2.31E + 05	4.7	0.06435
*sizepop*Pe*	2.87E + 06	10	7.17E + 05	5.6	0.3031
*sizepop*θ*	6.67E + 05	8	1.67E + 05	3.4	0.8457
*Pe*θ*	6.22E + 05	20	1.55E + 05	3.1	0.8609
Error	3.96E + 06	40	4.95E + 05		
Total	1.04E + 07	89			

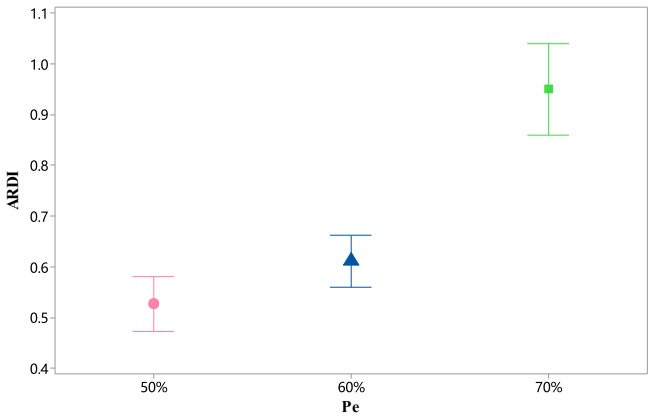


Fig. 10. Interval plot of parameter..*Pe*

important role in improving the accuracy of solutions in the proposed DKLFOA. An organized variable neighborhood descent (VND) is applied to search the potential region. The three different neighborhood structures mentioned above are guided for the search. Fruit fly individuals search from structure S1. If the performance of solution is improved, the local search of the current individuals is interrupted, and the next individual continues the local search from S1. If the current individual is not further improved, the search neighborhood is tuned to S2. If the solution is still not improved, the structure turns to S3. The new improved individuals replace the inferior ones and enter the next iteration. The details of search diagrams are shown in Fig. 7. The pseudocode is as follows.

Algorithm 3.

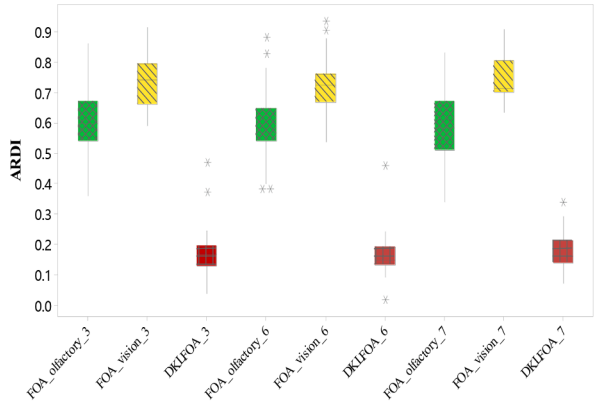
Algorithm 3. ***Olfactory Search***	
1	$S = 3, k = 1$
2	***for*** $j = 1$ ***to*** Ni
3	*Select different elitist individuals in the population center*
4	***While*** $k < S + 1$
5	*do local search in the structure k around the corresponding elitist individual e*
6	***If*** $twet_j < twet_e$
7	*The elitist individual e is replaced by the individual j*
9	***endif***
10	***endwhile***
11	***endfor***
12	***Output*** *the population and its corresponding twet*

Table 3
The comparison results of the proposed strategies.

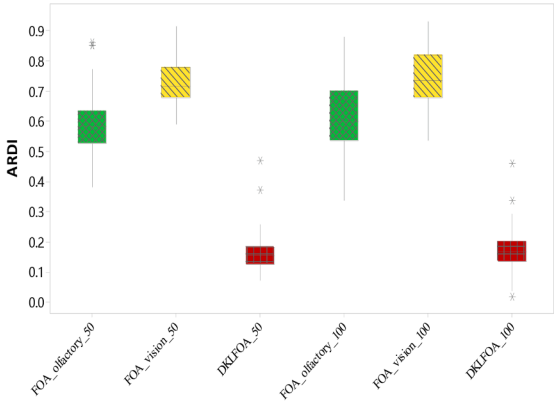
Category	Number	$FOA_{olfactory}$	FOA_{vision}	DKLFOA
Factory	3	0.614	0.740	**0.171**
	6	0.608	0.733	**0.163**
	7	0.597	0.754	**0.181**
Machine	10	0.598	0.735	**0.172**
	20	0.614	0.750	**0.171**
Jobs	50	0.592	0.733	**0.166**
	100	0.620	0.752	**0.177**
Mean		0.606	0.742	**0.172**

Table 2
ARDI of initialization mechanism.

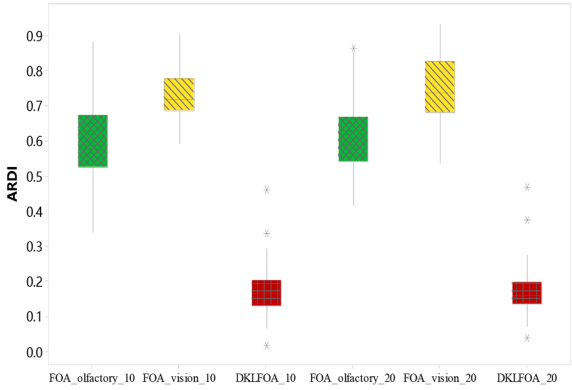
Factory	Jobs	$n \times m$	$KNEH^{dw}$	EDDWET	RANDOM	Factory	Jobs	$n \times m$	$KNEH^{dw}$	EDDWET	RANDOM
$F = 2$	20	20 × 5	**0.114**	0.271	0.800	$F = 3$	20	20 × 5	**0.058**	0.296	0.750
		20 × 10	**0.113**	0.225	0.673			20 × 10	**0.144**	0.218	0.765
		20 × 20	**0.161**	0.293	0.864			20 × 20	**0.155**	0.194	0.689
	50	50 × 5	**0.071**	0.370	0.762		50	50 × 5	**0.185**	0.247	0.750
		50 × 10	**0.102**	0.381	0.874			50 × 10	**0.124**	0.278	0.824
		50 × 20	**0.156**	0.375	0.768			50 × 20	**0.135**	0.350	0.784
	100	100 × 5	**0.069**	0.239	0.751		100	100 × 5	**0.050**	0.340	0.782
		100 × 10	**0.254**	0.106	0.811			100 × 10	**0.149**	0.208	0.643
		100 × 20	**0.224**	0.313	0.829			100 × 20	**0.255**	0.306	0.605
	200	200 × 10	**0.141**	0.160	0.842		200	200 × 10	**0.314**	0.339	0.749
		200 × 20	**0.161**	0.167	0.727			200 × 20	**0.145**	0.237	0.727
	500	500 × 20	**0.232**	0.326	0.831		500	500 × 20	**0.350**	0.414	0.784
		MEAN	**0.142**	0.261	0.794			MEAN	**0.180**	0.266	0.738
$F = 4$	20	20 × 5	**0.154**	0.212	0.813	$F = 5$	20	20 × 5	**0.122**	0.214	0.790
		20 × 10	**0.279**	0.350	0.718			20 × 10	**0.135**	0.255	0.735
		20 × 20	**0.130**	0.187	0.744			20 × 20	**0.124**	0.248	0.844
	50	50 × 5	**0.106**	0.145	0.698		50	50 × 5	**0.179**	0.192	0.799
		50 × 10	**0.138**	0.240	0.766			50 × 10	**0.056**	0.207	0.680
		50 × 20	**0.170**	0.273	0.792			50 × 20	**0.146**	0.269	0.803
	100	100 × 5	**0.049**	0.288	0.684		100	100 × 5	**0.115**	0.302	0.796
		100 × 10	**0.116**	0.212	0.944			100 × 10	**0.235**	0.291	0.768
		100 × 20	**0.157**	0.257	0.747			100 × 20	**0.059**	0.264	0.629
	200	200 × 10	**0.170**	0.227	0.809		200	200 × 10	**0.131**	0.265	0.781
		200 × 20	**0.220**	0.230	0.737			200 × 20	**0.206**	0.377	0.684
	500	500 × 20	**0.311**	0.421	0.692		500	500 × 20	**0.399**	0.403	0.740
		MEAN	**0.190**	0.245	0.762			MEAN	**0.159**	0.249	0.754
$F = 6$	20	20 × 5	**0.179**	0.218	0.580	$F = 7$	20	20 × 5	**0.262**	0.352	0.723
		20 × 10	**0.116**	0.138	0.750			20 × 10	**0.118**	0.165	0.704
		20 × 20	**0.125**	0.269	0.837			20 × 20	**0.132**	0.212	0.712
	50	50 × 5	**0.145**	0.214	0.660		50	50 × 5	**0.123**	0.172	0.753
		50 × 10	**0.526**	0.642	0.874			50 × 10	**0.466**	0.602	0.876
		50 × 20	**0.473**	0.587	0.751			50 × 20	**0.432**	0.633	0.772
	100	100 × 5	**0.128**	0.405	0.840		100	100 × 5	**0.047**	0.367	0.713
		100 × 10	**0.634**	0.684	0.735			100 × 10	**0.534**	0.694	0.714
		100 × 20	**0.547**	0.591	0.747			100 × 20	**0.557**	0.601	0.645
	200	200 × 10	**0.313**	0.424	0.721		200	200 × 10	**0.574**	0.580	0.773
		200 × 20	**0.403**	0.577	0.735			200 × 20	**0.431**	0.520	0.724
	500	500 × 20	**0.555**	0.634	0.767		500	500 × 20	**0.372**	0.550	0.686
		MEAN	**0.362**	0.440	0.750			MEAN	**0.346**	0.446	0.733



(a)The box-plot for different numbers of factories



(b) The box-plot for different numbers of jobs



(c) The box-plot for different numbers of machines

Fig. 11. The box-plot for DKLFOA,$FOA_{olfactory}$,.FOA_{vision}

4.6. The framework of DKLFOA

The concrete details of DKLFOA are described above. The whole flow steps are given in pseudocode as follows.

	The pseudo-code of DKLFOA
1	Input:the parameters of DKLFOA
2	Output: the value of *TWET* in best solution
3	**Begin**
4	/*Initiation stage*/
5	Initialize the parameters of the proposed algorithm
6	The structural solution is generated following the steps in Algorithm 1
7	The other solutions in the population are generated randomly.
8	/*Evolution stage*/
9	**while** the termination criterion is not satisfied
10	Divide the population into swarm center and inferior swarm according to fitness values *twet*
11	The inferior individuals perform the olfactory search in Algorithm 3
12	Update the swarm center
13	The vision search following the Algorithm 2 continues in the swarm center
14	**endwhile**
15	**Return the best value of** $TWET^{dw}$

5. The analysis and discussion of experimental results

DKLFOA is compared with the state-of-the-art algorithm in this section. The performance of DKLFOA is shown in the Taillard problem (Taillard, 1993). The test set is the common criterion in distributed shop scheduling problems. Six different distributed processing environments exist in the test set, which are the factory $f = \{2, 3, 4, 5, 6, 7\}$. In each processing environment, there are 120 different instances, which are

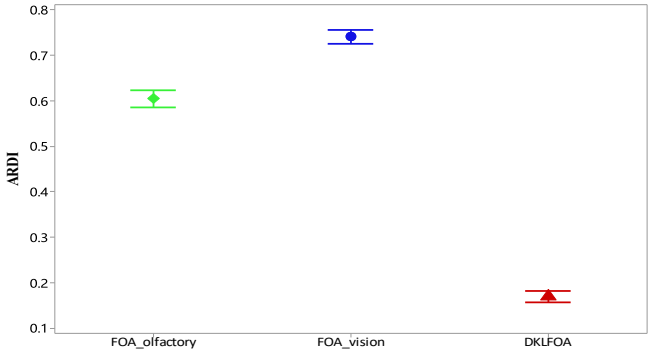


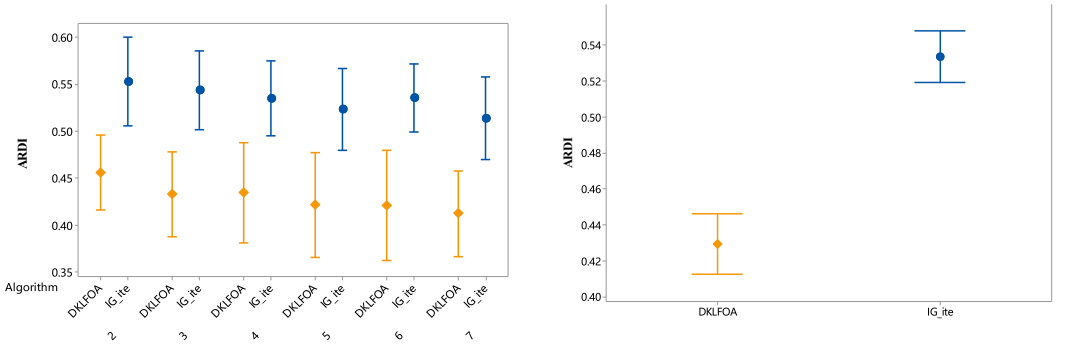
Fig. 12. The interval plot of interaction among DKLFOA,$FOA_{olfactory}$,.FOA_{vision}

divided into 12 different combinations of the number of machines and jobs. The combinations of the numbers are $\{20 \times 5, 20 \times 10, 20 \times 20, 50 \times 5, 50 \times 10, 50 \times 20, 100 \times 5, 100 \times 10, 100 \times 20, 200 \times 10, 200 \times 20, 500 \times 20\}$. 10 different instances are in each combination. Few algorithms concentrate on the optimization of $TWET^{dw}$ in the distributed flow shop problem. Only Iterated Greedy with Idle Time Insertion Evaluation (IG_{ITE}) (Jing, Pan, et al.2020) is selected as the compared algorithm. DOE is implemented to calibrate the key parameters of the proposed DKLFOA. The algorithm is coded by MATLAB and runs on a server with a 2.60 GHz Intel(R) Xeon(R) E5-2650CPU, 32 GB of RAM and 64-bit OS.

The criterion ARDI (Average Relative Deviation Index) is applied to measure the performance of the proposed DKLFOA and the compared algorithm. The value of ARDI ranges from 0 to 1. In the current solution,

Table 4
ARDI of the proposed algorithm and comparison algorithm.

Factory	Jobs	$n \times m$	DKLFOA	IG_{ITE}	Factory	Jobs	$n \times m$	DKLFOA	IG_{ITE}
$F = 2$	20	20 × 5	**0.350**	0.623	$F = 3$	20	20 × 5	**0.332**	0.599
		20 × 10	**0.392**	0.525			20 × 10	**0.371**	0.539
		20 × 20	**0.465**	0.472			20 × 20	**0.429**	0.502
	50	50 × 5	**0.420**	0.598		50	50 × 5	**0.443**	0.564
		50 × 10	**0.476**	0.563			50 × 10	**0.433**	0.503
		50 × 20	**0.498**	0.500			50 × 20	**0.496**	0.544
	100	100 × 5	**0.366**	0.711		100	100 × 5	**0.310**	0.718
		100 × 10	0.532	**0.426**			100 × 10	0.576	**0.458**
		100 × 20	**0.444**	0.566			100 × 20	**0.465**	0.558
	200	200 × 10	**0.513**	0.549		200	200 × 10	**0.460**	0.520
		200 × 20	0.540	**0.512**			200 × 20	**0.454**	0.500
	500	500 × 20	**0.473**	0.590		500	500 × 20	**0.426**	0.516
		MEAN	**0.456**	0.553			MEAN	**0.433**	0.543
$F = 4$	20	20 × 5	**0.272**	0.532	$F = 5$	20	20 × 5	**0.285**	0.506
		20 × 10	**0.320**	0.512			20 × 10	**0.332**	0.490
		20 × 20	**0.404**	0.452			20 × 20	**0.378**	0.422
	50	50 × 5	**0.443**	0.579		50	50 × 5	**0.453**	0.596
		50 × 10	**0.467**	0.519			50 × 10	**0.452**	0.545
		50 × 20	**0.503**	0.545			50 × 20	0.511	**0.510**
	100	100 × 5	**0.343**	0.665		100	100 × 5	**0.264**	0.596
		100 × 10	0.548	**0.416**			100 × 10	0.521	**0.392**
		100 × 20	**0.439**	0.570			100 × 20	**0.447**	0.596
	200	200 × 10	**0.473**	0.574		200	200 × 10	**0.426**	0.532
		200 × 20	**0.516**	0.523			200 × 20	0.516	**0.497**
	500	500 × 20	0.485	**0.484**		500	500 × 20	**0.468**	0.598
		MEAN	**0.434**	0.531			MEAN	**0.421**	0.523
$F = 6$	20	20 × 5	**0.269**	0.509	$F = 7$	20	20 × 5	**0.267**	0.495
		20 × 10	**0.348**	0.473			20 × 10	**0.326**	0.401
		20 × 20	**0.389**	0.430			20 × 20	**0.384**	0.398
	50	50 × 5	**0.443**	0.623		50	50 × 5	**0.412**	0.554
		50 × 10	**0.417**	0.560			50 × 10	**0.475**	0.531
		50 × 20	**0.434**	0.495			50 × 20	**0.479**	0.511
	100	100 × 5	**0.280**	0.622		100	100 × 5	**0.325**	0.635
		100 × 10	0.568	**0.516**			100 × 10	**0.444**	0.467
		100 × 20	**0.405**	0.524			100 × 20	**0.414**	0.575
	200	200 × 10	**0.459**	0.529		200	200 × 10	**0.463**	0.528
		200 × 20	**0.553**	0.577			200 × 20	**0.487**	0.496
	500	500 × 20	**0.481**	0.567		500	500 × 20	**0.466**	0.573
		MEAN	**0.421**	0.535		20	MEAN	**0.412**	0.514



(a) The Interval plot of compared algorithm with different numbers of factories

(b) The Interval plot of compared algorithm in overall performance

Fig. 13. The interval plot of all compared algorithms.

the smaller ARDI is, the closer the current best solution is to all the compared algorithm. On the contrary, the larger ARDI is, the closer the current solution is to the worst of all algorithms. If ARDI is 0, the current solution is the best. While ARDI is 1, it means that the current solution is the worst. ARDI is calculated as follows.

$$ARDI(TWET_i^{dw}) = \frac{1}{R}\sum_{i=1}^{R}\frac{(TWET_i^{dw} - TWET_{best}^{dw})}{(TWET_{worst}^{dw} - TWET_{best}^{dw})} \tag{23}$$

where R represents the running time, and $TWET_i^{dw}$ is the current objective function value of individual i, and $TWET_{best}^{dw}$ is the current best solution of all compared algorithms for the same instance so far. When $TWET_{worst}^{dw} = TWET_{best}^{dw}$, it means that all the algorithms obtain the same function values, then ARDI is set to 0.

5.1. The setting of due windows

A universally accepted method of due windows setting is lacked for DFSP. The method of due windows setting is proposed by (Jing et al., 2020) in this study. Since the common test, which is set with 720

Table 5
Rankings obtained through Wilcoxon's test for all the compared algorithms.

DKLFOA vs.	Factory	R+	R-	Z	*p-value*	*a = 0.05*
IG_{ITE}	2	4445.00	2815.00	−2.13E + 00	**3.28E-02**	**yes**
	3	4601.00	2659.00	−2.54E + 00	**1.10E-02**	**yes**
	4	4429.00	2831.00	−2.09E + 00	**3.64E-02**	**yes**
	5	4388.00	2872.00	−1.99E + 00	**4.70E-02**	**yes**
	6	4390.00	2870.00	−1.99E + 00	**4.66E-02**	**yes**
	7	4435.00	2825.00	−2.11E + 00	**3.50E-02**	**yes**

instances, is utilized to measure the performance of algorithm, Jing proposed a method to define the size of due windows based on the reduction and enlargement of the makespan among the instances. The unit weights of earliness and tardiness are determined before the experiment as follows. Each job has different weights.

$$w_j^E, w_j^T \in U[1,5] \tag{24}$$

The size of due windows is equal to the expansion and contraction of C_{max}. The mean of due date is determined as follow.

$$d_j = \max(0, U[C^*\left(1 - T - \frac{R}{2}\right), C^*(1 - T + \frac{R}{2})]) \tag{25}$$

where C is the *makespan*, which is calculated by NEH2 through the LPT rule. T is the late factor. R is the range of due date windows. The adopted comparison algorithm sets the values of T and R to 0.2. In order to keep consistent with the parameters and experimental environments of the comparison algorithm, this value is also adopted by this paper. If the values of T and R are too large, the earliness or tardiness is large. The effect for solving the problem is not obvious, resulting in cost waste. If T and R are too small, the control of earliness or tardiness time is not flexible enough, which imposes too much constraint on the supplier.

The threshold value of due windows in job j is determined as the Eqs. (26) - (27).

$$d_j^- = max(d_j * \left(1 + \frac{H}{100}\right), SUM(P_{j,i}) * \left(1 + \frac{H}{100}\right) \tag{26}$$

$$d_j^+ = max(d_j * \left(1 + \frac{H}{100}\right), SUM(P_{j,i}) * \left(1 + \frac{3*H}{100}\right) \tag{27}$$

where $H = U[1,10]$, $andSUM(P_{j,i})$ represents sum of processed time of job j on all machines.

5.2. Parameter analysis

Few parameters are used to control the performance of the original FOA. The advantage is retained in the improved DKLFOA. The search efficiency of DKLFOA is significantly improved without introducing extra parameters. Three factors affect experiment effect, including the size of population *sizepop*, the proportion of elite individuals in the population *Pe* and the probability of historical learning θ. Based on some experience from previous researches, the possible values of the three parameters are set as follows:$sizepop \in \{10, 20, 50\}$, $Pe \in \{10\%, 20\%, 50\%\}$, $\theta \in \{20\%, 30\%, 50\%\}$. There are $3 \times 3 \times 3 = 27$ different kinds of combinations of these parameters. Some instances are selected from the Taillard test set to verify the effects of different parameter combinations. Each instance with different combination runs 5 times respectively. The DOE method is applied to find the best combination of parameter values. As shown in Fig. 8, the main effect plot shows that $sizepop = 10, Pe = 50\%, \theta = 30\%$ are the most suitable combination.

The Analysis of Variance (ANOVA) experiment is employed to analyze the influence of each parameter on the experimental results. Three main hypotheses, including normality, homoscedasticity, and independence of residuals have been met after checking. The importance of the parameters and the interaction relationship between them are analyzed. The specific experimental results are shown in Table 1.

The $P-value$ of *sizepop* and *Pe* are less than 0.05 within the 95% confidence intervals from Table 1. The experiment proves that the proposed DKLFOA algorithm is sensitive to the two parameters, whose

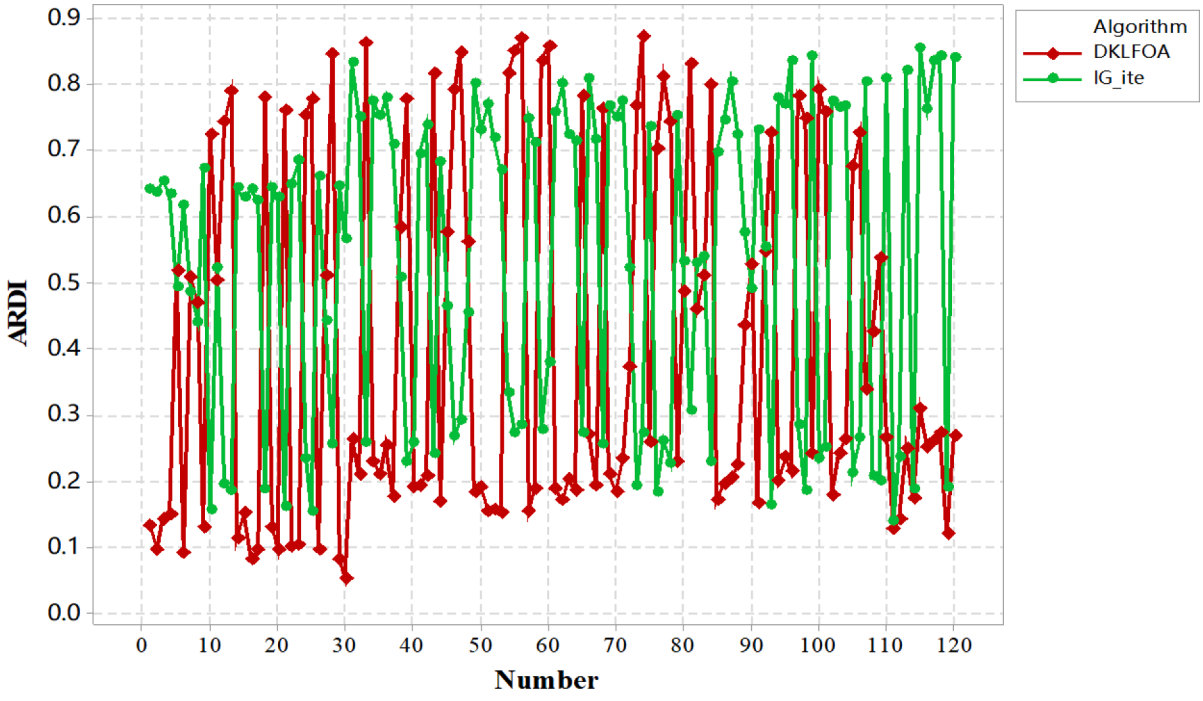


Fig. 14. The tendency chart of ARDI in 120 instances.

Table 6
ARDI of the proposed algorithm and comparison algorithm with $KNEH^{dw}$.

Factory	Jobs	n × m	DKLFOA	IG_{ITE}	Factory	Jobs	$n \times m$	DKLFOA	IG_{ITE}
$F = 2$	20	20 × 5	**0.222**	0.441	$F = 3$	20	20 × 5	**0.195**	0.447
		20 × 10	**0.243**	0.325			20 × 10	**0.259**	0.379
		20 × 20	**0.323**	0.377			20 × 20	**0.292**	0.351
	50	50 × 5	**0.243**	0.474		50	50 × 5	**0.315**	0.411
		50 × 10	**0.279**	0.467			50 × 10	**0.279**	0.391
		50 × 20	**0.328**	0.451			50 × 20	**0.318**	0.447
	100	100 × 5	**0.228**	0.475		100	100 × 5	**0.180**	0.531
		100 × 10	0.392	**0.261**			100 × 10	0.365	**0.333**
		100 × 20	**0.341**	0.442			100 × 20	**0.357**	0.436
	200	200 × 10	**0.328**	0.361		200	200 × 10	**0.387**	0.428
		200 × 20	0.353	**0.339**			200 × 20	**0.301**	0.369
	500	500 × 20	**0.352**	0.462		500	500 × 20	**0.388**	0.465
		MEAN	**0.231**	0.407			MEAN	**0.308**	0.405
$F = 4$	20	20 × 5	**0.213**	0.373	$F = 5$	20	20 × 5	**0.204**	0.362
		20 × 10	**0.300**	0.429			20 × 10	**0.234**	0.373
		20 × 20	**0.267**	0.320			20 × 20	**0.253**	0.337
	50	50 × 5	**0.275**	0.359		50	50 × 5	**0.317**	0.394
		50 × 10	**0.311**	0.380			50 × 10	**0.257**	0.377
		50 × 20	**0.337**	0.411			50 × 20	**0.329**	0.393
	100	100 × 5	**0.196**	0.477		100	100 × 5	**0.190**	0.449
		100 × 10	0.334	**0.314**			100 × 10	0.378	**0.342**
		100 × 20	**0.299**	0.414			100 × 20	**0.253**	0.433
	200	200 × 10	**0.322**	0.403		200	200 × 10	**0.279**	0.399
		200 × 20	**0.368**	0.378			200 × 20	**0.364**	0.437
	500	500 × 20	**0.398**	0.455		500	500 × 20	**0.434**	0.501
		MEAN	**0.313**	0.388			MEAN	**0.293**	0.386
$F = 6$	20	20 × 5	**0.224**	0.365	$F = 7$	20	20 × 5	**0.239**	0.424
		20 × 10	**0.233**	0.306			20 × 10	**0.222**	0.288
		20 × 20	**0.261**	0.351			20 × 20	**0.258**	0.309
	50	50 × 5	**0.294**	0.421		50	50 × 5	**0.269**	0.363
		50 × 10	**0.472**	0.601			50 × 10	**0.475**	0.567
		50 × 20	**0.454**	0.539			50 × 20	**0.456**	0.576
	100	100 × 5	**0.204**	0.514		100	100 × 5	**0.183**	0.506
		100 × 10	**0.559**	0.600			100 × 10	**0.489**	0.584
		100 × 20	**0.476**	0.558			100 × 20	**0.486**	0.587
	200	200 × 10	**0.387**	0.477		200	200 × 10	**0.519**	0.559
		200 × 20	**0.478**	0.579			200 × 20	**0.459**	0.511
	500	500 × 20	**0.518**	0.603		500	500 × 20	**0.419**	0.562
		MEAN	**0.393**	0.488		20	MEAN	**0.379**	0.478

changes have a significant impact on the search efficiency of DKLFOA. $F-ratio$ of Pe is the biggest among the three parameters, so Pe is the most important parameter. The $p-value$ of θ is bigger than 0.05 indicating that there is no significant influence on DKLFOA. $p-value$ of other parameter combinations in the results of ANOVA are also bigger than 0.05 from Table 1, which means that these parameters have no interactive relationship (Shao et al., 2018) (Shao et al., 2020). The main effect analysis is valid. Thus, $sizepop = 10, Pe = 50\%, \theta = 30\%$ are the optimal parameter combinations.

According to the experiment above, the parameters *sizepop* and *Pe* are artificially set as the bounds of threshold value. So, the scope of parameter setting values is expanded to increase the reliability of the experiment. The range of parameter values are redefined as follows: $sizepop \in \{5, 10\}$, $Pe = \{50\%, 60\%, 70\%\}$. The 6 combinations are considered further. The number of elite individuals is rounded off. When the number of elitists is greater than that of the inferior, the inferior does not repeatedly select the elite individuals from the whole elite as the objects of the local search. $sizepop = 10, Pe = 50\%$ are the optimal parameter combinations in Fig. 9 and Fig. 10. The proportion of elitist individuals *Pe* determines the number of learning objective of the inferior, which is the number of potential solutions for local search.

5.3. Performance analysis for each operation

The efficiency of each operation in DKLFOA is analyzed in this section. Three significant operations in DKLFOA include the initialization method $KNEH^{dw}$, the learning vision search and the olfactory local search based on VND, respectively. The proposed $KNEH^{dw}$ is combined with knowledge of the position between two fruit fly individuals in the initialization stage. The random initialization of original FOA is discretized to match the search method in DNWFSPDW. The initialization method of the rule based on the Earliest Due Date and the unit Earliness and Tardiness Weight (EDDWET) in IG_{ITE} (Jing et al., 2020) is applied to compare with the proposed $KNEH^{dw}$. These algorithms are run in 120 instances among different factory environments. The results of tests are the same every time, since both the $KNEH^{dw}$ and EDDWET are constructive initialization methods. So, each instance is run only one time for the two methods. The random initialization method is run 5 times and the average value is compared to avoid the contingency of experiment.

ARDI of the proposed initialization method is distinctively superior in the comparison algorithms in Table 2. In all different factory environments, the performance of $KNEH^{dw}$ is significantly better than that of other two initialization methods. The experimental results demonstrate that the quality of fruit fly initialization population is improved by the proposed $KNEH^{dw}$. The reason for the phenomenon is that $KNEH^{dw}$ is combined with the characteristics of due windows problems. The objective $TWET^{dw}$ of optimization is disassembled from the problem level, which effectively guides the search process. The specific knowledge of DNWFSP distance between jobs is used to control the increment of objective function effectively. To sum up, the high-quality center of fruit fly population is generated by the proposed $KNEH^{dw}$.

The other two critical operations in DKLFOA are the learning vision search and the knowledge-based olfactory search. The proposed learning visual search and knowledge-based olfactory are embedded into FOA framework respectively, which is called $FOA_{olfactory}$ and

FOA_{vision} to verify the effectiveness of the proposed strategies. An algorithm is defined as $FOA_{olfactory}$, in which the improved olfactory search is employed to local search around the single population center. The population center is replaced by the current optimal in the vision search, just like the original fruit fly algorithm. On the basis, the algorithm FOA_{vision} is embedded. A job is selected randomly and inserted into all possible positions of a randomly selected factory during the olfactory search stage. The algorithms are compared on a new test set to increase the reliability of the experiment, which are set up as follows, $n = \{50, 100\}$, $M = \{10, 20\}$, $F = \{3, 6, 7\}$. The processing time is randomly generated, $P_{j,i} \in U[1, 150]$. The relevant parameters of due windows are shown in Section 5.1. Every different combination of parameters contains 10 corresponding instances. The termination criterion is that $Terminal = n*m*fms$. The experimental results are shown in Table 3.

In Table 3, the mean value of ARDI in DKLFOA is smaller compared with other two algorithms. ARDI of DKLFOA is the smallest in the comparison algorithm, which means that the improved strategies are efficient for enhancing the performance of DKLFOA in solving DNWFSPDW problems. The main reasons are as follows: 1) Only one population center exists when only an olfactory search named $FOA_{olfactory}$ is introduced in the discrete FOA framework; FOA_{vision} introduces only multiple individuals to an elite group visual search strategy. 2) During the olfactory phase, knowledge flies randomly search for individuals in elite groups. 3) The single visual search of fruit fly population tends to fall into local optimum and is difficult to jump out, due to the lack of good cooperative mechanism. 4) Similarly, relying solely on olfactory search makes the population lack reasonable guidance in the process of regeneration, and blind olfactory search is equivalent to random search. Improving a single olfactory or visual search is not able to yield good performance. ARDI of $FOA_{olfactory}$ is usually smaller than that of FOA_{vision}, which means the results of olfactory search are closer to the best one in DKLFOA than the vision search. The olfactory search plays the critical role in DKLFOA.

ARDI of the two operations are much larger than that of the overall performance from Table 3. So, it is reasonable to infer that the cooperation effects of the two strategies are '$1 + 1 > 2$' rather than the simple overlay of effects.

The three algorithms are compared and analyzed by controlling different variables from the box-plots in Fig. 11, which is the box plot of the three algorithms compared in different numbers of factories. It is significant for the olfactory search to improve the performance of DKLFOA. The olfactory search is obviously the critical operation especially when the number of factories equals 7. The impact of the olfactory search and vision search of DKLFOA is outstanding in Fig. 11(b). The performance of the proposed DKLFOA is stable in Fig. 11(c), and the performance of the olfactory search and vision search is more unstable when the number of jobs is 100, which declares that the cooperation of the two operations effectively guarantees the stability of the algorithm. The interval plot of interaction within the 95% confidence interval is shown in Fig. 12. The performance of DKLFOA is availably enhanced by the collaboration of the olfactory and vision search. Both are effective to balance the exploration and exploitation in the search process. The combination of the previous figures and tables shows that the cooperation of olfactory and vision search strategies improves the performance of DKLFOA effectively.

5.4. The statistical analysis for all compared algorithms

Since the due windows are rarely considered in DFSP, only Jing introduces the due windows to the DPFSP for the first time. The objective function $TWET^{dw}$ is affected by many independent variables, so it is difficult to transplant the problem to other algorithms with different objectives. The state-of-the-art algorithm IG_{ITE} (Jing et al., 2020) is only selected as the comparison algorithm, in which the parameters are set according to the data in the original article. The terminal citation is set as $Terminal = n*m*f*30ms$. The proposed DKLFOA and the comparison algorithm IG_{ITE} are both performed 5 times independently. The specific results of these algorithms in the Taillard problem are shown in Table 4.

The ARDI mean values of the proposed DKLFOA in different factory environments outperform the comparison algorithm from Table 4. The proposed algorithm is slightly worse than the compared IG_{ITE} in some large-scale instances. DKLFOA is superior in overall performance for each factory. The performance of DKLFOA is better gradually with the increasement of the factory number. ARDI of DKLFOA is smaller than IG_{ITE} almost in every instance as demonstrated in Table 4.

Several reasons for the phenomenon are explained as follows. Firstly, the proposed algorithm retains the characteristics of the original fruit fly algorithm, which depends on the quality of the initial population. The proposed initialization method of $KNEH^{dw}$ is guided by the problem knowledge. The bigger the factory number is, the more time the jobs are processed on time. Thus, objective function decreases availably and the excellent overall performance of $KNEH^{dw}$ is obtained. The performance of DKLFOA is proportional to the quality of initialization population. Secondly, the improved vision search benefits from information accumulation and feedback. The problems are complex and require abundant information in the processing environment of the small quantity of factories and abundant jobs. The algorithm performance is weakened in the limited time of search. Finally, the smaller the factory number is, the more complex the neighbor structures are in single factory for the olfactory search.

There are significant differences between the DKLFOA and the compared algorithm except the $4-th$ factory in Fig. 13(a), which demonstrates that DKLFOA outperforms the compared algorithm in every different factory environment in the 95% confidence interval. DKLFOA is still under the level of IG_{ITE}, although the gap is not obvious for the $4-th$ factory. The obvious disparity is between DKLFOA and IG_{ITE} in Fig. 13(b), which illustrates that DKLFOA has better performance than the comparison algorithms. It is very suitable for solving the DNWFOADW problem.

All the results of these 720 instances are statistically analyzed by Wilcoxon's test to further scientifically verify the conclusions of the tables and figures above. The instances are divided into different groups based on the number of factories. Each group contains 120 instances, which are analyzed by group. The results of the experiments are exhibited in Table 5, where all p-values are less than 0.05, demonstrating that there are obvious differences in the proposed DKLFOA and comparison algorithm within the 95% confidence interval.

The level of DKLFOA is lower than that of IG_{ITE} in interval plot of Fig. 14. So, the performance of DKLFOA is prominently superior to the comparison algorithm in overall situation for solving the Taillard problem. The average ARDI of the 6 different numbers of factories is shown in Fig. 14. The red points, which are ARDI of DKLFOA, are obviously concentrated at a lower level compared with IG_{ITE}. In general, the proposed algorithm DKLFOA is effective for optimizing the DNWFSPDW problem, and achieves better performance compared with other specified algorithms.

$KNEH^{dw}$ is employed as the initialization stage of DKLFOA and IG_{ITE} respectively. As can be seen from the experimental results in Table 6, the overall effect of IG_{ITE} obtained by $KNEH$ is improved. IG_{ITE} with $KNEH^{dw}$ performs better than that with $EDDWET$ as its initialization method due to the guide of knowledge, but IG_{ITE} with $KNEH^{dw}$ does not perform as well as DKLFOA with $KNEH^{dw}$ because knowledge-based collaboration is utilized in DKLFOA.

6. Conclusion and future work

A discrete fruit fly algorithm based on knowledge is presented to solve the distributed no-wait flow shop scheduling problem with due windows in this paper. The proposed knowledge-based initialization method $KNEH^{dw}$ provides an initial solution with high quality for the

fruit fly population. The probability knowledge model of learning EDA contributes to improve the global search ability in vision search stage. The information of previous generations and the due windows are feedback as the probability information to guide the updating. Three different neighborhood structures attribute to the olfactory search stage, which improves the ability of the population to jump out of local optimum. All of these steps contribute significantly to the good performance of DKLFOA. The parameters of the proposed DKLFOA, calibrated through the DOE and ANOVA experiments, verify the availability of the proposed strategies in DKLFOA. The DKLFOA has obvious advantages in solving DNWFSPDW problem compared with the state-of-the-art algorithm. It also proves that the algorithm proposed in this paper is very suitable for solving DNWFSPDW. The knowledge-based methods and a combination with other heuristic algorithms are of certain reference significance for solving other scheduling problems.

Due window, which is a common problem in practical production, is a promising direction for future development in distributed flow shop scheduling. In this paper, the improved algorithm solves due windows of DNWFSP, but it does not take into account the total energy consumption, and green scheduling plays an important role in intelligent manufacturing. In the future, it is significant to find out a reinforcement learning due window setting method based on the investigation of actual production. Moreover, the due window problem will be considered in the application under the constraint of assembly and in industry problems under green indicators to meet the requirement of practical production.

CRediT authorship contribution statement

Ningning Zhu: Investigation, Software, Writing – original draft. **Fuqing Zhao:** Funding acquisition, Investigation, Supervision. **Ling Wang:** Methodology, Resources. **Ruiqing Ding:** Formal analysis, Writing – review & editing. **Tianpeng Xu:** Writing – review & editing. **Jonrinaldi:** Conceptualization, Visualization.

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.

Acknowledgments

This work was financially supported by the National Natural Science Foundation of China under grant 62063021. It was also supported by the Key talent project of Gansu Province (ZZ2021G50700016), the Key Research Programs of Science and Technology Commission Foundation of Gansu Province (21YF5WA086), Lanzhou Science Bureau Project (2018-rc-98), and Project of Gansu Natural Science Foundation (21JR7RA204), respectively.

References

Alidaee, B., Li, H., Wang, H., & Womer, K. (2021). Integer programming formulations in sequencing with total earliness and tardiness penalties, arbitrary due dates, and no idle time: A concise review and extension. *Omega (United Kingdom), 103*, Article 102446. https://doi.org/10.1016/j.omega.2021.102446

Behnamian, J., & Ghomi, S. (2016). A survey of multi-factory scheduling. *Journal of Intelligent Manufacturing, 27*(1), 231–249.

Cai, J., Lei, D., & Li, M. (2020). A shuffled frog-leaping algorithm with memeplex quality for bi-objective distributed scheduling in hybrid flow shop. *International Journal of Production Research, 2*, 1–18.

Chen, J. fang, Wang, L., & Peng, Z. ping. (2019). A collaborative optimization algorithm for energy-efficient multi-objective distributed no-idle flow-shop scheduling. *Swarm and Evolutionary Computation*, *50*(July), 100557. https://doi.org/10.1016/j.swevo.2019.100557.

Deng, J., & Wang, L. (2017). A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem. *Swarm and Evolutionary Computation*.

Eberhart, R. C. (2001). Swarm Intelligence. *Swarm*. Intelligence.

Fan, Y., Wang, P., Heidari, A. A., Wang, M., Zhao, X., Chen, H., & Li, C. (2020). Boosted hunting-based fruit fly optimization and advances in real-world problems. *Expert Systems with Applications, 159*, Article 113502. https://doi.org/10.1016/j.eswa.2020.113502

Fan, Y., Wang, P., Mafarja, M., Wang, M., Zhao, X., & Chen, H. (2021). A bioinformatic variant fruit fly optimizer for tackling optimization problems. *Knowledge-Based Systems, 213*, Article 106704. https://doi.org/10.1016/j.knosys.2020.106704

Fisher, M. L. (1981). The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Management Science, 27*(1), 1–18.

Fu, Y., Zhou, M. C., Guo, X., & Qi, L. (2021). Stochastic multi-objective integrated disassembly-reprocessing-reassembly scheduling via fruit fly optimization algorithm. *Journal of Cleaner Production, 278*, Article 123364. https://doi.org/10.1016/j.jclepro.2020.123364

Gao, K. Z., Pan, Q. K., & Li, J. Q. B. T.-S.-V. (2011). *Discrete harmony search algorithm for the no-wait flow shop scheduling problem with total flow time criterion*. 683–692.

Hall, N. G., & Sriskandarajah, C. (1996). A Survey of Machine Scheduling Problems with Blocking and No-Wait in Process. *Operations Research*, *44*(3), 510–525.

Han, Y., Gong, D., Li, J., & Zhang, Y. (2016). Solving the blocking flow shop scheduling problem with makespan using a modified fruit fly optimisation algorithm. *International Journal of Production Research*, 1–16.

Jing, X. L., Pan, Q. K., Gao, L., & Wang, Y. L. (2020). An effective Iterated Greedy algorithm for the distributed permutation flowshop scheduling with due windows. *Applied Soft Computing, 96*, Article 106629.

Khare, A., & Agrawal, S. (2019). Scheduling hybrid flowshop with sequence-dependent setup times and due windows to minimize total weighted earliness and tardiness. *Computers & Industrial Engineering, 135*(SEP), 780–792.

H. Li, X. Li, L. Gao (2020). A discrete artificial bee colony algorithm for the distributed heterogeneous no-wait flowshop scheduling problem - ScienceDirect. Applied Soft Computing.

Li, J. Q., Pan, Q. K., & Mao, K. (2016). A Hybrid Fruit Fly Optimization Algorithm for the Realistic Hybrid Flowshop Rescheduling Problem in Steelmaking Systems. *IEEE Transactions on Automation Science & Engineering*, 932–949.

Li, Y., Li, X., Gao, L., Zhang, B., & Meng, L. (2020). A discrete artificial bee colony algorithm for distributed hybrid flowshop scheduling problem with sequence-dependent setup times. *International Journal of Production Research, 10*, 1–20.

Lu, C., Gao, L., Yi, J., & Li, X. (2021). Energy-Efficient Scheduling of Distributed Flow Shop with Heterogeneous Factories: A Real-World Case from Automobile Industry in China. *IEEE Transactions on Industrial Informatics, 17*(10), 6687–6696. https://doi.org/10.1109/TII.2020.3043734

Mao, J. Y., Pan, Q. K., Miao, Z. H., & Gao, L. (2021). An effective multi-start iterated greedy algorithm to minimize makespan for the distributed permutation flowshop scheduling problem with preventive maintenance. Expert Systems with Applications, 169(December 2020), 114495. https://doi.org/10.1016/j.eswa.2020.114495.

Naderi, B., & Ruiz, R. (2010). The distributed permutation flowshop scheduling problem. *Computers & Operations Research, 37*(4), 754–768.

Pan, Q. K., Ruiz, R., & Alfaro-Fernandez, P. (2017). Iterated search methods for earliness and tardiness minimization in hybrid flowshops with due windows. *Computers & Operations Research, 80*(APR.), 50–60.

Pan, Q. K., Tasgetiren, M. F., Suganthan, P. N., & Chua, T. J. (2011). A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. *Information Sciences, 181*(12), 2455–2468.

Pan, Q. K., Gao, L., Wang, L., Liang, J., & Li, X. Y. (2019). Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem. *Expert Systems with Applications, 124*(JUN.), 309–324.

Pan, W. T. (2012). A new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example. *Knowledge-Based Systems, 26*(2), 69–74.

Qian, B., Wang, L., Huang, D. X., & Wang, X. (2009). An effective hybrid PSO-based algorithm for flow shop scheduling with limited buffers. *Computers & Operations Research, 33*(1), 2960–2971.

Qin, S., Pi, D., Shao, Z., & Xu, Y. (2022). Hybrid collaborative multi-objective fruit fly optimization algorithm for scheduling workflow in cloud environment. *Swarm and Evolutionary Computation*, *68*(October 2021), 101008. https://doi.org/10.1016/j.swevo.2021.101008.

Rossi, F. L., & Nagano, M. S. (2020). Heuristics and metaheuristics for the mixed no-idle flowshop with sequence-dependent setup times and total tardiness minimisation. *Swarm and Evolutionary Computation, 55*(March), Article 100689. https://doi.org/10.1016/j.swevo.2020.100689

Shao, W., Pi, D., & Shao, Z. (2019). A Pareto-Based Estimation of Distribution Algorithm for Solving Multiobjective Distributed No-Wait Flow-Shop Scheduling Problem With Sequence-Dependent Setup Time. *IEEE Transactions on Automation Science and Engineering*, 1–17.

Shao, Z. S., Pi, D., & Shao, W. (2018). A novel discrete water wave optimization algorithm for blocking flow-shop scheduling problem with sequence-dependent setup times. *Swarm and Evolutionary Computation*, *40*(December 2017), 53–75. https://doi.org/10.1016/j.swevo.2017.12.005.

Shao, Z. S., Pi, D., & Shao, W. (2020). Hybrid enhanced discrete fruit fly optimization algorithm for scheduling blocking flow-shop in distributed environment. *Expert Systems with Applications, 145*, Article 113147. https://doi.org/10.1016/j.eswa.2019.113147

Springer. (2016). *Scheduling theory, algorithms, and systems fifth edition*. Operations Research Proceedings 1991.

Taillard, E. D. (1993). Benchmarks for Basic Scheduling Problems. *European Journal of Operational Research, 64*(2), 278–285.

Tasgetiren, M. F., Ztop, H., Gao, L., Pan, Q. K., & Li, X. (2019). A Variable Iterated Local Search Algorithm for Energy-Efficient No-idle Flowshop Scheduling Problem. *Procedia Manufacturing, 39*, 1185–1193.

Toptal, A., & Sabuncuoglu, I. (2010). Distributed scheduling: A review of concepts and applications. *International Journal of Production Research, 48*(18), 5235–5262. https://doi.org/10.1080/00207540903121065

Wang, J. J., & Wang, L. (2018). A Knowledge-Based Cooperative Algorithm for Energy-Efficient Scheduling of Distributed Flow-Shop. *IEEE Transactions on Systems Man & Cybernetics Systems*, 1–15.

Wang, L., & Shen, W. (2007). *Process Planning and Scheduling for Distributed Manufacturing.* London: Springer.

Wang, L., Zheng, X. L., & Wang, S. Y. (2013). A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem. *Knowledge-Based Systems, 48*(aug.), 17–23.

Wang, L., Pan, Q. K., & Fatih Tasgetiren, M. (2010). Minimizing the total flow time in a flow shop with blocking by using hybrid harmony search algorithms. *Expert Systems with Applications, 37*(12), 7929–7936. https://doi.org/10.1016/j.eswa.2010.04.042.

Ying, K. C., Lin, & Shih-Wei. (2018). Minimizing makespan for the distributed hybrid flowshop scheduling problem with multiprocessor tasks. *Expert Systems with Application.*

Liang Y. S., Ren Z. G., Yao X. H., Feng Z. R., An, & Chen. (2018). Enhancing Gaussian Estimation of Distribution Algorithm by Exploiting Evolution Direction With Archive. *IEEE Transactions on Cybernetics.*

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

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

Zhang, W., Wang, Y., Yang, Y., & Gen, M. (2019). Hybrid Multiobjective Evolutionary Algorithm based on Differential Evolution for Flow Shop Scheduling Problems. *Computers & Industrial Engineering, 130*(APR.), 661–670.

Zhao, F. Q., He, X., Zhang, Y., Lei, W., & Song, H. (2019). A jigsaw puzzle inspired algorithm for solving large-scale no-wait flow shop scheduling problems. *Applied Intelligence, 3.*

Zhao, F. Q., Ma, R., & Wang, L. (2021). A Self-Learning Discrete Jaya Algorithm for Multiobjective Energy-Efficient Distributed No-Idle Flow-Shop Scheduling Problem in Heterogeneous Factory System. *IEEE Transactions on Cybernetics, 1–12.* https://doi.org/10.1109/TCYB.2021.3086181

Zhao, F. Q., Qin, S., Zhang, Y., Ma, W., Zhang, C., & Song, H. (2019). A Hybrid Biogeography-Based Optimization with Variable Neighborhood Search Mechanism for No-wait Flow Shop Scheduling Problem. *Expert Systems with Applications, 126*(JUL.), 321–339.

Zhao, F. Q., Zhang, L., Cao, J., & Tang, J. (2020). A Cooperative Water Wave Optimization Algorithm with Reinforcement Learning for the Distributed Assembly No-idle Flowshop Scheduling Problem. *Computers & Industrial Engineering, 153*(10), Article 107082.

Zhao, F. Q., Zhang, L., Zhang, Y., Ma, W., Zhang, C., & Song, H. (2020). A hybrid discrete water wave optimization algorithm for the no-idle flowshop scheduling problem with total tardiness criterion. *Expert Systems with Application, 146*(May), 113166.1, 113166.21.

Zhao, F. Q., Ding, R. Q., Wang, L., Cao, J., & Tang, J. (2021). A hierarchical guidance strategy assisted fruit fly optimization algorithm with cooperative learning mechanism. *Expert Systems with Applications, 183*(June), Article 115342. https://doi.org/10.1016/j.eswa.2021.115342

Zhao, F. Q., Zhao, J., Wang, L., & Tang, J. (2021). An optimal block knowledge driven backtracking search algorithm for distributed assembly No-wait flow shop scheduling problem. *Applied Soft Computing, 112*, Article 107750. https://doi.org/10.1016/j.asoc.2021.107750.

Zheng, D. Z., & Wang, L. (2003). An Effective Hybrid Heuristic for Flow Shop Scheduling. *International Journal of Advanced Manufacturing Technology, 21*(1), 38–44.

Zheng, X. L., & Wang, L. (2016a). A Collaborative Multiobjective Fruit Fly Optimization Algorithm for the Resource Constrained Unrelated Parallel Machine Green Scheduling Problem. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, PP (99)*, 1–11.

Zheng, X. L., & Wang, L. (2016b). A two-stage adaptive fruit fly optimization algorithm for unrelated parallel machine scheduling problem with additional resource constraints. *Expert Systems with Applications, 65*(dec.), 28–39.

Zheng, X. L., & Wang, L. (2018). A knowledge-guided fruit fly optimization algorithm for dual resource constrained flexible job-shop scheduling problem. *International Journal of Production Research, 54*(17–18), 1–13.

Zohali, H., Naderi, B., & Mohammadi, M. (2019). The economic lot scheduling problem in limited-buffer flexible flow shops: Mathematical models and a discrete fruit fly algorithm. *Applied Soft Computing, 80*, 904–919. https://doi.org/10.1016/j.asoc.2019.03.054.