Knowledge-Based Systems 243 (2022) 108471

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

An effective water wave optimization algorithm with problem-specific knowledge for the distributed assembly blocking flow-shop scheduling problem

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

[a] School of Computer and Communication, 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

Article history:
Received 19 November 2021
Received in revised form 13 February 2022
Accepted 16 February 2022
Available online 23 February 2022

Keywords:
Distributed assembly
Flow-shop scheduling problem
Total tardiness
Water wave optimization
Problem-specific knowledge

ABSTRACT

The distributed assembly blocking flow-shop scheduling problem (DABFSP), which is a promising area in modern supply chains and manufacturing systems, has attracted great attention from researchers and practitioners. However, minimizing the total tardiness in DABFSP has not captured much attention so far. For solving the DABFSP with the total tardiness criterion, a mixed integer linear programming method is utilized to model the problem, wherein the total tardiness during the production process and assembly process are optimized simultaneously. A constructive heuristic (KBNEH) and a water wave optimization algorithm with problem-specific knowledge (KWWO) are presented. KBNEH is designed by combining a new dispatching rule with an insertion-based improvement procedure to obtain solutions with high quality. In KWWO, effective technologies, such as the re-developed destruction–construction operator, four local search methods under the framework of the variable neighborhood search strategy (VNS), the path-relinking method are applied to improve the performance of the algorithm. Comprehensive numerical experiments based on 900 small-scale benchmark instances and 810 large-scale benchmark instances are conducted to evaluate the performance of the presented algorithm. The experimental results obtained by KWWO are 1 to 4 times better than those obtained by the other comparison algorithms, which demonstrate that the effectiveness of KWWO is superior to the compared state-of-the-art algorithms for the considered problem.

1. Introduction

The traditional blocking flow-shop scheduling problem (BFSP) follows a mono-factory production environment where all manufacturing processes take place in the same factory [1]. Under the globalization environment, manufacturing enterprises strive to achieve competitive manufacturing systems to confront the new requirements such as shorter product delivery time and more product categories for manufacturing systems of the market [2]. In recent years, production has shifted from the traditional mono-factory production environment to the multi-factory production pattern for reducing the risk and costs by increasing the production reliability, flexibility, and responsiveness levels [3].

Multi-factory scheduling is referred to as distributed scheduling or parallel scheduling [4]. Given its reality, Ribas et al. [5] proposed the distributed blocking flow-shop scheduling problem (DBFSP), which is extended by the classical BFSP. In DBFSP, n jobs have to be assigned in one of F ($F \geq 2$) identical factories, each of which consists of m machines required to be visited by all jobs in the same order. Therefore, the problem is solved based on two simultaneous decisions: the assignment of jobs to different factories and the sequence of the allotted jobs for each factory [6]. There is no buffer between machines. Hence, it is not allowed for the processing jobs to wait in the production system for the next operations. The job has to be blocked on the current machine until the next machine is available for processing. The blocking constraint widely exists in many manufacturing systems such as iron and steel [7], and waste treatment [8]. The assembly stage contains a single assembly machine for assembling the processed jobs into the products [9].

In the past decades, the assembly flow-shop scheduling problem (AFSP) as one of the most general scheduling problems, has been widely used for its practical interest [10]. In an assembly system, the final product is made of a given number of different assembly parts. The assembly procedure represents the relationship between different parts. The distributed assembly

* Corresponding author.
E-mail addresses: Fzhao2000@hotmail.com (F. Zhao), sdongqu@163.com (D. Shao), wangling@tsinghua.edu.cn (L. Wang), x.utp@lut.edu.cn (T. Xu), 307516638@qq.com (N. Zhu), j.onrinaldi@eng.unand.ac.id (Jonrinaldi).

https://doi.org/10.1016/j.knosys.2022.108471
0950-7051/

permutation flow-shop scheduling problem (DAPFSP) proposed by Hatami et al. [11] is the first report about the combination of flow-shop scheduling and assembly scheduling. Although the scheduling problems in the distributed assembly systems have been widely studied following Hatami [12–14], the literature on the distributed assembly blocking flow-shop scheduling problem (DABFSP) is very limited.

In industrial production configuration, total tardiness is a realistic index that related to customer satisfaction in scheduling jobs. If the enterprises are unable to meet the due date of the customer, it will lead to the penalty clause in the contract, loss of reputation, and damage to reputation [15]. Furthermore, total tardiness is different from the makespan, which is also a very important scheduling index. The two criteria have been proved to be in conflict in the production scheduling [16,17]. The first attempt to solve the flow-shop scheduling problem with the objective of minimizing total tardiness was by Gelders et al. in 1978s [18]. Then, the objective is quite attractive in various production scheduling environments [19,20]. The total tardiness criterion helps to allocate and queue jobs to meet their deadlines, which is essential for companies to survive in the current competitive market. Therefore, minimizing the total tardiness time is considered to be the most important scheduling objective in the current competitive environment [21]. It is worth mentioning that the existing researches about the DABFSP with total tardiness criterion are relatively scarce.

This paper considers the DABFSP with the objective of minimizing the total tardiness time, which is denoted as $DF|blocking|\sum TT_j$ [22]. DABFSP is a generalization of DBFSP with an assembly stage. A layout of DABFSP is shown in Fig. 1. The manufacturing process in DABFSP has two stages: distributed flow-shop production with blocking constraint and assembly in a single factory [23]. Especially, the assembly stage contains a single assembly machine for assembling the processed jobs into the products [24]. Thus, there are three decisions in DABFSP, including the assignment of jobs to different factories, the processing sequence of jobs in each factory, and the assembly sequence for each product. The DBFSP has been proven to be an NP-hard problem when the number of machines is more than two [25]. Thus, as the extension of DBFSP, it is easily concluded that the DABFSP is NP-hard well.

To address such an important scheduling problem, an effective water wave optimization algorithm with problem-specific knowledge (KWWO) is presented in this paper. The primary contributions of KWWO are summarized as follows.

- The distributed assembly blocking flow-shop scheduling problem with the total tardiness criterion is investigated. This problem is a generalization of blocking processing production with an assembly stage.
- Several properties are derived according to the problem characteristics of DABFSP. A new constructive heuristic (KB-NEH), which modifies the NEH algorithm by introducing a new assignment rule of jobs, is proposed to generate promising initial solutions with high quality.
- The neighborhood search methods of KWWO are newly designed to obtain the plummy equilibration between exploration and exploitation by introducing the problem-specific knowledge during the algorithmic search process.

The rest of the paper is structured as follows. Section 2 offers the related work on the distributed flow-shop scheduling problem and the approaches; The motivation of this paper is also provided. Section 3 presents a formal description of the DABFSP and the optimization objective. Section 4 is devoted to the description of the proposed constructive heuristic. Section 5 describes the proposed algorithm in detail. Section 6 provides a series of computational experiments and analyses. Finally, Section 7 concludes with the paper and presents some future work.

2. Related work

2.1. The related work on the blocking flow-shop scheduling problem

Production scheduling, which reasonably allocates resources in various types of the production environment, has received considerable attention and has been broadly studied from industrial and academic. As a typical scheduling problem, the BFSP has been investigated by researchers over the past decades since Wang et al. designed a novel hybrid discrete differential evolution algorithm (HDDE) for solving the BFSP with the makespan criterion [26]. The representative work about the blocking flow-shop scheduling problem in recent years is listed in Table 1. Following this work, many high-performing methods have been developed to tackle it, e.g., discrete artificial bee colony algorithm incorporating differential evolution (DE-ABC) [27], Estimation of distribution algorithm (P-EDA) [28], discrete invasive weed optimization (DIWO) [29], self-adaptive discrete invasive weed optimization (SaDIWO) [30], multi-objective discrete water wave optimization (MODWWO) algorithm [31], etc. Due to the increasing production demand and the market dispersion throughout the world, researchers have focused on scheduling in the distributed production environment, where the first work is that Ribas et al. extended the classical blocking flow-shop scheduling problem (BFSP) to the multi-factory environment to propose a distributed BFSP (DBFSP) [5]. The authors presented a mathematical model for the DBFSP, and some constructive heuristics along with the iterated local search (ILS) and the iterated greedy algorithm (IGA) were proposed to tackle the problem with the makespan criterion. For the same problem, Zhang et al. proposed a novel hybrid discrete differential evolution (DDE) algorithm [32]. Shao et al. established a hybrid enhanced discrete fruit fly optimization algorithm (HEDFOA), where an effective constructive heuristic was presented in the initial phase, and several neighborhood operators were utilized to improve the algorithmic search capacity [25]. Zhao et al. proposed an ensemble discrete differential evolution (EDE) algorithm [6] and Chen et al. designed six constructive heuristics and an iterated greedy (IG) algorithm to solve the DBFSP with the objective of minimizing the makespan [33]. Additionally, researchers also focused on solving the DBFSP with other objectives. For example, several constructive procedures and an iterated greedy algorithm were presented by Ribas et al. For solving the total tardiness in DBFSP [34]. Recently, Ribas et al. considered the sequence-dependent setup times in DABFSP and proposed an iterated greedy algorithm (IGA) to optimize the makespan criterion [35]. Furthermore, the distributed production scheduling problems in the blocking environment with other types of constraints have been also investigated. Shao et al. considered a distributed mixed permutation BFSP (DMBFSP), where three types of blocking constraint were included. To solve the considered problem, an improved NEH heuristic (NEH_P) with an efficient iterated greedy (EIG) algorithm was introduced [9]. Although the BFSP and DBFSP have been widely studied over the past years, the literature on the DABFSP is very limited. As shown in Table 1, the distributed assembly BFSP is only investigated by Shao et al. [36] very recently. Therefore, the DABFSP is quite a new issue. Considering the importance in today's manufacturing environment and infrequent research of DABFSP, it is worthwhile to develop more effective scheduling approaches to solve the DBFSP better.

2.2. The related work on water wave optimization algorithm

The literature on utilizing the metaheuristics algorithm to address complex optimization problems had grown significantly during the past decades [37]. Water Wave Optimization algorithm

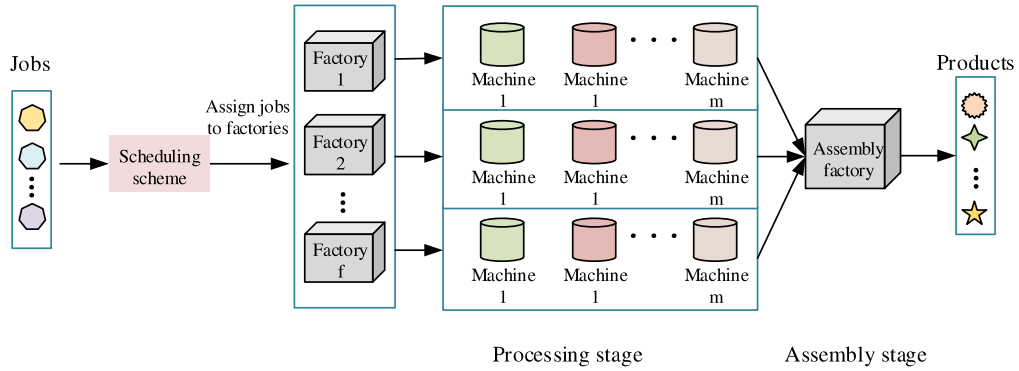


Fig. 1. A layout for the DABFSP.

Table 1
Review of representative work for the blocking flow-shop scheduling problem.

Problems	Objectives	Methods
BFSP	C_{max}	Hybrid discrete differential evolution algorithm (HDDE) [26]. Discrete artificial bee colony algorithm incorporating differential evolution (DE-ABC) [27]. Estimation of distribution algorithm (P-EDA) [28] Discrete invasive weed optimization (DIWO) [29]
	$\sum TTD_j$	Self-adaptive discrete invasive weed optimization (SaDIWO) [30].
	C_{max}, $\sum C_j$	Multi-objective discrete water wave optimization (MODWWO) algorithm [31].
DBFSP	C_{max}	Some constructive heuristics, iterated local search (ILS), and iterated greedy algorithm (IGA) [5]. Hybrid discrete differential evolution (DDE) algorithm [32]. Hybrid enhanced discrete fruit fly optimization algorithm (HEDFOA) [25]. An ensemble discrete differential evolution (EDE) algorithm [6]. Six constructive heuristics and an iterated greedy (IG) algorithm [33].
	$\sum TTD_j$	Several constructive procedures and an iterated greedy algorithm [34].
DBFSP-SDST	C_{max}	An iterated greedy algorithm (IGA) [35].
DMBPFSP	C_{max}	An improved NEH heuristic (NEH_P) and an efficient iterated greedy (EIG) algorithm [9].
DABFSP	C_{max}	A constructive heuristic and an iterated local search (ILS) [36].

Note: $\sum C_j$ total flow time $\sum TTD_j$ total tardiness.

(WWO) is a promising algorithm, which is inspired by the shallow wave theory [38]. WWO was first introduced to continuous optimization problems. There are three fundamental operations in the basic WWO, including propagation, breaking, and refraction operations. The equilibration between exploration and exploitation during iterations is achieved by the simple and effective mechanism of WWO. Due to its robustness and effectiveness, WWO has been applied to optimize the workflow scheduling in the cloud [39], Autonomous underwater vehicle (AUV) [40], machine utilization [41], and flow-shop scheduling problem [14].

In the previous work, Yun et al. [42] redesigned the basic operations of WWO based on the framework of the memetic algorithm to solve the combined arrangement flow-shop scheduling problem. A discrete WWO algorithm (DWWO) was proposed by Zhao et al. [43] to address the no-wait flow-shop scheduling problem based on the maximum completion time criterion. Zhao et al. [44] proposed a hybrid discrete WWO algorithm (HWWO) for the no-idle flow-shop scheduling problem with total tardiness criterion. Additionally, Zhao et al. [14] proposed a cooperative WWO algorithm (CWWO) for the distributed assembly no-idle flow-shop scheduling problem (DSNIFSP). Shao et al. [45] proposed the discrete WWO algorithm (DWWO) for the blocking flow-shop scheduling problem with sequence-dependent setup times (BFSP-SDST). Then, the single wave mechanism-the single WWO algorithm (SWWO) was presented by Shao et al. [46] for the no-wait block flow-shop scheduling problem (NWFSP) with the goal of the minimum completion time. Later, to minimize both makespan and total flow time in the blocking flow-shop scheduling problem, the novel multi-objective WWO algorithm (MOD-WWO) was introduced by Shao et al. [31].

2.3. The motivations of this paper

From the review in the preceding sections, several points are wondrously worth paying attention to.

- The DFSP and its different variants have received considerable attention and have been broadly investigated in recent years. Whereas, more realistic constraint needs to be considered to bring the problem suitable for the manufacturing systems in the real-word as close as possible. The DABFSP, which is one of the extensions of DAPFSP with the blocking constraint, is seldom studied before in the scheduling literature. The literature on the DABFSP is limited.
- Most of the previous work for the DBFSP acknowledged the makespan as the optimization function, and some other objective functions were ignored such as minimizing the total tardiness. The flow-shop scheduling problem with the makespan criterion and the flow-shop scheduling problem with the total tardiness criterion is quite different in the mathematical sense. In the customer-orientated market, minimization of the total tardiness ensures the products are delivered on time under the due date demands. However, the essential and relevant objective is studied scarcely. Therefore, it is worthwhile to optimize the total tardiness in the scheduling problem.
- On the one hand, WWO has been proven to be an effective algorithm for solving multitudinous problems since it was proposed. Several operators are easily adopted to modify the algorithm given the distinctive structure of WWO. On the other hand, there is no reported work on WWO for solving the DABFSP, the goal of this paper is to propose an effective

WWO to solve the DABFSP with the objective of minimizing the total tardiness.

Motivated by the above points, the distributed assembly blocking flow-shop scheduling problem to minimize the total tardiness criterion is presented and a knowledge-based self-adaptive water wave optimization algorithm (KWWO) is developed to solve the considered problem.

3. Distributed assembly blocking flow-shop scheduling problem

In this section, the considered problem $DF\,|blocking|\sum TTD_j$ is explained through the mixed integer linear programming (MILP) and a numerical illustration. Some necessary notations are given as follows to simplify the expression of the formula.

Indices

j	Index of the current jobs; $j \in \{1, 2, \ldots, n\}$
i	Index of the current machines; $i \in \{1, 2, \ldots, m\}$
k	Index of the current job position; $k \in \{1, 2, \ldots, n\}$
f	Index of the current factories; $f \in \{1, 2, \ldots, F\}$
h	Index of the current products; $h \in \{1, 2, \ldots, s\}$

Parameters

n	The total number of jobs
m	The total number of machines
π	The scheduling sequence, where $\pi = [\pi^1, \pi^2, \ldots, \pi^F]$
n^f	The number of jobs in factory f
$\{\pi_1^f, \pi_2^f, \ldots, \pi_n^f\}$	The processing sequence of jobs in factory f
F	The total number of factories
s	The total number of products
$P_{j,i}$	The processing time of job j on machine i
N_h	The number of jobs composing product h
T_h	The assembly time of product h
LJ_h	The last processed job that belongs to product h
DA_h	The assembly completion time of the product h
As	The assembly sequence of product, where $AS = \{As(1), As(2), \ldots, As(S)\}$
Due_j	The due date of job j

Decision Variable

$G_{j,h}$	1 If job j belongs to the product h, 0 otherwise
$x_{j,k,f}$	1 if job j is assigned to position k in factory f; 0 otherwise
$D_{k,i,f}$	The departure time of job at position k on machine i

3.1. Mixed integer linear programming

The scheduling process in DABFSP is divided into two stages: production and assembly. To make the problem definition visual and clear, the process of DABFSP is described in Fig. 2. In the processing stage, n jobs are assigned to a set of F identical and factories. The operation of job j that processed on the machine i is donated as $O_{j,i}$. Once the $O_{j,i}$ is started, it is not allowed to be interrupted during the processing time $P_{j,i}$. In the assembly stage, the n jobs are assembled into a set of s products $P = \{P_h | h = 1, 2, \ldots, s\}$. Each product consists of N_h defined jobs. The start assembly time of the product P_h is determined by the completion time of all the N_h jobs. To be specific, the N_h jobs are released to the assembly factory only when all the jobs that belong to the current product P_h is produced. There is only one assembly machine M_A in the assembly factory. Meanwhile, M_A handle only one product at a time, products have to be assembled in sequence. If the product $P_h (h > 1)$ is uncompleted, the next N_{h+1} jobs that belong to the product P_{h+1} have to remain in the processing factories until the assembly machine M_A is available for assembling.

A mixed integer linear programming (MILP) model is presented to describe the considered problem $DF\,|blocking|\sum TTD_j$ [34,47]. The goal is to minimize the total tardiness (TTD) objective in both the production stage and assembly stage.

$$\text{Minimize}: TTD \tag{1}$$

Subject to :

$$\sum_{f=1}^{F}\sum_{k=1}^{n} x_{k,i,f} = 1, \forall k \tag{2}$$

$$\sum_{i=1}^{m} x_{k,i,f} \le 1, \forall k, f \tag{3}$$

$$D_{j,0,f} \ge 0, \forall j, f \tag{4}$$

$$D_{k,1,f} \ge D_{k-1,0,f}, \forall k, f \tag{5}$$

$$D_{k,i,f} \ge D_{k,i-1,f} + \sum_{j=1}^{n} x_{j,i,f} \cdot P_{j,i}, \forall i \ge 2, k, f \tag{6}$$

$$D_{j,k,f} \ge D_{j-1,k+1,f}, \forall j \ge 2, i, f \tag{7}$$

$$D_j \ge D_{n,m,f} - L*\left(1 - x_{j,k,f}\right), \forall j, k, f \tag{8}$$

$$DA_h \ge DA_{h-1} + T_h, \forall h \tag{9}$$

$$DA_h \ge D_i + T_h - L*\left(1 - G_{j,h}\right), \forall j, h \tag{10}$$

$$C_{max} \ge DA_h, \forall h \tag{11}$$

where Eq. (1) shows the objective function. Eq. (2) indicates each job is allotted to only one position and each position is filled with one job in a certain factory. Eq. (3) ensures that no more than one is assigned per position in a factory. Eq. (4) implies the initial condition, which is a dummy value. Eq. (5) defines the departure time of the jobs processed on the first machine in each factory. Eq. (6) gives the relationship between the departure time of two adjacent processes of the same job in each factory. Eq. (7) gives the relationship of the departure time between two adjacent jobs and emphasizes the blocking constraint in the model. Eq. (8) determine the time for the job to be assembled. Eq. (9) determines the relationship between the assembly complete time of two adjacent products. Eq. (10) ensures that only one product is assembled for a time in the assembly factory. Eq. (10) defines the maximum assembly completion time. The TTD is divided into two parts: the tardiness time in all the processing factories and the tardiness time in the assembly factory. The computing method in the processing stage is shown as follows.

$$F_{1,i,f} = P_{\pi_1^f,i+1,f}, i = 1, 2, \ldots, m-1, f =, 2, \ldots, F \tag{12}$$

$$F_{\pi_j^f,i,f} = \max\left\{F_{\pi_{j-1}^f,i,f} - P_{\pi_j^f,i,f}, 0\right\} + P_{\pi_j^f,i+1,f} \tag{13}$$

$$j = 1, 2, \ldots, n^f, \underset{m-1}{i} = 1, 2, \ldots, \underset{n}{m} - 1, f = 1, 2, \ldots, F$$

$$C_{max,f} = C_{n^f,f} = \sum_{i=1}^{n^f,i,f} F_{n^f,i,f} + \sum_{j=1}^{n} P_{\pi_j^f,1,f}, f = 1, 2, \ldots, F \tag{14}$$

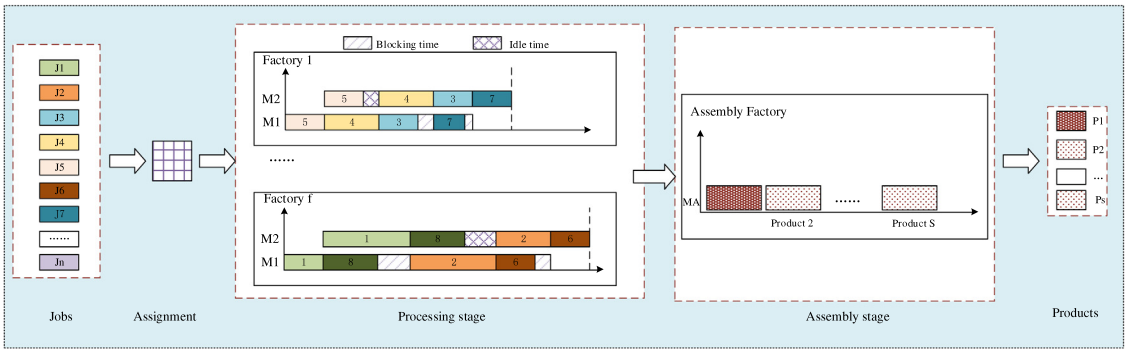


Fig. 2. The scheduling process of DABFSP.

$$C_{\pi_j^f,f} = C_{\pi_{j+1}^f,f} - P_{\pi_{j+1}^f,m,f}, j = n^f - 1, n^f - 2, \ldots, 1 \quad (15)$$

$$TD_f = \sum_{j=1}^{F} (\max(C_{\pi_j^f,f} - Due_{\pi_j^f}, 0)) \quad (16)$$

$$TD_F = \sum_{f=1} TD_f \quad (17)$$

where $F_{\pi_j^f,i,f}$ represents the completion time difference of the scheduled job π_j^f between machine i and machine $i+1$ in factory f. According to Eq. (16), the total tardiness in each factory is determined by the difference value between the completion time of each job on the last machine and the certain due time of the job.

In the assembly stage, the start assembly time of the product P_h is determined by the release time of all the jobs that belong to the product P_h. Therefore, the completion time of LJ_h is the start assembly time of the product P_h. According to Eq. (18), the tardiness time of the first product is 0 in the assembly factory. Then, the tardiness of the $2nd{\sim}snd$ products is calculated based on Eq. (19). If the time of the product $As(h)$, which is permitted to be assembled, is larger than the assembly completion time of the product $As(h-1)$, the tardiness time $TD_{As(h)}$ is 0. Otherwise, the time difference between the assembly completion time of the product $As(h-1)$ and the completion time of LJ_h is defined as the tardiness time between the product $AS(h-1)$ and product $AS(h)$.

$$TD_1 = 0 \quad (18)$$

$$TD_{AS(h)} = \{D_{AS(h-1)} - D_{LJ_h}, 0\}, h = 2, 3, \ldots, S \quad (19)$$

$$TD_S = \sum_{h=2} TD_{AS(h)} \quad (20)$$

$$TTD = TD_F + TD_S \quad (21)$$

3.2. Numerical illustration

An example with three products, two factories, eight jobs, two machines, and one assembly line is taken to further describe the considered problem. The relationship of the parameters is given in Table 2. Note that, the jobs that belong to the same product have been defined in the table. P_1 ={1,4,8}, P_2 ={2,7}, P_1 ={3,5,6}. The scheduling Gantt chart with the two-stage production process is shown in Fig. 3. Consuming that the final scheduling sequence found for the problem is π ={7,3,1,2;6,5,8,4}, which means that π^1 ={7,3,1,2} is the scheduling sequence of jobs assigned to factory 1, π^2 ={6,5,8,4} is the scheduling sequence of jobs assigned to factory 2. According to Fig. 3, in the processing stage, the TD_F is calculated based on Eqs. (12)–(17). $TD_F = \sum_{f=1}^{F} \sum_{j=1}^{n} (\max(C_{\pi_j^f,f} - Due_{\pi_j^f}, 0))$ =0 + 84 + 184 + 144 + 0 + 72 + 150 + 198 = 832. In the assembly stage, the assembly sequence of the product is $AS = \{P_3, P_2, P_1\}$. $LJ_1 = 4$, $D_{LJ_1} = 317$, $LJ_2 = 2$, $D_{LJ_1} = 308$, and $LJ_3 = 5$, $D_{LJ_1} = 180$. The TD_S is calculated based on Eqs. (18)–(20). $TD_S = \sum_{h=2}^{3} TD_{AS(h)} = 0 + 0 + 41 = 41$ Therefore, the $TTD = TD_F + TD_S = 873$.

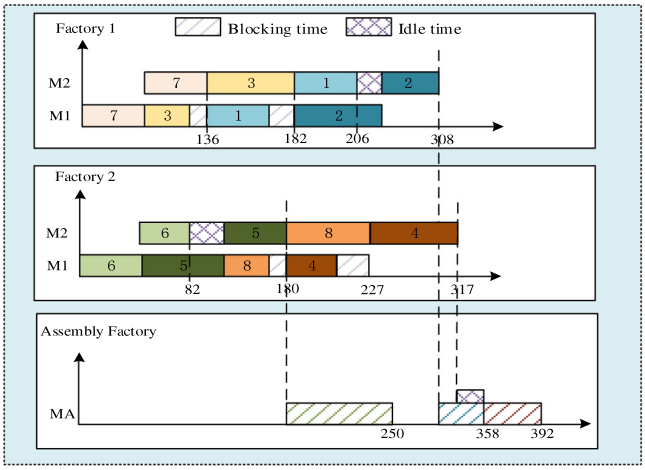


Fig. 3. The scheduling Gantt chart for the example instance.

4. Constructive heuristic

The constructive heuristic is commonly applied to construct a promising solution with a limited computational time on the flow-shop scheduling problems [5]. Furthermore, exploring the characteristics of the problem is conducive to extracting the specific-problem knowledge to construct an effective heuristic method [25]. Therefore, several characteristics of the considered problem $DF|blocking|\sum TTD_j$ are explored as follows.

(1) Characteristic 1: Jobs are allocated to any factory for processing. A product is allowed to be assembled only when all the jobs that belong to the product are processed.

(2) Characteristic 2: The assembly sequence of the products is determined by the completion time of the last processed job that belongs to the product.

(3) Characteristic 3: The total tardiness includes both the tardiness in the processing factories and the assembly factory. When the time the current product is permitted to be assembly factory is closer to the completion time of the previous assembled product, the tardiness is lower.

Inspired by the mentioned characteristics, three kinds of knowledge for the problem $DF|blocking|\sum TTD_j$ are extracted as follows.

Table 2
The data for the example.

Products	Jobs	Processing time				
		Machine 1	Machine 2	Due time	Assembly machine	Due time
1	1	41	24	65	75	75
	4	29	90	119		
	8	30	47	77		
2	2	75	51	126	50	50
	7	61	75	136		
3	3	52	46	98	70	70
	5	56	64	120		
	6	60	22	82		

Algorithm 1 KBNEH

Input: n
Output: $\prod[\pi_1, \pi_2, \cdots, \pi_f]$, TTD, f_c

Calculate $Due(j)$ for each job, $j = 1,2,\cdots,n$;
Obtain $\beta = \{\beta_1, \beta_2, \cdots, \beta_S\}$ by sorting products with an ascending order based on the assembly time of products P_h;
Obtain $\theta = \{\theta_1, \theta_2, \cdots, \theta_{N_h}\}$ by sorting the jobs of the products P_h with an ascending order based on the front delay time;
For $h = 1:S$
  For $k = 1$
    Select the first job in θ_k for $\beta_{h,1}$;
  For $k = 2:N_h$
    Evaluate the TTD when assigning the jobs to the position k, select the job with the smallest TTD;
  End for
End for
While $k \le f$
  $\pi_k(1) = \beta(k)$, $k = 1,2,\cdots,f$;
End while
$U = \{\beta_k\} \cup U$, $k = f+1,2,\cdots,n$;
While $k \le n$
  Obtained the π_l by inserting the job $\beta(k)$ in the factory resulting in the lowest TTD_{max}, $l = 1,2,\cdots,f$;
  Record the TTD, f_c;
End while

(1) Knowledge 1: The jobs belonging to an identical product are distributed evenly to each factory for processing. Knowledge 1 ensures that jobs belonging to the same product are processed in parallel to a certain extent. Therefore, the earliest starting assembly time of the product is more advanced than the jobs to be processed.

(2) Knowledge 2: The product with less assembly time is processed earlier.

(3) Knowledge 3: As an evaluation criterion, *TTD* is considered when constructing the initial solution.

In this paper, knowledge-based constructive heuristic KBNEH is designed as the heuristic method. The extracted knowledge is embedded into the original NEH algorithm. Supposing an example with eight jobs, two factories, two machines, and two products. The product P_1 includes jobs {1,2,4,8}. And the product P_2 includes jobs {3,5,6,7}. The job sequence is given as $\theta = \{1,2,4,8,3,5,6,7\}$, and the front delay time of each job is {19,23,77,21,56,51,45,18}. The assembly time for each product is {226, 130}. There are three procedures in KBNEH. For the first stage, the assembly time of products is sorted based on the ascending order. Therefore, the job sequence is updated as {3,5,6,7,1,2,4,8}. For the second stage, the jobs that belong to the same product are sorted in an ascending order based on the front delay time. The sequence of product P_2 is updated as {7,6,5,3}. The first partial sequence of P_2 is 7. The unscheduled jobs are assigned into the sequence one by one based on the NEH. Thus, the initial job sequence $\theta = \{7,6,5,3,1,8,4,2\}$ is calculated. In the last stage, the first f jobs in θ are selected for the first processed job in each processing factory. For the unscheduled jobs, the job is allocated to all the possible positions in the processing factories. The position with the lowest *TTD* is determined to insert the job. The final job sequence is intended as the initial solution. The procedure of KBNEH is shown in Algorithm 1.

5. Water wave optimization with problem-specific knowledge

5.1. Water wave optimization

There are three fundamental operations, including propagation, breaking, and refraction operations to perform in WWO. The collective intelligence of WWO is achieved by a simple and effective mechanism via obtaining an effective equilibration between exploration and exploitation during iterations. The procedure of the traditional WWO is described as follows.

(1) Initialization: randomly generate N_p waves, where N_p is a given number.

(2) Propagation: each wave in the population spreads outwards and searches in the global scope for a new worthful wave.

(3) Breaking: local reinforce search for the waves found in the propagation stage. The new wave is promising to be the optimum solution.

(4) Refraction: a disturbance operation is performed on the wave to generate a new solution to increase the probability of escaping the local optima.

The traditional WWO is initially proposed to solve continuous optimization problems. Thus, the DABFSP with discrete characteristics is unable to be optimized by WWO. Due to the equilibration between exploration and exploitation obtained by the dynamic cooperation of three fundamental search phases of WWO, effective technologies are promising to be embedded in WWO. Given the above reasons, the WWO with problem-specific knowledge (KWWO) is presented to solve the considered problem $DF\,|blocking|\sum TTD_j$ in this paper. In the following subsections, each phase of KWWO is described in detail.

5.2. Initialization in KWWO

WWO begins with a population with N_p individuals. Herein, the KBNEH is embedded in the initialization of KWWO to obtain

a desirable solution with high quality. Additionally, to ensure the diversity of the population, the heuristic NR_a proposed by Shao et al. [31] is adopted as the allocation rule to generate the remaining solutions after initializing the job sequence randomly. The pseudocode of the initialization is displayed in Algorithm 2.

Algorithm 2 Initialization

Input: N_p
Output: $S,\ TTD,\ S_{best},\ TTD_{best}$

Obtain an initial solution π_1 based on Algorithm 1;
Calculate TTD_s;
$S_1 = \pi_1,\ TTD_1 = TTD_{\pi_1}$;
For $i = 2: N_p$
  $initialsequence = random(job)$;
  π_i is allocated according to the job allocation rule NR_a;
  Calculate the fitness of π_i;
  $S_i = \pi_i,\ TTD_i = TTD_{\pi_i}$;
End for
$S_{best} = Min(S),\ TTD_{best} = S_{best}$;

5.3. The neighborhood search based on the problem-specific knowledge

The population-based metaheuristic, which belongs to an iteration approach, has a requirement to take time to obtain more promising solutions. In theory, most of the initial solutions are far from the optimal solution in the early iterations. The algorithm explores the global range to maintain the population with sufficient diversity and prevent premature convergence of the population. The solutions gradually approach the optimum by deep local search in the iterative process of the algorithm. WWO is such a metaheuristic. In WWO, the propagation operation is in favor of exploring large scopes. The breaking operation and refraction operation help local exploitation. Therefore, the evolutionary process of a solution is strictly following a global search first and then a local search during the iterations. Based on the above analysis, make the individuals that far from the optimal solution perform the global search and then local search, and make the individuals that close to the optimal solution conduct the local search, which ensures that more potential solutions are searched in a limited time. Therefore, the neighborhood search approach based on the problem-specific knowledge is designed to balance dynamically the exploration and exploitation of the algorithm.

In KWWO, the population is grouped into three kinds of sub-populations. The way to distribute the sub-populations is according to the normal distribution, which is an extremely common continuous probability distribution [48]. The probability density function curve of the normal distribution is bell-shaped. where most of the statistical data are concentrated in the intermediate stage, and the data at both ends can be considered to represent relatively extreme situations. The normal distribution has wide applicability in social life and production practice, such as the group distribution law of height, blood pressure, test scores, and measurement errors. After the initialization, the individuals are evaluated. The fitness of each individual is considered as the indicator of distribution. The three kinds of sub-populations include the superior sub-population (the top 15% individuals selected from the population), the ordinary sub-population (the middle 70% individuals selected from the population), and the inferior sub-population (the lower 15% individuals selected from the population). The illustration of the neighborhood search based on the problem-specific knowledge is shown in Fig. 4. Individuals in different sub-populations choose the evolutionary strategies adaptively. Individuals in the ordinary sub-population conduct propagation operation and then breaking operation or refraction operation. The breaking operation to find the optimal solution is performed on individuals in the superior sub-population. For those in the inferior sub-population, the refraction operation is conducted to find valuable individuals and replace them. After the current iteration, a new population is generated. The operations in KWWO are performed dynamically on the individuals to achieve the equilibration between exploration and exploitation during the search phase.

5.4. Design of exploration phase

The propagation operation plays a significant role in searching for promising individuals. Namely, the ability of the exploration in basic WWO is determined by the propagation operator to a large extent. Thus, an effective neighborhood operator, which is performed both on the critical factory and the non-critical factories, is presented to access a desirable solution.

The destruction operation and the reconstruction operation, are designed in the proposed propagation operator to generate a new solution. In the destruction phase, p jobs are selected to form a temporary partial sequence γ. p_c jobs are selected randomly and removed from the critical factory. Here, the factory with the lowest value of *TTD* is defined as the critical factory. For the remaining $(p - p_c)$ jobs, a job is selected and removed from the non-critical factory each time. The selection is executed for $(p - p_c)$ times. In the reconstruction phase, the jobs in γ are reinserted into all the positions in order in the processing factory, and the position with the lowest *TTD* is decided to insert the job. As the description in Fig. 5, The job sequence π is set as {7,3,1,2;6,5,8,4}. Supposing the factory 1 is the critical factory f_c. In the destruction phase, job 1 is selected randomly and removed from f_c, job 5 is selected and removed from the non-critical factory. Thus, $\gamma = \{1,5\}$, $\pi' = \{7,3,2;6,8,4\}$. In the reconstruction phase, job 1 is tested on all the eight positions to calculate the *TTD*, the final position is determined based on the lowest *TTD*. The pseudocode of the propagation is displayed in Algorithm 3.

Algorithm 3 Propagation operation

Input: $N_p\ S,\ fit,\ p_c$
Output: $S_{new},\ fit_{new}$

For $i = 1: N_p$
  Find the critical factory f_c in S_i;
  Obtain γ by removing p_c jobs from the π_{f_c} and $(p - p_c)$ jobs from other factories;
  For $k = 1: p$
    Insert job $\gamma(k)$ in the position that results in the lowest TTD;
  End for
  Obtain S_{new};
  $fitness_{new} = TTD$;
End for

5.5. Design of exploitation phase

5.5.1. Breaking operation with local reinforcement strategy

In canonical WWO, the breaking operation is responsible to exploit the neighborhood region of the preponderant individual found by the propagation to generate a new individual. The new individual is promising to be the new optimum [49]. Whereas, the local search capability of the breaking operation is relatively weak. To tackle the problem, a breaking operation with a local reinforcement strategy under the framework of the variable neighborhood search strategy (VNS) [50] is proposed to enhance the performance of the breaking operation. Eight kinds of sequence-related local search methods are designed in the local reinforcement strategy. The methods including subsequence swapping in single-factory (SS_s), subsequence insertion

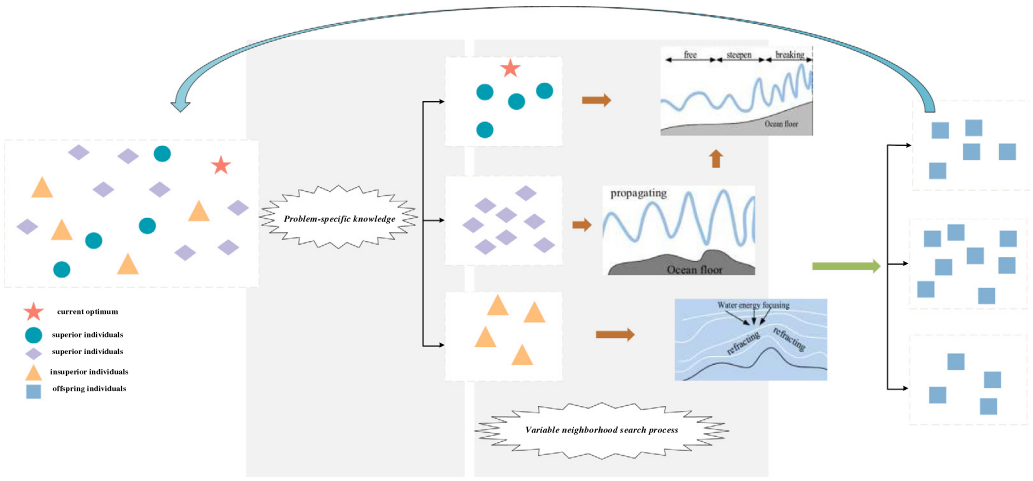


Fig. 4. The illustration of the neighborhood search based on the problem-specific knowledge.

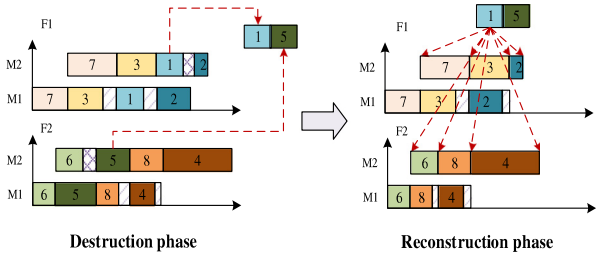


Fig. 5. The process of destruction–construction strategy.

in single-factory (SI_s), subsequence swapping in multi-factory (SS_m), subsequence insertion in multi-factory (SI_m), single-point swapping in single-factory (SPS_s), single-point insertion in single-factory (SPI_s), single-point swapping in multi-factory (SPS_m), single-point insertion in multi-factory (SPI_m). The illustration of the local search methods is given in Fig. 6. π_1, π_2 are the job sequences in two random factories.

SS_s: select a factory randomly, the swap operator is performed on the jobs in the subsequence. The subsequence, which is a sequence with φ jobs, is selected randomly from the chosen factory.

SI_s: select a factory randomly, then select randomly the insert point and the subsequence. The subsequence is inserted into the next position of the insert point.

SS_m: select two different factories randomly, then select two subsequences with equal length from the chosen factory, respectively. Swapping the two subsequences.

SI_m: select two different factories randomly, then select two subsequences with equal length from the chosen factory, respectively. Then, the insert and swap operators are both executed on the two subsequences.

SPS_s: two different jobs are chosen as the first point and the second point from a randomly selected factory. Then the two jobs are swapped.

SPI_s: the first point is inserted into the next position of the second point.

SPS_m: two jobs are chosen as the first point and the second point from two randomly selected factories, respectively. Then the two jobs are swapped.

SPI_m: the first point is inserted into the next position of the second point selected from another factory.

The local search methods are grouped into four configurations including $operator_1 = \{SS\_s, SI\_s\}$, $operator_2 = \{SS\_m, SI\_m\}$, $operator_3 = \{SPS\_s, SPI\_s\}$, $operator_4 = \{SPS\_m, SPI\_m\}$. To enhance the quality of solution for the optimum and maintain the diversity of the population, the four proposed configurations are executed on the current best solution under the framework of VNS. The details of the search phase are given in Algorithm 4.

Algorithm 4 Breaking operation

Input: the current solution S_i, $fitness_i$
Output: the new solution S_{new}, $fitness_{new}$

Set $K_{max} = 4$, $K = 1$, $fitness_0 = fitness_i$;
While $K \le K_{max}$
  Switch K
    Case 1: $S_{temp} = operator_1(S_i)$;
    Case 2: $S_{temp} = operator_2(S_i)$;
    Case 3: $S_{temp} = operator_3(S_i)$;
    Case 4: $S_{temp} = operator_1(S_i)$;
  End switch
  If $fitness_{S_{temp}} < fitness_0$
    $fitness_0 = fitness_{S_{temp}}$, $K = 1$, $S_i = S_{temp}$;
  else
    $K = K + 1$;
  End if
  $S_{new} = S_i$, $fitness_{new} = fitness_{S_{temp}}$;
End while

5.5.2. Refraction operation based on path-relinking

For the refraction operation in WWO, the individual that is unable to find a desirable individual after a few iterations are removed from the current population. Then a new individual is generated by perturbing the optimal individual. Therefore, the refraction operation in KWWO is modified to address the proposed problem with discrete characteristics. The path-relinking strategy [51], which is an effective search method to explore the neighborhood of the two given solutions, is applied in the phase to generate a new worthful solution. Here, the individual with the best performance S_{best} are selected to compare with the current individual S_i. To transform S_i to S_{best}, the insertion or swapping operator is executed on the current individual S_i, and an intermediate solution is generated after each time performing an operator. In the end, all the intermediate solutions are evaluated

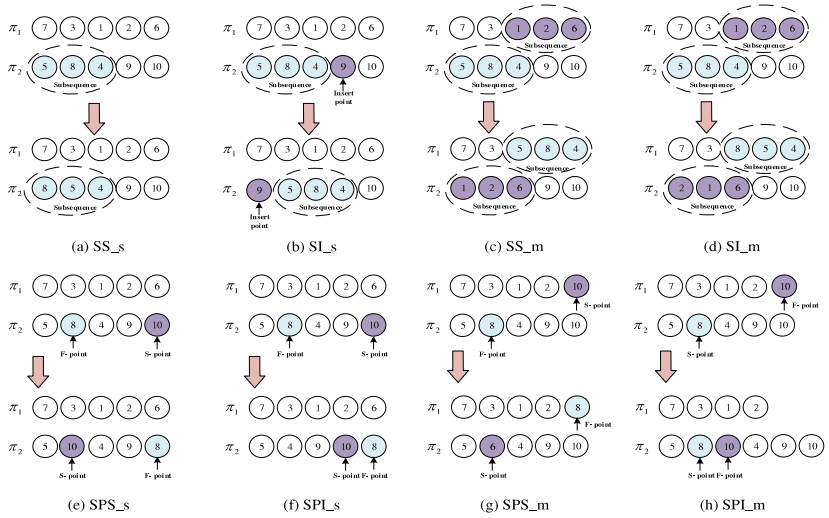


Fig. 6. Illustration of the local search methods.

and the solution with the lowest fitness is remained. The pseudocode of the refraction is displayed in Algorithm 5. The process of generating a new individual is described in Fig. 7. As shown in Fig. 7, the $\pi_{best} = \{1,2,3,4,5\}$ is the guiding solution, $\pi_{current} = \{5,4,3,2,1\}$ is the current initial solution. Three intermediate solutions, including {1,5,4,3,2}, {1,2,5,4,3}, {1,2,3,5,4} are generated by selecting the insertion operator, but only one intermediate solutions {5,2,4,3,1} is generated by performing path-relinking with swapping operator.

5.6. The framework of KWWO

According to the discussions on each operation of KWWO, thus, the integral operation mechanism in KWWO is expounded detailedly in this section. Meanwhile, the general procedure of KWWO is given in Algorithm 6. The flowchart of KWWO is described in Fig. 8. The knowledge-based KBNEH is designed as the constructive heuristic algorithm to obtain the desired solution in the initialization phase. Meanwhile, the heuristic NR_a is adopted to generate the population with high quality and diversity.

During the algorithmic search process, the solution in different sub-populations spontaneously selects the operations. The propagation operation is performed on the individuals, which are part of the ordinary sub-population, to explore large scopes for improving the diversity of the population. The next search behavior is determined by the comparison between the parent solutions and the offspringing solutions. On the one hand, if the offering solution is better than the parent solution and the historical optimal solution, the breaking operation based on with local reinforcement strategy under the framework of VNS is performed on the offering solution. Moreover, the individuals, which are distributed in the superior sub-population, further exploit small scope by performing the breaking operation to obtain the current local optimal solution. On the other hand, if the offspring individual is worse than the parent individual, the wave height of the offspring solution is reduced. Once the wave height is reduced to 0, the refraction operation is executed on the current individual. Furthermore, for the poor sub-population, the refraction operation based on the path-relinking method is applied to generate a new individual between the individual and the current optimum. The enhanced refraction operation is effective to avoid algorithm search stagnation. The trade-off between maintaining high diversity and encouraging fast convergence is achieved by KWWO.

Algorithm 5 Refraction operation

Input: the current solution S_i, S_{best}, n
Output: the new solution S_{new}, $fitness_{new}$

For $h = 1{:}n$
  If $S_i(h) \neq S_{best}(h)$
    Generate u randomly;
    Find job δ in S_i which is identical to $S_{best}(h)$;
    If $\mu > 0.5$
      Remove job δ from the position of S_i and insert it into position h of S_i;
    Else
      Swap the position of job δ and $S_i(i)$;
    End if
    If the new solution is not identical to S_{best}
      Put the new solution s into an external archive A;
    End if
  End if
End for
Obtain TTD_{min} by calculating the TTD of the solutions in A;
$S_{new} = s_{min}$, $fitness_{new} = TTD_{min}$;

Algorithm 6 KWWO

Input: N_p, PT, h_{max}
Output: TTD_{best}

//initialization
Obtain S, TTD based on Algorithm 2;
Record S_{best}, TTD_{best};
While the stop criterion is not satisfied
  Generate sub-populations based on the fitness;
  For $i = 1{:}N_p$
    If the individual is part of the ordinary sub-population;
      //propagation
      Obtain S_p and TTD_p based on Algorithm 3;
      If $TTD_p < TTD_{prey}$
        $S_{prey} = S_p$, $TTD_{prey} = TTD_p$;
        If $TTD_p < TTD_{best}$ ‖ the individual is part of the superior sub-population or S_{best}
          $S_{best} = Solution_p$, $TTD_{best} = TTD_p$;
          //breaking
          Obtain S_b and TTD_b based on Algorithm 4;
          If $TTD_b < TTD_{best}$
            $S_{best} = S_b$, $TTD_{best} = TTD_b$;
          End if
          $h = h_{max}$;
        Else $h--$;
          If $h = 0$ ‖ the individual is part of the inferior sub-population
            //refraction
            Obtain S_r and TTD_r based on Algorithm 5;
            If $TTD_r < TTD_{best}$
              $S_{best} = S_r$, $TTD_{best} = TTD_r$;
            Else
              $S_i = S_r$, $TTD_i = TTD_r$;
            End if
          End if
        End if
      End if
    End if
  End for
End while

5.7. The time complexity of KWWO

The meta-heuristic algorithm obtains an approximate optimal solution by the iteration approach. Therefore, time complexity

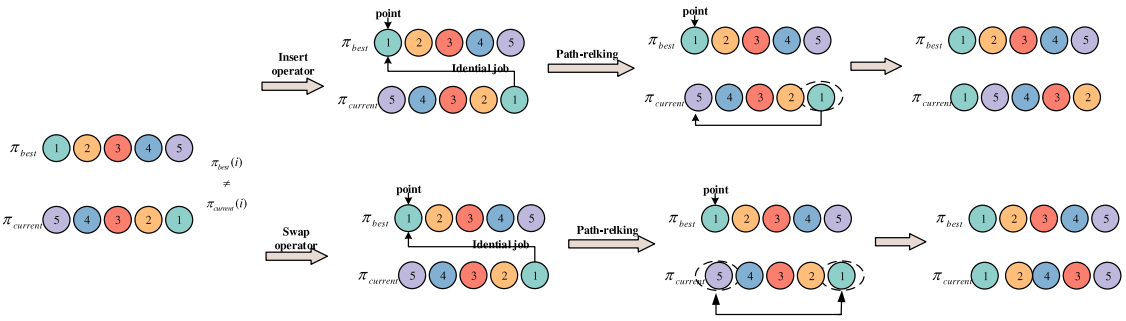


Fig. 7. Description of the path-relinking strategy.

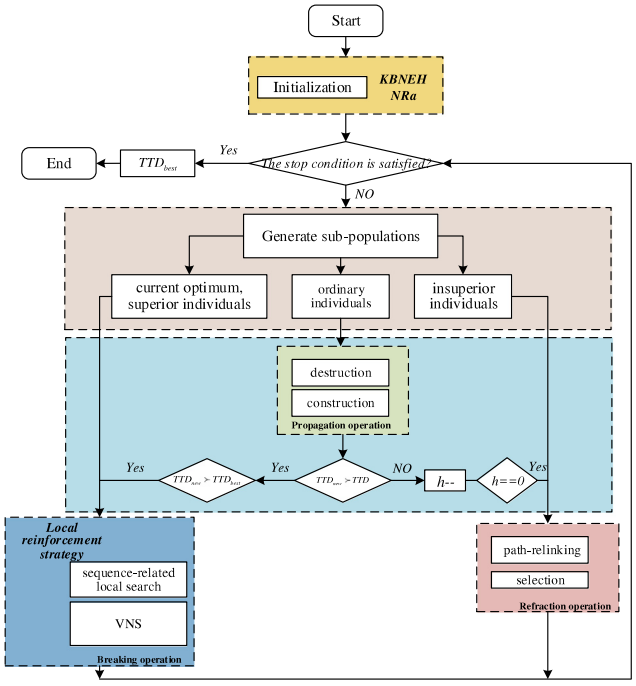


Fig. 8. The flow chart of KWWO.

plays a role in analyzing the performance of the algorithm. In this section, the time complexity of the proposed KWWO is analyzed in detail. Suppose that the DABFSP is proposed with n jobs, f factories, and s products. Meanwhile, the population size of KWWO is set to N_p, the number of iterations is T. Note that, there is only one machine during the assembly stage, the time complexity for assembling products is $O(1)$. KWWO consists of four phases, including the initialization for allocating jobs to the factories and three main operations for further search. In the initial phase, the individuals are generated based on Algorithm 2. The time complexity of generating the initial solutions is $O(n^2f)*O(s)$. In the propagation phase, the time complexity of the destruction and reconstruction operation is $O(N_p)$. Furthermore, the time complexity in the breaking phase is decided by the local reinforcement strategy and the VNS. Thus, the time complexity is $O(n^4f)$. For the refraction phase, the time complexity for the worst case is $O(n*f)$. In short, the time complexity of KWWO is calculated as follows.

$$\begin{aligned} &O\left(n,f,s,N_p,T\right) = O\left(T\right)*[O\left(n^2f\right)+O\left(N_p\right) \\ &+O\left(n^4f\right)+O\left(n*f\right)]*O\left(s\right) \\ &\approx O\left(T\right)*O\left(n^4f\right)*O\left(s\right) \\ &\approx O(Tn^4fs) \end{aligned}$$

Table 3
The data of the benchmark instances.

Scale	n	m	F	s
Small-scale	{8,12,16,20,24}	{2,3,4,5}	{2,3,4}	{2,3,4}
Large-scale	{100,200,500}	{5,10,20}	{4,6,8}	{30,40,50}

6. Experimental results and analysis

In this section, numerical tests based on two sets of benchmark instances are utilized to evaluate the performance of the proposed KWWO. The data of the benchmark instances are shown in Table 3. Therefore, the first set consists of 900 small-scale instances, where including 180 combinations of n, m, F, and s with each combination having 5 instances. The second set consists of 810 large-scale instances, where 81 combinations of n, m, F, and s with each combination has 10 instances are included. The design of the due date for each job is $Due_j = \tau * P_j$. Here, the value of τ is set to 1. To make a fair comparison, all the computational experiments are carried out on the personal computer with an Intel(R) Core (TM) i7-10510H CPU @ 2.30 GHz with 16 GB RAM in the Windows 10 Operation System. The algorithms are tested 5 times with the same termination criterion $t_{max} = 10*n*m*f$ milliseconds. Average relative percentage deviation (ARPD) [20] over the solutions found in the test is adopted as the performance measure. The index of ARPD is calculated as follows.

$$ARPD = {}^{1}/_{R_{max}} * \sum_{R=1}^{R_{max}} {}^{(TTD_i - TTD_{min})}/_{TTD_{min}} * 100 \quad (22)$$

where TTD_i is the makespan obtained by the algorithm and TTD_{min} is the lowest makespan obtained in the experiments. R_{max} is the time to run the algorithm on each instance. The smaller the ARPD is, the performance of the algorithm is better.

6.1. Parameter calibration

The main control parameters in KWWO include the population size (N_p), the wave height (h_{max}), and the destruction rate (D_s). The design of experiments (DOE) [52] is adopted to design the test for the impact of the control parameters. Some preliminary experiments are firstly conducted to determine the potential levels of all parameters. Here, the levels of each control parameter are listed as follows: $N_p \in\{5,10,30\}$, $h_{max} \in\{3,4,5,6\}$, $D_s \in\{0.1,0.3,0.5,0.6\}$. Therefore, three parameters are leading to a total of $3*4*4=48$ configurations. Meanwhile, the test set consists of 64 instances, where n is set as {100,200}, m is set as {5,10}, F is set as {4,6}, and s is set as {30,40}. In the implemented test set, each instance is tested 5 times and the average ARPD value of 5 runs is obtained for comparison.

Table 4
Results of ANOVA for parameters calibration.

Source	Sum of squares	Degrees of freedom	Mean square	F-ratio	P-value
N_p	47 878.6	2	23 939.3	3607.32	0
h_{max}	6.7	3	2.2	0.34	0.7992
D_s	48.7	3	16.2	2.44	0.0974
$N_p * h_{max}$	64	6	10.7	1.61	0.2025
$N_p * D_s$	291.6	6	48.6	7.32	0.0004
$h_{max} * D_s$	90.8	9	10.1	1.52	0.2147
Error	119.5	18	6.6		
Total	48 499.7	47			

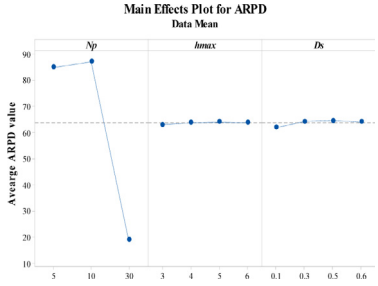


Fig. 9. Main effects plot of the control parameters.

Additionally, the analysis of variance technique (ANOVA) [53] is utilized to analyze the effects of different levels on the experimental results. The results of ANOVA are given in Table 4, where the larger the F-ratio, the smaller the P-value, and the greater the significant effect on the performance of KWWO. Therefore, the population size N_p has the most significant effect on the performance of the algorithm according to Table 4. Meanwhile, the main effects plot of the control parameters is presented in Fig. 9 to determine the value of the control parameters. From Fig. 9 (N_p), KWWO performs better at $N_p = 30$ than $N_p = 5$ and $N_p = 10$. Additionally, the P-value of $N_p{*}D_s$ is less than 0.05, which means the main effects plot is insufficient to determine the final level of the N_p and D_s. Hence, the interaction effect plots of $N_p{*}D_s$ and $h_{max}{*}D_s$ are shown in Fig. 10. Compared with Fig. 10(a), when the N_p is set to 30, the performance of KWWO is better when $D_s = 0.1$ than any other levels. Additionally, Fig. 10(b) shows that the performance of KWWO is better at $h_{max} = 3$ when the D_s is determined as 0.1. Therefore, the satisfactory configuration of the control parameters in KWWO is set as $N_p = 30$, $h_{max} = 3$, $D_s = 0.1$.

6.2. Effectiveness analysis of the components in KWWO

The performance of KWWO is improved by the modified operations in this paper. The dominant components in KWWO are divided into four parts, including the initialize method, the propagation phase with the destruction–construction operator, the breaking phase with local reinforcement strategy, and the refraction phase based on the path-relinking method. To analyze the effectiveness and influence of each component, two sets of simulation experiments are designed. The first set of simulation experiments is designed to compare WWO with WWO/In (WWO with the initial method), WWO/In/P (WWO with the initial method and the modified propagation operation), WWO/In/P/B (WWO with the initial method, the modified propagation operation, and the modified breaking operation), and KWWO. In the second set of simulation experiments, KWWO is carried out to compare with KWWO-In (KWWO without the initial method), KWWO-P (KWWO without the destruction–construction operator in the propagation phase), KWWO-B (KWWO without the local reinforcement strategy in the breaking phase), and KWWO-R (KWWO without path-relinking method in the refraction phase). The tests are carried out by the 810 large-scale instances, which are shown in Table 3.

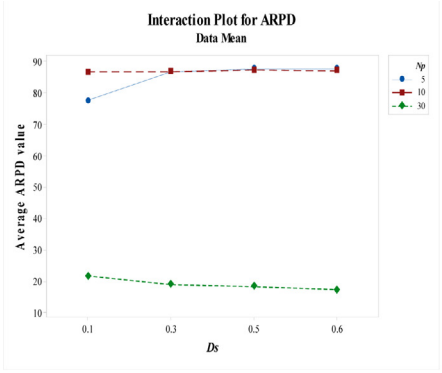
The first set of simulation experiments is to add the proposed methods to WWO gradually. The results of the experiment are shown in Table 5. For a clear comparison of the effectiveness of each component in improving the performance of WWO. Fig. 11 is utilized for the visualization of experimental results. Table 5 reveals that the average ARPD values of KWWO are smallest than the average ARPD values of the other algorithms. Specifically, the ARPD values of KWWO are superior to the comparison algorithms on almost all the combinations of f, n, and m except for the $6 * 100 * 5$, $6 * 100 * 10$, $8 * 200 * 20$, and $8 * 500 * 20$. The average ARPD values are reduced as more methods are embedded into the basic WWO. Furthermore, the decreasing order of the average ARPD values is WWO, WWO/In, WWO/In/P, WWO/In/P/B, and KWWO. The order shows that the performance of the KWWO is better when more components are introduced in WWO. According to Fig. 11, most of the red spots representing the KWWO are at the bottom of the whole figure, which means the performance of KWWO is superior to WWO/In, WWO/In/P, and WWO/In/P/B in most of the test instances. The experimental results indicate that the three methods introduced in KWWO are useful to enhance the ability of searching.

To analyze the impact of each component in improving the performance of KWWO, the adopted components are removed from the proposed algorithm individually. The results of the second set of simulation experimental are listed in Table 6. The visualization of experimental results is given in Fig. 12. From the results, the average ARPD values of KWWO are smallest than the overage ARPD values of all the other algorithms. It demonstrates that KWWO, which is enhanced by the multiple strategies, is superior to the compared algorithms. All the improved strategies influence the performance of KWWO. Compared to the other algorithm, the average ARPD values of KWWO-P are the largest, which explains that KWWO-P obtains higher ARPD values for most of the instances. Additionally, the results demonstrate that the propagation operation enhanced by the destruction–construction method has the most significant impact on KWWO. The reasons lie in the balance between the exploration and exploitation is achieved by the propagation operation. Once the destruction–construction method is removed, the ability for KWWO-P to explore the global scopes is lost, which leads to the algorithm being caught in a local search. Furthermore, it is observed that when the number of factories is 4, the number of jobs is 500, the ARPD value of KWWO-In is smaller than the ARPD value of KWWO-P but larger than the other algorithm, and the ARPD values of KWWO-In get larger as the number of factories and jobs increases. Therefore, the proposed initial method has vital effectiveness on KWWO. The proposed initial method plays an important role in obtaining worthful initial solutions with a very limited computational time. The initial method removed leads to the convergence rate of the algorithm being weakened. Based on the above analysis, the influence of the different strategies in KWWO is different, the propagation operation has the most significant impact on KWWO.

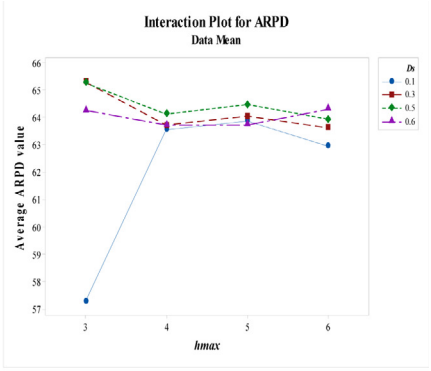
6.3. Effectiveness analysis of the initial method

6.3.1. The effectiveness of each rule in KBNEH

The constructive heuristic KBNEH is designed by incorporating three knowledge-based rules into NEH. Therefore, the experiment is applied to verify the effectiveness of each rule. In this experiment, KBNEH is compared with KBNEH-R_1, KBNEH-R_2, KBNEH-R_3.



(a). $N_p * D_s$.



(b). $h_{max} * D_s$.

Fig. 10. The interaction plots of $N_p * D_s$ and $h_{max} * D_s$.

Table 5
Results of the first set of simulation experiments.

F	$n * m$	ARPD				
		WWO	WWO/In	WWO/In/P	WWO/In/P/B	KWWO
	100 * 5	7.208	4.993	2.543	3.469	**2.204**
	100 * 10	8.275	5.346	2.517	3.899	**2.443**
	100 * 20	7.810	5.904	2.075	3.376	**2.046**
	200 * 5	3.548	5.485	3.395	3.022	**1.949**
4	200 * 10	3.268	3.621	1.933	1.832	**1.283**
	200 * 20	3.677	3.343	2.420	2.519	**0.884**
	500 * 5	1.282	7.181	5.721	5.557	**0.803**
	500 * 10	1.368	4.400	3.372	3.277	**0.649**
	500 * 20	1.893	3.567	3.448	3.482	**0.532**
	100 * 5	9.802	5.316	**2.014**	4.832	2.042
	100 * 10	12.301	7.220	**2.307**	5.366	2.414
	100 * 20	12.876	8.962	1.601	6.111	**1.514**
	200 * 5	7.047	5.565	3.543	2.511	**2.424**
6	200 * 10	6.630	4.962	1.784	3.082	**1.427**
	200 * 20	7.149	5.275	1.650	2.822	**1.509**
	500 * 5	2.310	4.698	3.566	3.520	**0.900**
	500 * 10	1.945	4.456	2.966	2.920	**0.679**
	500 * 20	2.284	3.732	2.689	2.681	**0.441**
	100 * 5	14.070	7.225	7.393	1.783	**1.726**
	100 * 10	15.735	8.201	7.192	1.804	**1.499**
	100 * 20	16.955	10.786	8.450	1.860	**1.820**
	200 * 5	9.215	5.349	4.409	1.919	**1.710**
8	200 * 10	10.626	6.921	1.611	4.846	**1.610**
	200 * 20	10.806	8.490	**1.430**	5.781	1.625
	500 * 5	2.768	4.511	2.539	2.783	**0.832**
	500 * 10	3.199	4.305	1.878	1.902	**0.891**
	500 * 20	3.558	1.500	1.741	**0.593**	3.681
Average		6.948	5.604	3.192	3.243	**1.538**

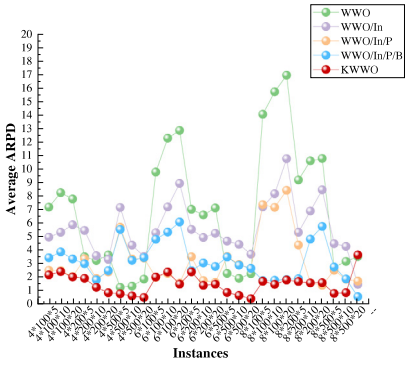


Fig. 11. The visualization of the first set of simulation experimental results.

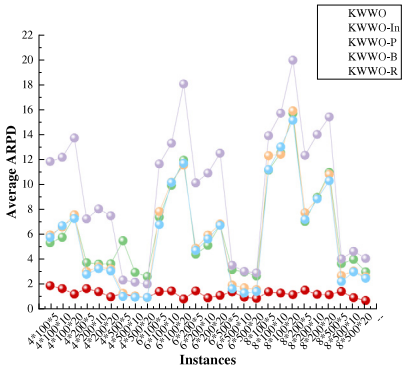


Fig. 12. The visualization of the second set of simulation experimental results.

Table 6
Results of the first set of simulation experiments.

F	$n * m$	*ARPD*				
		KWWO	KWWO-In	KWWO-P	KWWO-B	KWWO-R
	100 * 5	**1.934**	5.375	11.865	5.987	5.800
	100 * 10	**1.708**	5.805	12.200	6.610	6.724
	100 * 20	**1.263**	7.492	13.754	7.600	7.302
	200 * 5	**1.708**	3.786	7.271	3.087	2.847
4	200 * 10	**1.450**	3.671	8.067	3.438	3.297
	200 * 20	**1.037**	3.700	7.523	3.278	3.112
	500 * 5	1.318	5.530	2.386	1.333	**1.072**
	500 * 10	1.021	2.999	2.221	1.121	**1.016**
	500 * 20	0.996	2.679	2.067	**0.975**	0.989
	100 * 5	**1.471**	7.409	11.675	7.858	6.809
	100 * 10	**1.520**	9.953	13.330	10.211	10.185
	100 * 20	**0.878**	11.974	18.066	11.582	11.721
	200 * 5	**1.520**	4.460	10.156	4.935	4.763
6	200 * 10	**0.961**	5.165	10.935	5.974	5.669
	200 * 20	**1.161**	6.781	12.534	6.858	6.754
	500 * 5	**1.435**	3.219	3.575	1.986	1.679
	500 * 10	**1.015**	3.035	3.081	1.768	1.350
	500 * 20	**0.918**	2.715	2.965	1.628	1.470
	100 * 5	**1.446**	11.169	13.930	12.333	11.227
	100 * 10	**1.345**	12.552	15.738	12.444	13.036
	100 * 20	**1.220**	15.723	19.964	15.926	15.163
	200 * 5	**1.580**	7.057	12.364	7.763	7.220
8	200 * 10	**1.237**	8.993	14.033	8.870	8.870
	200 * 20	**1.212**	10.999	15.437	10.846	10.324
	500 * 5	**1.477**	3.701	4.084	2.747	2.264
	500 * 10	**0.975**	4.049	4.688	3.108	3.042
	500 * 20	**0.749**	3.053	4.117	2.644	2.535
Average		**1.280**	6.409	9.557	6.034	5.787

Each of the comparison algorithms represents that one rule is removed from KBNEH. Furthermore, the effect of removing one of the rules in constructing the job sequence is replaced by the original job sequence given by the customers. The number of

Table 7
The ARPD values obtained by KBNEH, KBNEH-R_1, KBNEH-R_2, KBNEH-R_3.

n_m_F_s	Algorithms			
	KBNEH	KBNEH-R_1	KBNEH-R_2	KBNEH-R_3
8_2_2_2	0	4.143	2.072	6.780
8_2_2_4	0	3.120	13.345	25.390
8_2_4_2	0	8.571	23.214	0
8_2_4_4	0	6.271	13.531	1.980
8_5_2_2	0	5.697	3.357	24.212
8_5_2_4	0	15.407	39.704	34.963
8_5_4_2	0	26.895	14.670	34.230
8_5_4_4	0	20.450	13.906	46.830
24_2_2_2	0	0.295	3.614	27.305
24_2_2_4	0	3.186	1.685	2.563
24_2_4_2	0	5.029	11.022	43.713
24_2_4_4	0	2.447	0.499	50.587
24_5_2_2	0	5.153	1.456	4.891
24_5_2_4	0	1.658	10.723	10.084
24_5_4_2	0	6.362	26.042	41.146
24_5_4_4	0	8.020	1.182	15.507
Average	0	7.669	11.251	25.480

jobs (n), machines (m), factories (F), and products (s) are set as n{8,24}, m{2,5}, F{2,4}, s{2,4}. Each algorithm is independently performed 5 times on each instance, and the average ARPD values are summarized. The configurations in the test along with the ARPD values of the experimental results are summarized in Table 7.

As seen in Table 7, KBNEH achieves the lowest average ARPD with 0 on all of the instances. It means that three knowledge-based rules are effective in constructing feasible solutions. This is because the three rules construct the initial solution from the perspective of job processing and product assembly. When the rules are removed, the job processing sequence and product assembly sequence are determined according to the order sequence. Additionally, the performance of KBNEH-R_1, KBNEH-R_2, and KBNEH-R_3 is different in different configurations. According to the average ARPD values of the other algorithms, KBNEH-R_3 gives the worst result, followed by KBNEH-R_2. Firstly, when makespan replaces TTD as the comparison standard for each job insertion, that is, the blocking time generated in the job processing process is ignored to some extent. The result is an increase in the total tardiness time. Secondly, for the product assembly stage, the machine follows the principle of first come and first process rather than order requirements, which greatly reduces the tardiness time. Besides, the jobs belonging to the same product are evenly distributed in each factory. This rule ensures basically that there is little difference in the processing completion time of all jobs. Therefore, there are no jobs that belong to the same product spend plenty of time waiting for other jobs that belong to the same product to process. Based on the analysis above, each of the knowledge-based rules is effective in KBNEH.

6.3.2. The effectiveness of KBNEH

In this section, the knowledge-based constructive heuristic named KBNEH is designed as the initial method by analyzing the characteristics of the DABFSP with minimizing the total tardiness criterion. The computational experiments based on the proposed benchmark set are executed on KBNEH and three other algorithms to compare the performance of KBNEH in solving the DABFSP. The compared algorithms, which have proven to be effective in constructing feasible solutions quickly, including NEH [54], DNEH [6], NR_a [31]. The tests are carried out by the 900 small-scale instances and 810 large-scale instances.

The average ARPD values of the experimental results are summarized in Tables 8 and 9 according to the number of jobs,

Table 8
The average ARPD values on the small-scale benchmark set.

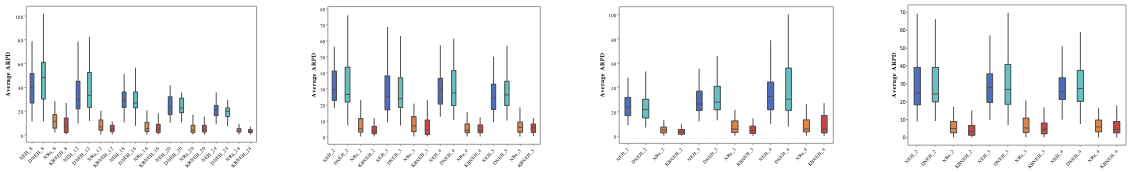
Parameter	Scale	Mean ARPD			
		NEH	DNEH	NR_a	KBNEH
	8	41.491	48.389	11.748	**9.283**
	12	34.320	39.021	9.145	**7.438**
n	16	30.282	28.914	6.562	**6.321**
	20	25.099	24.175	**5.220**	5.651
	24	21.335	18.981	3.737	**3.655**
	2	34.195	34.781	8.415	**5.616**
m	3	30.182	29.158	8.412	**6.687**
	4	30.666	31.319	**5.394**	5.723
	5	26.978	32.326	6.908	**6.852**
	2	26.096	23.748	5.419	**3.735**
F	3	31.249	34.395	7.820	**5.879**
	4	34.171	37.545	**8.608**	9.795
	2	30.034	31.615	6.820	**5.739**
s	3	31.453	32.196	8.183	**6.970**
	4	30.029	31.877	6.844	**6.700**
Average		30.505	31.896	7.282	**6.386**

Table 9
The average ARPD values on the large-scale benchmark set.

Parameter	scale	Mean ARPD			
		NEH	DNEH	NR_a	KBNEH
	100	7.816	8.547	5.729	**1.796**
n	200	6.084	6.619	4.993	**1.222**
	300	5.249	5.004	1.984	**0.510**
	5	7.629	7.032	5.805	**1.009**
m	10	5.746	6.228	3.960	**1.323**
	20	5.776	6.911	2.941	**1.196**
	4	6.513	6.266	3.286	**1.176**
F	6	6.054	6.812	4.383	**1.035**
	8	6.583	7.093	5.037	**1.316**
	30	7.340	7.836	3.555	**1.323**
s	40	6.014	6.690	4.088	**1.178**
	50	5.796	5.644	5.063	**1.026**
Average		6.383	6.724	4.235	**1.176**

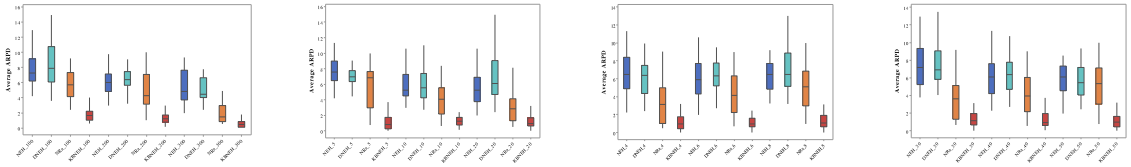
machines, factories, and products. The average ARPD values of KBNEH are superior to the average ARPD values of NEH, and DNEH. When the number of jobs is 20, the number of machines is 4, and the number of factories is 4, NR_a obtains the smallest average ARPD values among NEH, DNEH, and KBNEH. According to the results tested on the large-scale benchmark set, the performance of KBNEH is better than the compared constructive heuristic on all the combinations. It demonstrates that the performance of KBNEH is better than the compared constructive heuristic on all the combinations.

Besides, the box plots with 95% confidence are utilized to further analyze the performance of the proposed KBNEH. Figs. 13 and 14 show the box plots of the ARPD values on the two different benchmark sets. Box plot, which is often applied in data analysis, is a statistical chart to show the discrete distribution of data. The middle line in the box plot is the median of the data (Q_2), which represents the average level of the sample data. The upper and lower limits of the box plot are the upper quartile (Q_3) and the lower quartile (Q_1) of the data respectively. In addition, the difference between Q_3 and Q_1 is the quartile difference (IQR). The degree of fluctuation of the data is reflected by the IQR. Compared with other algorithms, the larger the value of IQR, the greater the fluctuation of the data and the weaker the stability of the algorithm. Therefore, the stability of the algorithm is analyzed intuitively. According to Fig. 13(a), the ARPD values obtained by the algorithms set are divided into five groups. When the number of jobs is 8, the Q_1, Q_2, and Q_3 of KBNEH are lower than that of NEH, DNEH, and NR_a. Furthermore, the IQR of KBNEH is 12.326, the IQR of NEH is 25.079, the IQR of DNEH is 30.645, and the



(a). Jobs (b). Machines (c). Factories (d). Products

Fig. 13. Box plot of the compared algorithms on small-scale benchmark set.



(a). Jobs (b). Machines (c). Factories (d). Products

Fig. 14. Box plot of the compared algorithms on large-scale benchmark set.

IQR of NR_a is 17.027. The results demonstrate that KBNEH has the least volatility than the compared algorithms. In combination with Fig. 14(a), when the number of jobs increases from 8 to 300, the IQR of KBNEH is decreased from 12.326 to 0.768. In other words, when the number of jobs increases gradually, the IQR of KBNEH shows a downward trend on the whole. Similarly, the values of Q_1, Q_2, Q_3 and IQR of KBNEH are smaller than that of the compared algorithms according to the box plots in Fig. 13(b) to (d) and Fig. 14(b) to (d). Based on the data comparison and analysis, the conclusion is conducted that KBNEH generates better results than NEH, DNEH, and NR_a, which demonstrates that the stability of KBNEH is better than NEH, DNEH, and NR_a. Hence, the experimental results and analysis show that KBNEH is effective in constructing solutions with high quality.

6.4. Effectiveness analysis of KWWO

In this section, the proposed KWWO algorithm is compared against four other state-of-the-art to evaluate the effectiveness of the algorithm for the DABFSP with the total tardiness criterion. The compared algorithms including TDIWO [55], ILS [11], EDE [6], and HEDFOA [25]. The mentioned metaheuristics are competitive algorithms, representing outstanding performance against different scheduling problems. The algorithms are reimplemented carefully and the values of adjustable parameters for the compared algorithms are set according to the original literature. It is worth mentioning that the algorithm for solving DABFSP is scarce. This paper extends the comparison algorithm by introducing the assembly stage. The computational experiments are executed based on the 900 small-scale instances and 810 large-scale instances. Furthermore, the Wilcoxon-test and Friedman-test [56], which are the parametric methods of non-parametric testing to show the differences of the experimental results are significant or not, are applied to make comparisons between the results obtained by KWWO and the results obtained by well-established algorithms.

6.4.1. Experimental results and analysis for small-scale instances

In the 900 small-scale instances, all the algorithms perform 5 times independently and the average ARPD values are obtained. The results, grouped by each combination of products (s), and factories (F) are summarized in Table 10 due to the reasons of space. Each cell contains the average of the 100 instances per value of products (s) $*$ factories (F). Meanwhile, the experimental results are visualized to make the comparison results clearer. Note that, there are 300 instances in each group, and the average ARPD value is taken for every 5 instances. The scatter diagrams, which show the ARPD values grouped by the products (s) are given in Fig. 15. Each scatter diagram has 90 instances.

The results of the Wilcoxon-test are shown in Table 11 to evaluate the significance level of all the compared algorithms. Note that, R_+, $R_=$, and R_- indicate that the results of KWWO are better than, similar than, and worse than its competitor on the corresponding quantity of problems, respectively. The value of α is set to 0.1 and 0.05, which represent the 90% confidence interval and the 95% confidence interval, respectively. If P-value is less than the value of α, the corresponding row is filled with yes, otherwise the current row is filled with no. The yes means that there are significant differences between KWWO and the comparison algorithms from the result of the experiment. Furthermore, the Friedman-test, which is a statistical test of the homogeneity of multiple related samples, is applied to investigate the statistical validity of the experimental results. According to the results, the lower the mean rank value of an algorithm, the better the performance of the algorithm. The results of the Friedman-test on the small-scale instances are shown in Table 12 in terms of products and factories. The Bonferroni–Dunn method [57] is considered as the post-hoc test to calibrate the significance level of the compared algorithms. The critical differences at 95% ($\alpha = 0.05$) and 90% ($\alpha = 0.1$) confidence intervals are utilized as the test thresholds. The results of the Bonferroni–Dunn experiment under the different circumstances of products and factories are shown in Figs. 16 and 17.

According to the experiments, the results and discussion are shown as follows.

(1) As seen from Table 10, KWWO obtains the smallest average ARPD values than the other four compared algorithms. The ARPD value of KWWO is 1.0096, which is 4 times smaller than TDIWO, 2 times smaller than ILS, 1 time smaller than EDE and HEDFOA. In each of the combinations of $s * F$, KWWO achieves the smallest ARPD values among all the compared algorithms.
(2) From Fig. 15, most of the red dots are located at the bottom of the graph and are lower than the dots of other colors in each scatter diagram. It shows that the ARPD

Table 10
The average ARPD values on the small-scale benchmark instances.

$s * F$	Algorithm				
	TDIWO	ILS	EDE	HEDFOA	KWWO
2 * 2	4.1664	1.8102	1.4585	1.2291	**0.8798**
2 * 3	4.0026	1.7427	1.3208	1.1281	**0.9038**
2 * 4	3.9867	1.7317	1.2611	1.0760	**0.9349**
3 * 2	5.8734	3.3221	2.8629	3.4274	**1.8965**
3 * 3	4.1725	2.0302	1.6609	2.0164	**1.3978**
3 * 4	3.8169	1.6765	1.3742	1.0574	**0.6980**
4 * 2	5.3464	2.5938	2.3241	2.1051	**1.2756**
4 * 3	4.2294	1.5963	1.4008	1.3926	**0.5752**
4 * 4	3.6449	1.7154	1.2807	1.1371	**0.5249**
Average	4.3599	2.0243	1.6604	1.6188	**1.0096**

values obtained by KWWO for most of the instances are smaller than those of other algorithms. As the size of the product increases from 2 to 4, the overall comparison performance summarized by Fig. 15(b) and (c) is almost unchanged. Most specifically, according to Fig. 15 with Table 11, KWWO is superior to TDIWO on 177 instances, ILS on 146 instances, EDE on 135 instances, and HEDFOA on 125 instances. Meanwhile, the value of the P-value listed in Table 11 is less than 0.05 and 0.1 in every row, which means there are significant differences between KWWO and the comparison algorithms from the experimental results. In other words, for the majority of benchmark instances, KWWO finds a solution that is better than those of other algorithms within a certain range.

(3) As seen from Table 12, the values of the mean rank obtained by KWWO are smaller than those of other algorithms in each different configuration of products (s) and factories (F), which demonstrates that the performance of KWWO is better than the compared algorithms. The P-value in each row is less than 0.05. It indicates that the difference between all the algorithms is statistically significant. The critical difference of the small-scale benchmark instances at the 95% ($C.D_{\alpha=0.05}$) and 90% ($C.D_{\alpha=0.1}$) are set to 0.7211 and 0.6472.

(4) In the Bonferroni–Dunn experiment, the solid line and dotted line represent 95% and 90% critical differences, which are calculated in Table 12. the mean rank of each algorithm is compared with the solid line and dotted line. When the mean rank of the algorithm is less than the threshold, there is a significant difference between the performance of algorithms. On the contrary, there is no significant difference. According to Fig. 16(a), the mean rank of KWWO is under the two lines and the mean rank of the compared algorithms is over the two lines. The result indicates that the performance of KWWO has significant differences with TDIWO, ILS, EDE, and HEDFOA when the number of products is 2. The conclusion is not changed when the number of products is 3 and 4. However, Fig. 17(b) and (c) show that there is no significant difference between KWWO and HEDFOA in terms of the number of factories is 3 and 4.

6.4.2. Experiments results and analysis for large-scale instances

The performance of the proposed KWWO is compared with four meta-heuristic algorithms on the 810 large-scale instances. All the algorithms perform 5 times independently and the average ARPD values are obtained. Due to space constraints, the statistical results, grouped by each combination of products (s), and factories (F), are shown in Table 13. Each cell contains the average of the 90 instances per value of products (s) $*$ factories (F). The visualization of the experimental results group by the products (s) is given in Fig. 18. Therefore, there are 270 instances in each group, and the average ARPD value is taken for every 10 instances. Meanwhile, the experimental results of the Wilcoxon-test and Friedman-test are listed in Tables 13 and 14. The results of the Bonferroni–Dunn experiment under the different circumstances of products and factories are shown in Figs. 19 and 20.

Following the results from Tables 13 to 15, and Figs. 18 to 21, the concrete analyses have been listed as follows.

(1) According to the average ARPD values, the ARPD value of KWWO is smaller than that of the other four compared algorithms. In other words, the performance of KWWO outperforms the other algorithms in terms of the overall level of 810 large-scale instances.

(2) From scatter diagrams in Fig. 18, most of the red dots, represented KWWO, are located at the bottom of the graph. Meanwhile, the red dots are lower than the dots of other colors. Based on the results of Table 14, KWWO is superior to TDIWO on 79 instances, ILS on 61 instances, EDE on 65 instances, and HEDFOA on 71 instances. In addition, The P-value in each row is less than 0.05, which indicates that there is a significant difference between KWWO and the corresponding comparison algorithm in the experimental results when the confidence intervals are set to 95% and 90%.

(3) As seen from Table 15, KWWO achieves the lowest values of mean rank in different configurations of products (s) and factories (F). The P-value in each row is less than 0.05 on large-scale instances. It indicates that the performance of KWWO is better than all the compared algorithms on the large-scale instances. From Fig. 19, the mean rank of KWWO is lower than the 95% and 90% confidence intervals which indicates that the difference between all the algorithms has statistically significant according to the results summarized based on the products. Besides, the comparison results are almost unchanged according to the Bonferroni–Dunn experiment in Fig. 20 when the results are considered based on the factories. The above findings suggest that the proposed KWWO performs better than the compared algorithms.

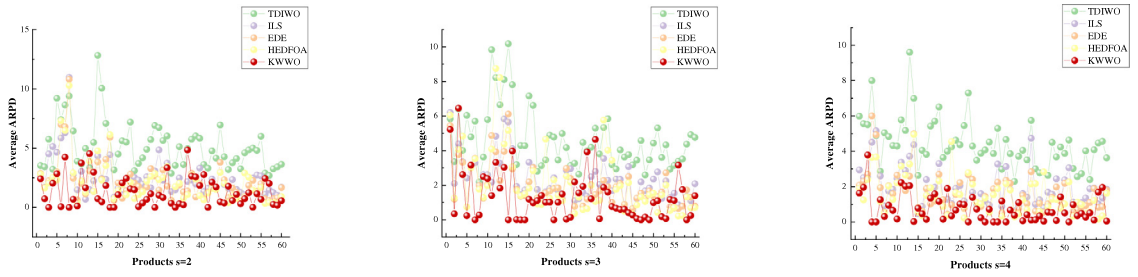
To further investigate the performance of KWWO in solving the considered problem $DF|blocking|\sum TTD_j$, the interval plots of all the algorithms under the different numbers of products (s) and products (F) with 95% confidence intervals are given in Figs. 21 and 22. In the interval plot, the interval represents the maximum and minimum values of data, such as the interval of TDIWO in Fig. 21(a) is (4.60825, 5.63886). The lines in the figures connect the mean value of the data. The performance of the algorithm, including the stability and obtaining the optimal solution, is analyzed through the interval and mean value. As seen eloquently from Fig. 21(a), KWWO obtains the minimum ARPD values when the number of products is 4. On the one hand, the interval of KWWO extends from 0.988535 to 1.63029, which is smaller than the interval lower limit of other compared algorithms. The result states that the performance of KWWO is better than other algorithms in obtaining excellent solutions. Meanwhile, the experimental results are proved to be statistically significant. On the other hand, the span of the interval of TDIWO, ILS, EDE, and HEDFOA is $5.63886 - 4.60825 = 1.031$, $2.97399 - 2.11764 = 0.856$, $2.65675 - 1.74438 = 0.9123$, $2.59152 - 1.65459 = 0.937$, and $1.63029 - 0.988535 = 0.642$. The smaller the span of the confidence interval, the smaller the data variation. The span of the confidence interval of KWWO is smaller than the data of the other compared algorithms, the analysis result is obtained that the stability of KWWO is better than the stability

Table 11
The Wilcoxon-test on the small-scale benchmark instances.

Control algorithm	Compared algorithm	R_+	R_-	$R_=$	P-value	$\alpha = 0.05$	$\alpha = 0.1$
KWWO	TDIWO	177	3	0	3.89E−31	Yes	Yes
	ILS	146	34	0	1.32E−17	Yes	Yes
	EDE	135	45	0	3.82E−10	Yes	Yes
	HEDFOA	125	55	0	9.89E−09	Yes	Yes

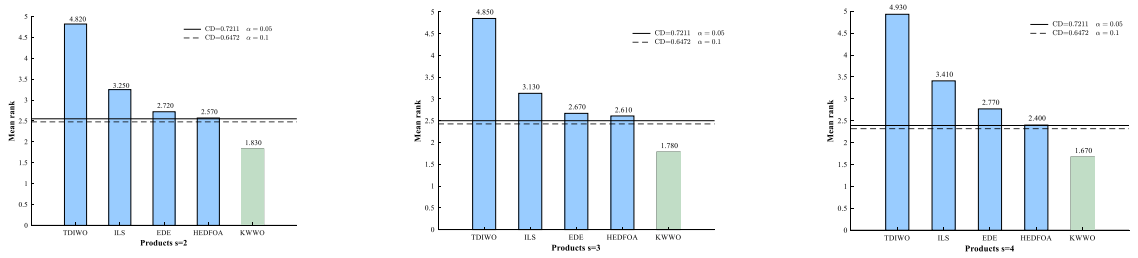
Table 12
The Friedman-test on the small-scale benchmark instances.

Index		Mean rank					P-value	$C.D_{\alpha=0.05}$	$C.D_{\alpha=0.1}$
		TDIWO	ILS	EDE	HEDFOA	KWWO			
F	2	4.72	3.32	2.68	2.58	1.70	4.68E−25	0.7211	0.6472
	3	4.93	3.23	2.68	2.30	1.85	1.29E−28		
	4	4.98	3.55	2.78	1.95	1.73	3.16E−35		
s	2	4.82	3.25	2.72	2.57	1.83	1.23E−26		
	3	4.85	3.13	2.67	2.61	1.78	8.57E−29		
	4	4.93	3.41	2.77	2.40	1.67	3.54E−32		



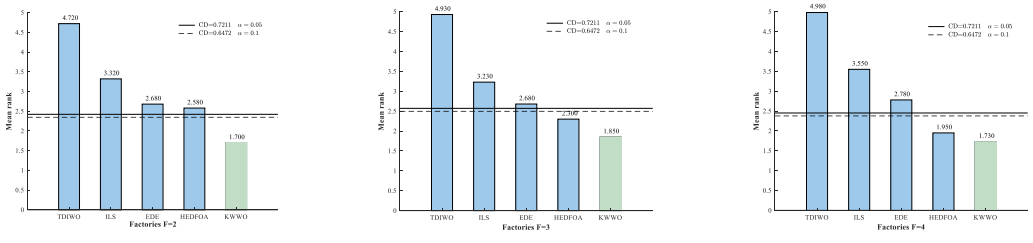
(a). Products s=2 (b). Products s=3 (c). Products s=4

Fig. 15. The scatter diagrams of the ARPD values on small-scale.



(a). Products s=2 (b). Products s=3 (c). Products s=4

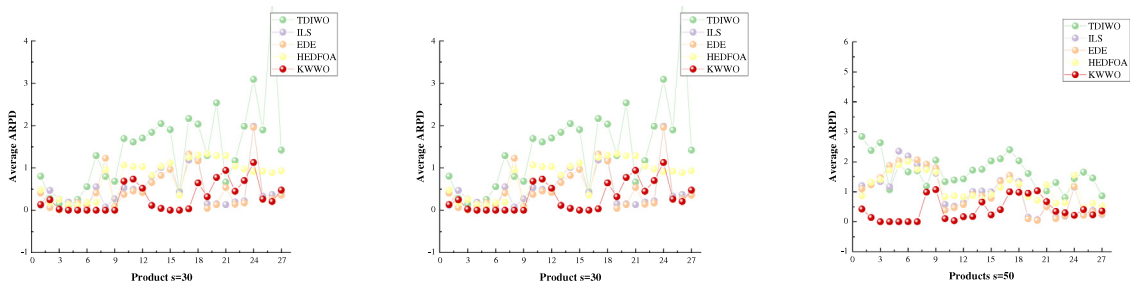
Fig. 16. The results of the Bonferroni–Dunn experiment according to the production (s).



(a). Factories F=2 (b). Factories F=3 (c). Factories F=4

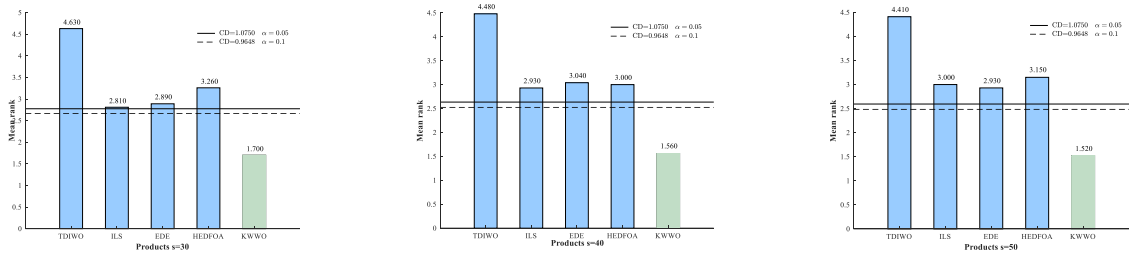
Fig. 17. The results of the Bonferroni–Dunn experiment according to the factories (F).

of the other algorithms. Furthermore, From Fig. 21(a) and (d), the span of the confidence interval of each algorithm increases along with the increasing scale of the number of products. The comparison results are unchanged in most situations in Fig. 22,



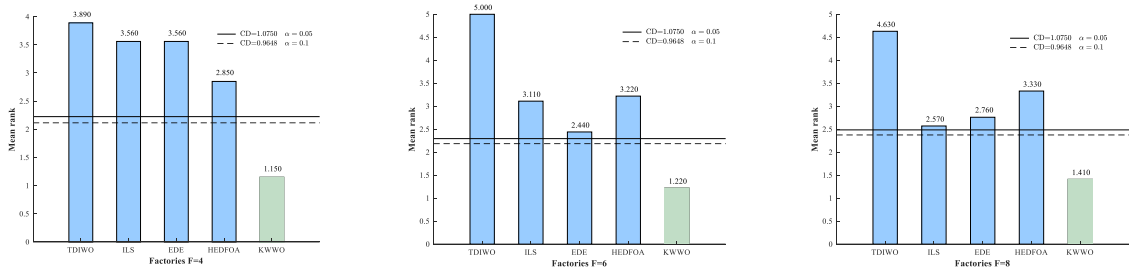
(a). Production s=30 (b). Production s=40 (c). Production s=50

Fig. 18. The scatter diagrams of the ARPD values on large-scale.



(a). Products s=30 (b). Products s=40 (c). Products s=50

Fig. 19. The results of the Bonferroni–Dunn experiment according to the production (s).



(a). Factories F=4 (b). Factories F=6 (c). Factories F=8

Fig. 20. The results of the Bonferroni–Dunn experiment according to the factories (F).

Table 13
The average ARPD values on the large-scale benchmark instances.

$s * F$	Algorithm				
	TDIWO	ILS	EDE	HEDFOA	KWWO
30 * 4	0.5485	0.2286	0.2830	0.2803	**0.0457**
30 * 6	1.7155	0.8034	0.8033	0.9248	**0.3093**
30 * 8	2.1108	0.4375	0.7808	0.7932	**0.4502**
40 * 4	1.9866	1.5647	1.5521	1.4102	**0.4575**
40 * 6	1.8523	0.9156	0.8799	1.0070	**0.2175**
40 * 8	1.3578	0.4022	0.6989	0.8755	**0.7034**
50 * 4	1.9385	1.6488	1.7253	1.5950	**0.2914**
50 * 6	1.7985	0.9912	0.9045	0.9883	**0.4084**
50 * 8	1.2447	0.3473	0.5106	0.6445	**0.4740**
Average	1.6170	0.8155	0.9043	0.9465	**0.3731**

except when the number of factories is 8, the stability of ILs is slightly better than that of KWWO. Therefore, KWWO is a winner in the statistical sense.

On the basis of the experiments and comparisons above, it is concluded that the proposed KWWO is an efficient and effective algorithm in solving DABFSP. The reasons are analyzed as follows.

(1) The constructive heuristic named KNBEH is proposed by exploring the characteristics of the DABFSP. The initialization of the population based on KBNEH and NR_a heuristic provides excellent initial solutions with high quality and diversity.

(2) The problem-specific knowledge is utilized to design the neighborhood search in KWWO. The population-based metaheuristic, which belongs to an iteration approach, has a requirement to take time to obtain more promising solutions. Therefore, the KWWO with problem-specific knowledge easily searches for most of the solutions during the available time once the termination criterion is given.

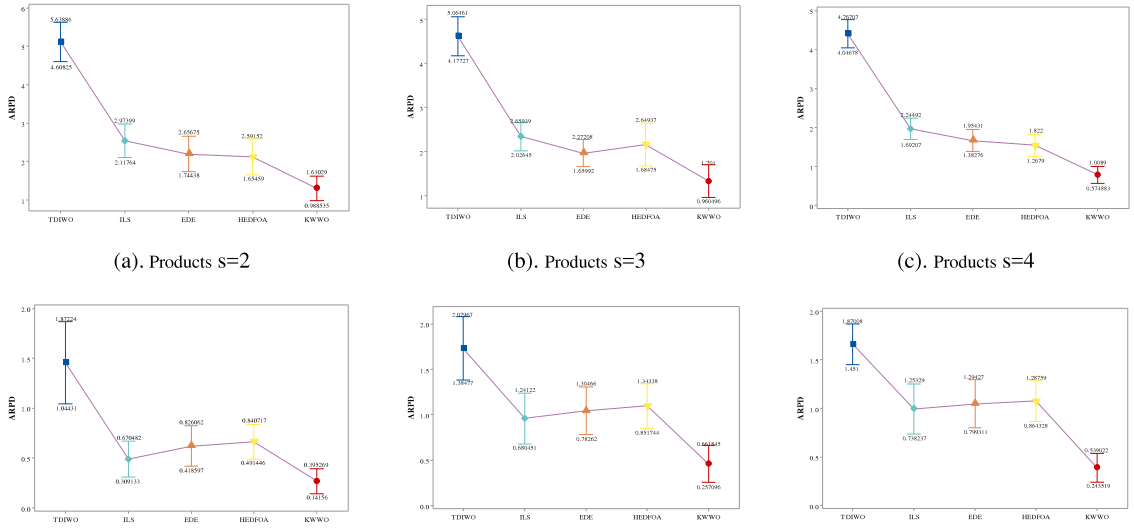
(3) To enhance the performance of the algorithm in solving the DABFSP with discrete characteristics, the operators are embedded to effectively ensure the search capacity. Such as the propagation operation is modified by the destruction–reconstruction method to search for promising individuals, the breaking operation is assisted by the local reinforcement strategy to improve the local search ability, and the refraction operation with path-relinking mechanism is introduced to keep the vitality of searching.

Table 14
The Wilcoxon-test on the large-scale benchmark instances.

Control algorithm	Compared algorithm	R_+	R_-	$R_=$	P-value	$\alpha = 0.05$	$\alpha = 0.1$
KWWO	TDIWO	79	2	0	1.41E−14	Yes	Yes
	ILS	61	20	0	4.27E−07	Yes	Yes
	EDE	65	16	0	5.53E−10	Yes	Yes
	HEDFOA	71	10	0	1.32E−10	Yes	Yes

Table 15
The Friedman-test on the large-scale benchmark instances.

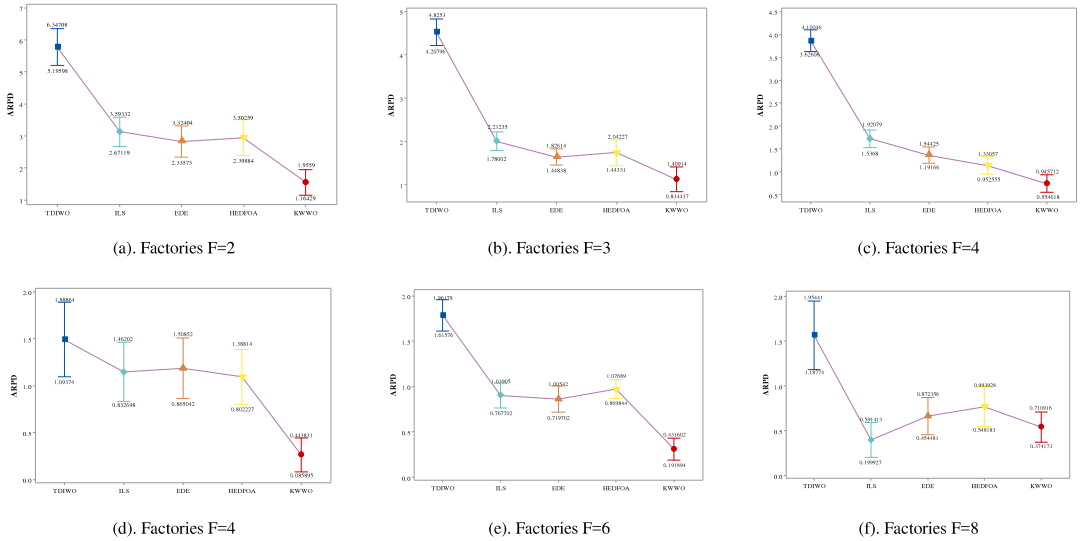
Index		Mean rank					P-value	$C.D_{\alpha=0.05}$	$C.D_{\alpha=0.1}$
		TDIWO	ILS	EDE	HEDFOA	KWWO			
F	4	3.89	3.56	3.56	2.85	1.15	1.10E−10	1.0750	0.9648
	6	5.00	3.11	2.44	3.22	1.22	9.09E−17		
	8	4.63	2.57	2.76	3.33	1.41	3.85E−09		
s	30	4.63	2.81	2.59	3.26	1.70	4.13E−10		
	40	4.48	2.93	3.04	3.00	1.56	2.12E−09		
	50	4.41	3.00	2.93	3.15	1.52	3.29E−09		



(a). Products s=2 (b). Products s=3 (c). Products s=4

(d). Products s=30 (e). Products s=40 (f). Products s=50

Fig. 21. The interval plots of the algorithms under the different numbers of products (*s*).



(a). Factories F=2 (b). Factories F=3 (c). Factories F=4

(d). Factories F=4 (e). Factories F=6 (f). Factories F=8

Fig. 22. The interval plots of the algorithms under the different numbers of factories (*F*).

7. Conclusion and future work

This paper investigates the DABFSP with the minimization criterion of the total tardiness. An effective KWWO is designed to solve the considered problem. Numerical experimental results show that the proposed algorithm has better performance than the well-established algorithms in terms of solution quality due to the usage of problem-specific knowledge. Therefore, the proposed KWWO is confirmed to be an effective method to improve factory efficiency by optimizing the goals determined by the decision maker. It should be pointed out that the algorithm may not be applied directly to other scheduling problems since the problem-specific operators are designed. However, the ideas based on the problem-specific knowledge are of certain guidelines for solving other complex problems in the modern manufacturing system.

In future research, we will take into account other realistic scopes and constraints in the DABFSP, such as transportation time, energy consumption, sequence-dependent setup time, etc. Additionally, develop an effective multi-objective algorithm for decision makers by analyzing the properties in the DABFSP with multiple optimization criteria. It is also important to apply the algorithm to real industrial problems.

CRediT authorship contribution statement

Fuqing Zhao: Supervision, Project administration, Idea, Conceptualization, Funding acquisition, Methodology. **Dongqu Shao:** Experiments of the algorithms, Writing. **Ling Wang:** Conceptualization. **Tianpeng Xu:** Validation. **Ningning Zhu:** Review and editing. **Jonrinaldi:** Review and editing.

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.

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent

Informed consent was obtained from all individual participants included in the study.

References

[1] X. He, Q. Pan, L. Gao, L. Wang, P.N. Suganthan, A greedy cooperative co-evolutionaryalgorithm with problem-specific knowledge for multi-objective flowshop group scheduling problems, IEEE Trans. Evol. Comput. (2021) 1, http://dx.doi.org/10.1109/TEVC.2021.3115795.

[2] Y. Du, J. qing Li, C. Luo, L. lei Meng, A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations, Swarm Evol. Comput. 62 (2021) http://dx.doi.org/10.1016/j.swevo.2021.100861.

[3] W. Shao, Z. Shao, D. Pi, Modeling and multi-neighborhood iterated greedy algorithm for distributed hybrid flow shop scheduling problem, Knowl.-Based Syst. 194 (2020) http://dx.doi.org/10.1016/J.KNOSYS.2020.105527.

[4] J. Zheng, L. Wang, J. jing Wang, A cooperative coevolution algorithm for multi-objective fuzzy distributed hybrid flow shop, Knowl.-Based Syst. 194 (2020) http://dx.doi.org/10.1016/J.KNOSYS.2020.105536.

[5] I. Ribas, R. Companys, X. Tort-Martorell, Efficient heuristics for the parallel blocking flow shop scheduling problem, Expert Syst. Appl. 74 (2017) 41–54, http://dx.doi.org/10.1016/j.eswa.2017.01.006.

[6] F. Zhao, L. Zhao, L. Wang, H. Song, An ensemble discrete differential evolution for the distributed blocking flowshop scheduling with minimizing makespan criterion, Expert Syst. Appl. 160 (2020) http://dx.doi.org/10.1016/j.eswa.2020.113678.

[7] Q. Chen, Q. Pan, B. Zhang, J. Ding, S. Member, Algorithms in compact strip production, 16 (2019) 1933–1951.

[8] V. Riahi, M. Khorramizadeh, M.A.H. Newton, A. Sattar, Scatter search for mixed blocking flowshop scheduling, Expert Syst. Appl. 79 (2017) 20–32, http://dx.doi.org/10.1016/j.eswa.2017.02.027.

[9] Z. Shao, W. Shao, D. Pi, Effective constructive heuristic and iterated greedy algorithm for distributed mixed blocking permutation flow-shop scheduling problem, Knowl.-Based Syst. 221 (2021) http://dx.doi.org/10.1016/j.knosys.2021.106959.

[10] S. Niroomand, Hybrid artificial electric field algorithm for assembly line balancing problem with equipment model selection possibility, Knowl.-Based Syst. 219 (2021) http://dx.doi.org/10.1016/J.KNOSYS.2021.106905.

[11] S. Hatami, R. Ruiz, C. Andrés-Romano, The distributed assembly permutation flowshop scheduling problem, Int. J. Prod. Res. 51 (2013) 5292–5308, http://dx.doi.org/10.1080/00207543.2013.807955.

[12] W. Shao, D. Pi, Z. Shao, Local search methods for a distributed assembly no-idle flow shop scheduling problem, IEEE Syst. J. 13 (2019) 1945–1956, http://dx.doi.org/10.1109/JSYST.2018.2825337.

[13] D. Ferone, S. Hatami, E.M. González-Neira, A.A. Juan, P. Festa, A biased-randomized iterated local search for the distributed assembly permutation flow-shop problem, Int. Trans. Oper. Res. 27 (2020) 1368–1391, http://dx.doi.org/10.1111/itor.12719.

[14] F. Zhao, L. Zhang, J. Cao, J. Tang, A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem, Comput. Ind. Eng. 153 (2021) 107082, http://dx.doi.org/10.1016/j.cie.2020.107082.

[15] J.N. Shen, L. Wang, S.Y. Wang, A bi-population EDA for solving the no-idle permutation flow-shop scheduling problem with the total tardiness criterion, Knowl.-Based Syst. 74 (2015) 167–175, http://dx.doi.org/10.1016/J.KNOSYS.2014.11.016.

[16] Y. Fu, G. Tian, A.M. Fathollahi-Fard, A. Ahmadi, C. Zhang, Stochastic multi-objective modelling and optimization of an energy-conscious distributed permutation flow shop scheduling problem with the total tardiness constraint, J. Clean. Prod. 226 (2019) 515–525, http://dx.doi.org/10.1016/J.JCLEPRO.2019.04.046.

[17] A. Allahverdi, H. Aydilek, A. Aydilek, No-wait flowshop scheduling problem with two criteria; total tardiness and makespan, European J. Oper. Res. 269 (2018) 590–601, http://dx.doi.org/10.1016/J.EJOR.2017.11.070.

[18] S. Hasija, C. Rajendran, Scheduling in flowshops to minimize total tardiness of jobs, Int. J. Prod. Res. 42 (2004) 2289–2301, http://dx.doi.org/10.1080/00207540310001657595.

[19] Q.Q. Zheng, Y. Zhang, H.W. Tian, L.J. He, An effective hybrid meta-heuristic for flexible flow shop scheduling with limited buffers and step-deteriorating jobs, Eng. Appl. Artif. Intell. 106 (2021) http://dx.doi.org/10.1016/J.ENGAPPAI.2021.104503.

[20] A. Khare, S. Agrawal, Effective heuristics and metaheuristics to minimise total tardiness for the distributed permutation flowshop scheduling problem, Int. J. Prod. Res. (2020) 1–17, http://dx.doi.org/10.1080/00207543.2020.1837982.

[21] C. Koulamas, The proportionate flow shop total tardiness problem, European J. Oper. Res. (2020) http://dx.doi.org/10.1016/J.EJOR.2020.01.002.

[22] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G.R. Kan, Optimization and heuristic in deterministic sequencing and scheduling: a survey, Ann. Discrete Math. 5 (1979) 287–326.

[23] F. Zhao, X. He, L. Wang, A two-stage cooperative evolutionary algorithm with problem-specific knowledge for energy-efficient scheduling of no-wait flow-shop problem, IEEE Trans. Cybern. (2020) 1–13, http://dx.doi.org/10.1109/tcyb.2020.3025662.

[24] Q.K. Pan, L. Gao, L. Xin-Yu, J.M. Framinan, Effective constructive heuristics and meta-heuristics for the distributed assembly permutation flowshop scheduling problem, Appl. Soft Comput. 81 (2019) 105492, http://dx.doi.org/10.1016/j.asoc.2019.105492.

[25] Z. Shao, D. Pi, W. Shao, Hybrid enhanced discrete fruit fly optimization algorithm for scheduling blocking flow-shop in distributed environment, Expert Syst. Appl. 145 (2020) 113147, http://dx.doi.org/10.1016/j.eswa.2019.113147.
[26] L. Wang, Q.K. Pan, P.N. Suganthan, W.H. Wang, Y.M. Wang, A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems, Comput. Oper. Res. 37 (2010) 509–520, http://dx.doi.org/10.1016/j.cor.2008.12.004.
[27] Y.Y. Han, D. Gong, X. Sun, A discrete artificial bee colony algorithm incorporating differential evolution for the flow-shop scheduling problem with blocking, Eng. Optim. 47 (2015) 927–946, http://dx.doi.org/10.1080/0305215X.2014.928817.
[28] Z. Shao, D. Pi, W. Shao, Estimation of distribution algorithm with path relinking for the blocking flow-shop scheduling problem, Eng. Optim. 50 (2018) 894–916, http://dx.doi.org/10.1080/0305215X.2017.1353090.
[29] Z. Shao, D. Pi, W. Shao, P. Yuan, An efficient discrete invasive weed optimization for blocking flow-shop scheduling problem, Eng. Appl. Artif. Intell. 78 (2019) 124–141, http://dx.doi.org/10.1016/J.ENGAPPAI.2018.11.005.
[30] Z. Shao, D. Pi, W. Shao, Self-adaptive discrete invasive weed optimization for the blocking flow-shop scheduling problem to minimize total tardiness, Comput. Ind. Eng. 111 (2017) 331–351, http://dx.doi.org/10.1016/j.cie.2017.07.037.
[31] Z. Shao, D. Pi, W. Shao, A novel multi-objective discrete water wave optimization for solving multi-objective blocking flow-shop scheduling problem, Knowl.-Based Syst. 165 (2019) 110–131, http://dx.doi.org/10.1016/j.knosys.2018.11.021.
[32] G. Zhang, K. Xing, F. Cao, Discrete differential evolution algorithm for distributed blocking flowshop scheduling with makespan criterion, Eng. Appl. Artif. Intell. 76 (2018) 96–107, http://dx.doi.org/10.1016/J.ENGAPPAI.2018.09.005.
[33] S. Chen, Q.K. Pan, L. Gao, Production scheduling for blocking flowshop in distributed environment using effective heuristics and iterated greedy algorithm, Robot. Comput. Integr. Manuf. 71 (2021) http://dx.doi.org/10.1016/J.RCIM.2021.102155.
[34] I. Ribas, R. Companys, X. Tort-Martorell, An iterated greedy algorithm for solving the total tardiness parallel blocking flow shop scheduling problem, Expert Syst. Appl. 121 (2019) 347–361, http://dx.doi.org/10.1016/j.eswa.2018.12.039.
[35] I. Ribas, R. Companys, X. Tort-Martorell, An iterated greedy algorithm for the parallel blocking flow shop scheduling problem and sequence-dependent setup times, Expert Syst. Appl. 184 (2021) http://dx.doi.org/10.1016/J.ESWA.2021.115535.
[36] Z. Shao, W. Shao, D. Pi, Effective constructive heuristic and metaheuristic for the distributed assembly blocking flow-shop scheduling problem, Appl. Intell. 50 (2020) 4647–4669, http://dx.doi.org/10.1007/S10489-020-01809-X.
[37] X. Wu, X. Liu, N. Zhao, An improved differential evolution algorithm for solving a distributed assembly flexible job shop scheduling problem, Memet. Comput. (2018) http://dx.doi.org/10.1007/s12293-018-00278-7.
[38] Y.J. Zheng, Water wave optimization: A new nature-inspired metaheuristic, Comput. Oper. Res. 55 (2015) 1–11, http://dx.doi.org/10.1016/j.cor.2014.10.008.
[39] R. Medara, R.S. Singh, Amit, Energy-aware workflow task scheduling in clouds with virtual machine consolidation using discrete water wave optimization, Simul. Model. Pract. Theory 110 (2021) 102323, http://dx.doi.org/10.1016/j.simpat.2021.102323.
[40] Z. Yan, J. Zhang, J. Tang, Path planning for autonomous underwater vehicle based on an enhanced water wave optimization algorithm, Math. Comput. Simulation 181 (2021) 192–241, http://dx.doi.org/10.1016/j.matcom.2020.09.019.
[41] X.Q. Lu, H.F. Yan, Z.L. Su, M.X. Zhang, X.H. Yang, H.F. Ling, Metaheuristics for homogeneous and heterogeneous machine utilization planning under reliability-centered maintenance, Comput. Ind. Eng. 151 (2021) 106934, http://dx.doi.org/10.1016/j.cie.2020.106934.
[42] X. Yun, X. Feng, X. Lyu, S. Wang, B. Liu, A novel water wave optimization based memetic algorithm for flow-shop scheduling, in: 2016 IEEE Congr. Evol. Comput., 2016, pp. 1971–1976, http://dx.doi.org/10.1109/CEC.2016.7744029.
[43] F. Zhao, H. Liu, Y. Zhang, W. Ma, C. Zhang, A discrete water wave optimization algorithm for no-wait flow shop scheduling problem, Expert Syst. Appl. 91 (2018) 347–363, http://dx.doi.org/10.1016/j.eswa.2017.09.028.
[44] F. Zhao, L. Zhang, Y. Zhang, W. Ma, C. Zhang, H. Song, A hybrid discrete water wave optimization algorithm for the no-idle flowshop scheduling problem with total tardiness criterion, Expert Syst. Appl. 146 (2020) http://dx.doi.org/10.1016/j.eswa.2019.113166.
[45] Z. Shao, D. Pi, W. Shao, A novel discrete water wave optimization algorithm for blocking flow-shop scheduling problem with sequence-dependent setup times, Swarm Evol. Comput. 40 (2018) 53–75, http://dx.doi.org/10.1016/j.swevo.2017.12.005.
[46] F. Zhao, L. Zhang, H. Liu, Y. Zhang, W. Ma, C. Zhang, H. Song, An improved water wave optimization algorithm with the single wave mechanism for the no-wait flow-shop scheduling problem, Eng. Optim. 51 (2019) 1727–1742, http://dx.doi.org/10.1080/0305215X.2018.1542693.
[47] J. Deng, L. Wang, S.Y. Wang, X.L. Zheng, A competitive memetic algorithm for the distributed two-stage assembly flow-shop scheduling problem, Int. J. Prod. Res. 54 (2016) 3561–3577, http://dx.doi.org/10.1080/00207543.2015.1084063.
[48] F. Zhao, J. Zhao, L. Wang, J. Cao, J. Tang, A hierarchical knowledge guided backtracking search algorithm with self-learning strategy, Eng. Appl. Artif. Intell. 102 (2021) 104268, http://dx.doi.org/10.1016/j.engappai.2021.104268.
[49] Y.-J. Zheng, X.-Q. Lu, Y.-C. Du, Y. Xue, W.-G. Sheng, Water wave optimization for combinatorial optimization: Design strategies and applications, Appl. Soft Comput. 83 (2019) http://dx.doi.org/10.1016/j.asoc.2019.105611.
[50] X. Xu, J. Li, M. Zhou, X. Yu, Precedence-constrained colored traveling salesman problem: An augmented variable neighborhood search approach, IEEE Trans. Cybern. (2021) 1–12, http://dx.doi.org/10.1109/TCYB.2021.3070143.
[51] F. Glover, Tabu search and adaptive memory programming — Advances, applications and challenges, 1997, pp. 1–75, http://dx.doi.org/10.1007/978-1-4615-4102-8_1.
[52] J.R. Smith, C. Larson, Statistical approaches in surface finishing. Part 3. Design-of-experiments, Trans. Inst. Met. Finish. 97 (2019) 289–294, http://dx.doi.org/10.1080/00202967.2019.1673530.
[53] T.F. Jaeger, Categorical data analysis: Away from ANOVAs (transformation or not) and towards logit mixed models, J. Mem. Lang. 59 (2008) 434–446, http://dx.doi.org/10.1016/j.jml.2007.11.007.
[54] V. Fernandez-Viagas, J.M. Framinan, NEH-based heuristics for the permutation flowshop scheduling problem to minimise total tardiness, Comput. Oper. Res. 60 (2015) 27–36, http://dx.doi.org/10.1016/j.cor.2015.02.002.
[55] H. Sang, Q. Pan, J. Li, P. Wang, Y. Han, K. Gao, Effective invasive weed optimization algorithms for distributed assembly permutation flowshop problem with total flowtime criterion, 2019, http://dx.doi.org/10.1016/j.swevo.2018.12.001.
[56] S. Garcia, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization, J. Heuristics 15 (2009) 617–644, http://dx.doi.org/10.1007/s10732-008-9080-4.
[57] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.