Applied Intelligence
https://doi.org/10.1007/s10489-022-03236-6

An ensemble discrete water wave optimization algorithm for the blocking flow-shop scheduling problem with makespan criterion

Fuqing Zhao[1] · Dongqu Shao[1] · Tianpeng Xu[1] · Ningning Zhu[1] · Jonrinaldi[2]

Accepted: 11 January 2022

Abstract
Production scheduling plays a pivotal role in smart factories due to the development of intelligent manufacturing. As a typical scheduling problem, the blocking flow-shop scheduling problem (BFSP) has attracted enormous attention from researchers. In this paper, an ensemble discrete water wave optimization algorithm (EDWWO) is proposed with the criterion to minimize the makespan. In the proposed algorithm, a constructive heuristic is presented to suit the needs of initial solutions quality. The constructive heuristic is based on a new dispatching rule combined with the well-known NEH heuristic. The algorithmic characteristics are explored and effective technologies, such as data-driven mechanism in the propagation phase, a block-shifting operator based on the framework of the variable neighborhood search in the breaking phase, and perturbation strategy, are employed to improve the performance of the algorithm. The effectiveness of operators and parameters in EDWWO are analyzed and calibrated based on the design of experiments. To evaluate the algorithmic performance, the well-known benchmark problem is adopted for comparison with five other state-of-the-art algorithms. Meanwhile, the statistical validity of the results is investigated by introducing the Friedman-test and Wilcoxon-test. The statistical results demonstrate the effectiveness of EDWWO for solving the BFSP.

Keywords Production scheduling · Blocking flow-shop scheduling problem · Water wave optimization algorithm · Makespan · Data-driven mechanism

1 Introduction

Under the background of economic globalization, intelligent manufacturing, which brings tremendous benefits for enterprises, has become the focus of current and future industrial development in smart factories [1, 2]. Production scheduling is the problem of allocating limited production resources to optimize one or more objectives determined by the decision maker [3, 4]. Thus, production scheduling in the manufacturing system is particularly important for meeting the various production requirements like shorter product delivery time and shorter completion time [5]. As a typical production scheduling problem, the flow-shop scheduling problem (FSP) has attracted enormous attention in manufacturing systems over the past decade, such as steelmaking-continuous casting processes, surface mounting of integrated circuit boards, and automobile painting production lines [6]. Therefore, Academia and industry are devoted to the study of effective scheduling technology to improve efficiency and enhance the core competitiveness of manufacturing enterprises [7].

In the FSP, n jobs are processed on m machines in the flow-shop production environment. Regulations are made so that: (1) each job has the same processing path, which is unable to be changed; (2) each job is processed on one machine only

✉ Fuqing Zhao
Fzhao2000@hotmail.com

Dongqu Shao
sdongqu@163.com

Tianpeng Xu
xutp@lut.edu.cn

Ningning Zhu
307516638@qq.com

Jonrinaldi
jonrinaldi@eng.unand.ac.id

[1] School of Computer and Communication, Lanzhou University of Technology, Lanzhou 730050, China

[2] Department of Industrial Engineering, Universitas Andalas, Padang 25163, Indonesia

Published online: 18 March 2022

once, and the process is not allowed to be interrupted; and (3) a job is not permitted to be processed on different machines at the same time [8]. The scheduling objective is to determine the processing sequence of the jobs to satisfy different requirements, such as minimizing the total tardiness time and minimizing the makespan. Specifically, the FSP is divided into the permutation flow-shop scheduling problem (PFSP) [9], blocking flow-shop scheduling (BFSP) [10], no-wait flow-shop scheduling (NWFSP) [11], and no-idle flow-shop scheduling [12] (NIFSP) based on the different constraints.

As a branch of the FSP, BFSP is commonly found in modern manufacturing and production systems, such as metallic cells of the manufacturing systems [13], the iron and steel industry [14], chemicals and pharmaceuticals [15], and waste treatment [16]. The BFSP is different from the other FSP models, which have been investigated widely by researchers [17]. In the BFSP, there is no buffer between machines. It is not allowed for the processing jobs to wait in the production system for the following operations [18]. The job has to be blocked on the current machine until the next machine is available for processing. Therefore, the completion time of jobs tends to be increased by the blocking constraint. In other words, especially in large-scale manufacturing systems, the difficulty of scheduling in the blocking environment is sharply increased because of the possible blockage. The FSP has been proven to be an NP-hard problem when there are more than two machines in the production system [19]. It is easily concluded that the BFSP is NP-hard well. Although the BFSP has been widely studied over the past years, literature on the BFSP is limited.

Methods to solve the BFSP increasingly emerge due to its complexity. However, the traditional mathematical methods, such as the branch and bound and enumeration method, are uncontrollable in terms of timeliness and performance. Therefore, heuristics and meta-heuristics are widely utilized in solving complex optimization problems. Regarding the representative heuristic, a Nawaz-Enscore-Ham (NEH) for solving the BFSP that minimizes the makespan was presented by Nawaz et al. [20]. A profile fitting (PF) heuristic algorithm was designed by McCormick et al. to solve the BFSP with the minimization cycle time [21].

In recent years, meta-heuristics have increasingly been proven to have satisfactory performance in solving a variety of optimization problems. The representative work about the BFSP is briefly listed in Table 1 since Wang et al. designed a novel hybrid discrete differential evolution algorithm (HDDE) for solving the BFSP with the makespan criterion [22]. Following this work, researchers have proposed many other effective methods to solve the considered problem, for example, a dynamic multi-swarm particle swarm optimizer (DMS-PSO) of Liang et al. [23], a hybrid modified global-best harmony search (hmgHS) algorithm of Wang et al. [24], a discrete artificial bee colony algorithm incorporating differential evolution (DE-ABC) of Han et al. [25], a modified fruit fly optimization (MFFO) algorithm of Han et al. [26], a hybrid combinatorial particle swarm optimization algorithm (HCPSO) of Eddaly et al. [27], and an estimation of distribution algorithm (P-EDA) and a discrete invasive weed optimization (DIWO) of Shao et al. [28]. Additionally, researchers also focused on solving the BFSP with other objectives. The multi-objective discrete invasive weed optimization (MODIWO) and multi- objective discrete water wave optimization (MODWWO) algorithms were proposed by Shao et al. for simultaneously considering the makespan and total flow time [13, 29]. For the same problem, Abu et al. presented a modified harmony search algorithm with neighboring heuristics methods (MHSNH) [30]. Afterward, Zhao et al. proposed a discrete gravitational search algorithm (DSGA) for the BFSP with the total flow time criterion [18]. Shao et al. proposed self-adaptive discrete invasive weed optimization (SaDIWO) for the total tardiness criterion [31].

With the deepening of research, several researchers have considered the BFSP with more complex constraints. Shao et al. considered two heuristics, as well as a discrete water wave optimization (DWWO) algorithm for the makespan criterion, and Han et al. proposed a discrete evolutionary multi-objective optimization (DEMO) algorithm for two objectives (i.e., makespan and total energy consumption) in the BFSP with sequence-dependent setup times [32, 33]. Ribas et al. investigated the distributed blocking flow-shop scheduling problem (DBFSP) with a mathematical model, and several constructive heuristics and meta-heuristics were presented to tackle the considered problem [34]. Following this work, Ribas et al. proposed an iterated greedy algorithm to solve the DBFSP with the total tardiness criterion [35]. Shao et al. proposed a hybrid enhanced discrete fruit fly optimization algorithm with an effective constructive heuristic and Zhao et al. proposed an ensemble discrete differential evolution (EDE) algorithm to solve the DBFSP with makespan criterion [36]. On this basis, Shao et al. studied a distributed fuzzy blocking flow-shop scheduling problem (DFBFSP) in which the processing time is uncertain. To address the problem with the fuzzy makespan criterion, the constructive heuristics named INEH and DPFNEH, and two iterated greedy methods were presented [37]. A constructive heuristic, as well as an iterated local search (ILS), was proposed by Shao et al. to solve the distributed assembly blocking flow-shop scheduling problem (DABFSP) [8]. Shao et al. considered an extension of distributed permutation flow-shop scheduling problem (DPFSP) with the blocking constraint. Meanwhile, an efficient iterated greedy algorithm with the proposed NEH_P heuristic was presented to solve the distributed mixed permutation blocking flow-shop scheduling problem (DMBPFSP) [38]. Recently, Ribas et al. investigated the sequence-dependent setup time in the parallel BFSPs [39, 40].

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

Problem	Objectives	Methods
BFSP	C_{max}	hybrid discrete differential evolution (HDDE) algorithm [22]
		dynamic multi-swarm particle swarm optimizer (DMS-PSO) [23]
		hybrid modified global-best harmony search (hmgHS) algorithm [24]
		discrete artificial bee colony algorithm incorporating differential evolution (DE-ABC) [25]
		modified fruit fly optimisation (MFFO) algorithm [26]
		hybrid combinatorial particle swarm optimization algorithm (HCPSO) [27]
		Estimation of distribution algorithm (P-EDA) [28]
		discrete invasive weed optimization (DIWO)
	C_{max}, $\sum C_j$	multi-objective discrete invasive weed optimization (MODIWO) algorithm [29]
		multi-objective discrete water wave optimization (MODWWO) algorithm [13]
		modified harmony search algorithm with neighboring heuristics methods (MHSNH) [30]
	$\sum T_j$	self-adaptive discrete invasive weed optimization (SaDIWO) [31]
	$\sum C_j$	discrete gravitational search algorithm (DSGA) [18]
BFSP- SDST	C_{max}	Two heuristics as well as discrete water wave optimization (DWWO) algorithm [32]
	C_{max}, TEC	discrete evolutionary multi-objective optimization (DEMO) algorithm [33]
DBFSP	C_{max}	effective constructive heuristic and hybrid enhanced discrete fruit fly optimization algorithm (HEDFOA) [36]
		ensemble discrete differential evolution (EDE) algorithm [41]
	$\sum T_j$	iterated greedy algorithm [35]
DBFSP- SDST	C_{max}	iterated greedy algorithms [40]
	$\sum T_j$	Iterated greedy algorithm (IGA) [35]
DABFSP	C_{max}	a constructive heuristic and iterated local search (ILS) [8]
DFBFSP	C_{max}	INEH, DPFNEH, and two iterated greedy (IG) methods [37]

From the short review above, it is seen that makespan is one of the most significant performance measures in the customer-oriented competitive market [42]. The minimization of the makespan enables manufacturing industries in saving processing costs and reduces machine utilization. According to the literature [43], the BFSP with the makespan criterion is denoted as $F_m \mid blocking \mid C_{max}$, where blocking means the model of the blocking flow-shop problem. Based on the previous work, the priority rule for assigning the job in addressing the BFSP with the makespan criterion is relatively single. An effective constructive heuristic method is necessary to be adopted to obtain a desirable solution with high quality in the initial phase [38]. Moreover, neighborhood operations are another critical issue for designing a meta-heuristic algorithm [44]. The insert operator and swap operator are commonly utilized to generate the neighboring solution. The computational complexity when adopting different search methods is quite different. For example, the full insert operation for each job is adopted to be executed on a sequence with n jobs. Each job has n positions to insert. The computational complexity in the phase is $O(n^n)$. Thus, designing effective and flexible search methods is needed to save computational efforts and improve algorithmic performance.

The nature-inspired optimization algorithms, which are utilized to address complex optimization problems, have attracted increasing attention from researchers during the past decades [45]. The water wave optimization algorithm (WWO) is such a promising algorithm inspired by shallow wave theory, which was first introduced for continuous optimization problems [46]. 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 [47, 48]. Due to its robustness and effectiveness, it has attracted the attention of various researchers for application to tackle multitudinous problems, for example, workflow scheduling in the cloud [49], autonomous underwater vehicles [50], machine utilization [51], optimal reactive power dispatch [52], and so on.

In recent years, researchers have attempted to solve the FSP by adopting WWO. The operations in WWO were redesigned by Yun et al. [53], where an improved NEH algorithm and the memetic algorithm were utilized to solve the combined

arrangement FSP. A discrete WWO (DWWO) was proposed by Zhao et al. [54] to address the no-wait FSP based on the makespan criterion. Zhao et al. [55] proposed a hybrid discrete WWO (HWWO) for the no-idle FSP with total tardiness criterion. Additionally, Zhao et al. [56] proposed a cooperative WWO (CWWO) for the distributed assembly no-idle FSP (DSNIFSP). Shao et al. [32] proposed a discrete WWO (DWWO) for the blocking FSP with sequence-dependent setup times (BFSP-SDST). Then, a single wave mechanism-the single WWO (SWWO) was presented by Shao et al. [57] for the no-wait FSP with the goal of the minimum completion time. Furthermore, a novel multi-objective WWO (MOD-WWO) was introduced by Shao et al. to minimize both makespan and total flow time in BFSP [13]. Therefore, motivated by the extensive applicability of WWO, an ensemble discrete WWO (EDWWO) is proposed to solve the BFSP with the criterion to minimize the maximum completion time. The main contributions of EDWWO are summarized as follows:

1. A new constructive heuristic method, called SNEH, considering both the processing time and the front delay time is proposed to generate the initial solution.
2. Three basic operations of WWO are redesigned to solve the considered problem. Specifically, the data-driven mechanism is introduced in the propagation to control the wavelength and balance the exploration and exploitation of the proposed algorithm. In the breaking phase, the optimal block-shifting operator is proposed in the framework of variable neighborhood search to enhance the search near the current optimal solution. Ultimately, a perturbation strategy is adopted in the refraction operation phase to avoid the algorithm falling into the local optimum.

The structure of this paper is organized as follows: Section 2 presents the definition and mathematic model of the BFSP. Section 3 briefly introduces the basic WWO. The procedure of EDWWO is described in detail in Section 4. Then, the experimental results and detailed analysis are given in Section 5. Finally, the conclusions and future work are discussed in Section 6.

2 Formulation of the blocking flow-shop scheduling problem

2.1 Problem description

The BFSP was proposed by Lawler et al. [58]. Supposing there is a job set $J = \{J_j | (j = 1, 2, 3, \cdots, n)\}$ and a machine set $M = \{M_i | (i = 1, 2, 3, \cdots, m)\}$, then, n jobs have to be sequenced through m machines, which constitute a flow-shop production system, according to the processing sequence $\pi = \{\pi(1), \pi(2), \pi(3), \cdots, \pi(n)\}$. Here, $O_{j,i}$ from set $O_j = \{O_{j,i} | (i = 1, 2, 3, \cdots, m)\}$ represents the operation executed on the machine M_i with a processing time $P_{j,i}$. Note that, there are no intermediated buffers between machines. That is, the job J must be blocked on the current machine M_i until machine M_{i+1} is available for processing. A Gantt chart for the BFSP is displayed in Fig. 1.

2.2 Mathematical model of the blocking flow-shop scheduling problem

The notations utilized in this paper are presented below.

Indices

j	Index of the current jobs; $j \in \{1,2,\cdots,n\}$
i	Index of the current machines; $i \in \{1,2,\cdots,m\}$
k	Index of the current job position; $k \in \{1,2,\cdots,n\}$

Parameters

n	The total number of jobs
m	The total number of machines
π	The scheduling sequence, where $\pi = \{\pi(1),\pi(2),\pi(3),\cdots,\pi(n)\}$.

Decision Variable

$O_{j,i}$	The operation that job j executed on the machine i.
$x_{j,k}$	1 if job j is assigned to position k; 0 otherwise
$D_{k,i}$	the departure time of a job that occupies position k in the machine i.

The optimization objective is to minimize the C_{max}. A reasonable sequence of jobs ineeds to be formulated for optimizing the objective. The mathematical model of the BFSP is formalized as follows [30]. $D_{k,i}$ denotes the departure time of a job that occupied position k in machine i.

$$Minimize\ C_{max}\ subject\ to \tag{1}$$

$$\sum_{k=1}^{n} x_{j,k} = 1 \quad j = 1, 2, \ldots, n \tag{2}$$

$$\sum_{j=1}^{n} x_{j,k} = 1 \quad k = 1, 2, \ldots, n \tag{3}$$

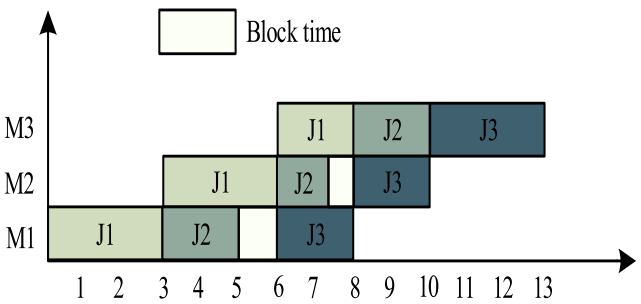


Fig. 1 A Gantt chart for the BFSP

$$D_{1,0} \geq 0 \tag{4}$$

$$D_{k,1} \geq D_{k-1,1} \quad k = 2, \ldots, n \tag{5}$$

$$D_{k,i} \geq D_{k,i-1} + \sum_{j=1}^{n} x_{j,k} \times P_{j,i} \quad \begin{matrix} k = 1, 2, \ldots, n \\ i = 1, 2, \ldots, m \end{matrix} \tag{6}$$

$$D_{k,i} \geq D_{k-1,i+1} \quad \begin{matrix} k = 2, \ldots, n \\ i = 1, 2, \ldots, m-1 \end{matrix} \tag{7}$$

$$C_{max} = D_{n,m} \tag{8}$$

where, $x_{j,\,k}$ denotes the decision variable. Each job has to be processed on a machine only once based on Eqs. (2) and (3). Besides, $D_{1,\,0}$ is a dummy variable, which is designed to show the start time of the first job on the first machine. Equations (4) and (5) define the departure time of the job j processed on the first machine. The relationship of the departure time of each job between the two machines is explained based on Eqs. (6) and (7). Finally, the objective function is confined as $C_{max} = D_{n,\,m}$, where $D_{n,\,m}$ represents the departure time of the final job in the sequence executed on the final machine.

2.3 Water wave optimization algorithm

In the traditional WWO, the three fundamental operations are implemented dynamically after initialization [55]. The flowchart of WWO is shown in Fig. 2.

Propagation phase: Each wave has a wavelength to control the search degree of the population. By moving the dimension of the parent wave, the propagation operation is used to generate a new wave.

Breaking phase: The core of the breaking operation is to perform further exploitation around the best wave. Specifically, k different dimensions are selected for generating a train of solitary waves.

Refraction phase: After each wave gets propagated, the height of the new wave is recalculated. Once the height of the wave decreases to zeros, the algorithm performs the refraction operation to regenerate a new wave.

2.4 Ensemble discrete water wave optimization algorithm for blocking flow-shop scheduling problem

The BFSP with discrete characteristics is unable to be optimized by the canonical WWO, which is presented to solve continuous optimization problems. Therefore, a new algorithm named EDWWO is proposed to solve the BFSP with the makespan criterion. In this section, the distinctive structure of WWO, and the technologies in EDWWO are illustrated. EDWWO consists of an effective initialization scheme, a

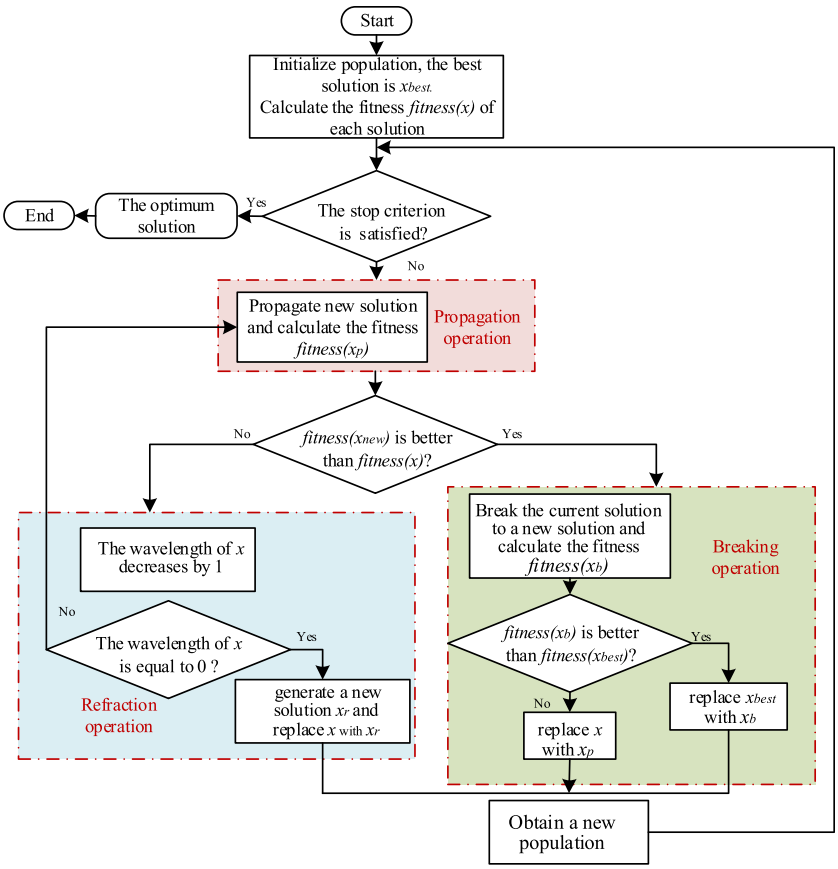


Fig. 2 The flowchart of WWO

propagation operation with a data-driven mechanism, a breaking operation based on optimal block-shifting, and a refraction operation with a perturbation strategy.

2.5 Constructive heuristic algorithm

The constructive heuristic algorithm is utilized to construct a desirable solution quickly with a very limited computational time [34]. Inspired by NEH, which is one of the most effective heuristic methods to solve the FSP, SNEH is designed as the initializal method. The rule for assigning the job is determined by the total processing time of each job in the traditional NEH. The front delay time of a job plays an essential role in influencing the makespan. Supposing an example with three jobs and two machines. The processing time of the jobs $\{J_1, J_2J_3\}$is set as $\begin{bmatrix} 2 & 3 \\ 1 & 5 \\ 4 & 4 \end{bmatrix}$. The two job sequences are set as π_1= {1,2,3}, and π_2= {3,1,2}. The Gantt charts of π_1,and π_2 are shown in Fig. 3, where the $C_{max}(\pi_1)$=14, $C_{max}(\pi_2)$=16. The results demonstrate that the job sequence with different front delay times results in a different makespan to a certain extent.

Accordingly, a new constructive heuristic algorithm (SNEH), which includes a new assignment rule and a reinsertion operation, is proposed to obtain a desirable solution. In SNEH, a bicriteria index S is adopted for assigning the jobs, where the processing time and the front delay time are considered as follows:

$$Dt_j = \sum_{i=1}^{m}(m-i) \times p_{j,i} \tag{9}$$

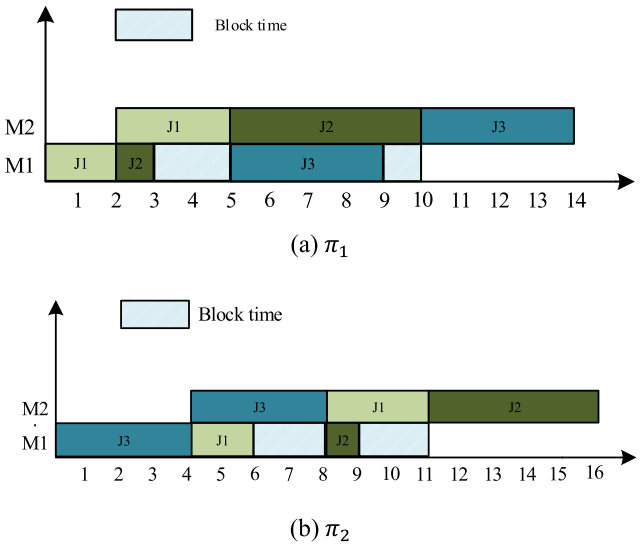


Fig. 3 The Gantt chart of π_1, π_2

$$Pt_j = \sum_{i=1}^{m} p_{j,i} \tag{10}$$

$$S(j) = 2 \times \sigma \times Dt_j/(m-1) + (1-\sigma) \times Pt_j \tag{11}$$

where, where Dt_j and Pt_j denote the front delay time and the total processing time of job j, respectively. The weight parameter σ is determined based on the parameter test in Section 5. The bicriteria index $S(j)$ is calculated based on Eq. (11). Meanwhile, the jobs are sorted in ascending sequence and the job with minimum S is selected to be the first processing job.

All the remaining jobs are inserted in the suitable position with the smallest makespan one by one based on the NEH [59]. After a job is inserted into the sequence at the best position, a reinsertion operation is trigged on the previous or following job of the inserted job. The reinsertion operation improves the quality of the solution to a certain extent. The procedure of SNEH is given in Algorithm 1.

Algorithm 1 SNEH

Input: PT
Output: $Solution_o$ $Fitness_0$

```
Calculate S(j) for each job, j = 1,2,⋯,n;
Obtain the initial sequence θ = {θ(1), θ(2), ⋯, θ(n)} according to S;
π(1) = θ(1);
for k = 2:n do
    Obtain π by inserting the job θ(k) at position P_min resulting in the smallest C^k_max;
//reinsertion operation
    Remove the job R from the position P_min − 1 or P_min + 1 and insert R at the position resulting in the smallest C^k_max;
end for
Calculate the C_max of π;
Solution_o = π;
Fitness_0 = C_max;
return Solution_o, Fitness_0
```

2.6 Ensemble discrete water wave optimization algorithm

2.6.1 Population initialization

WWO is a population-based meta-heuristic algorithm., in which N_p identical initial solutions are generated in the population initialization phase. To generate an initial population with a high level of quality and diversity, the constructive heuristic SNEH proposed in Section 4.1 is employed to obtain a desirable solution. Then, the rest of the N_p-1 solution is generated by performing a two-stage neighborhood operator. The two operators are displayed in Fig. 4. The pseudocode of the population initialization is presented in Algorithm 2.

Algorithm 2 Population initialization

Input: N_p
Output: S, $Fitness$, S_{best}, $Fitness_{best}$

```
Obtain an initial solution π_1 based on Algorithm 1;
Calculate Fitness_{π_1};
S_1 = π_1, Fitness_1 = Fitness_{π_1};
for i = 2: N_p do
    ϵ=rand (0,1);
    if ϵ>0.5
        π_i = Insert − Swap(π_1);
    else
        π_i = Swap − Insert(π_1);
    end if
    Calculate the fitness of π_i;
    S_i = π_i, Fitness_i = Fitness_{π_i};
end for
Fitness_best = Min(Fitness);
return S, Fitness, S_best, Fitness_best
```

2.6.2 Data-driven propagation operation

In WWO, the search capability of WWO is impacted to a great extent by the propagation operation, which is designed as a global search action to balance the exploration and the exploitation. The wavelength λ is one of the attributes of each wave. The search range is proportional to the wavelength of each solution. The way to generate a new solution is determined by the propagation operation. Propagation is performed on each wave under the control of the wavelength. When a new solution is generated, wavelength λ is updated. The wavelength plays a matter role in the propagation behavior.

In the proposed EDWWO, a propagation operation based on a data-driven mechanism with a destruction-reorganization strategy is designed to improve the performance of the algorithm in finding the current optimal. The data-driven mechanism, which is utilized to control the update of the wavelength, is proposed based on the feedback of the comparison between the current fitness fit_i and the fitness fit_{i-1} is the last iteration. The new wavelength λ is defined as follows:

$$\lambda(i) = \begin{cases} \lambda_{max}, & fit_i > fit_{i-1} \\ \lambda_{min}, & fit_i < fit_{best} \\ \lambda_{max} - (\lambda_{max} - \lambda_{min}) \times \alpha^{\frac{fit_i - fit_{min}}{(fit_{max} - fit_{min}) * (1+\varepsilon)}}, & otherwise \end{cases} \tag{12}$$

where fit_i is the fitness of the current individual; fit_{i-1} is the fitness of the parent individual; fit_{best} means the fitness of the historically best individual; fit_{min} and fit_{max} are the individuals with the current optimal and worst value, respectively; α is the wavelength reduction coefficient, and the value is set to 1.0026. ε is introduced to prevent the denominator from being 0.

A diagram of the destruction-reorganization strategy is presented in Fig. 5. The number of destruction points is according to the destruction rate (D_s), which is calibrated by the method presented in Section 4.1, and the location of the destruction point is selected randomly. As shown in Fig. 5, the destruction point is J_2. Then, J_2 is removed from the initial sequence, and inserted in all the locations of the remaining sequence. The sequence with the smallest makespan is obtained as the final sequence. The procedure of the propagation operation based on a data-driven mechanism with the destruct-reorganization strategy is given in Algorithm 3.

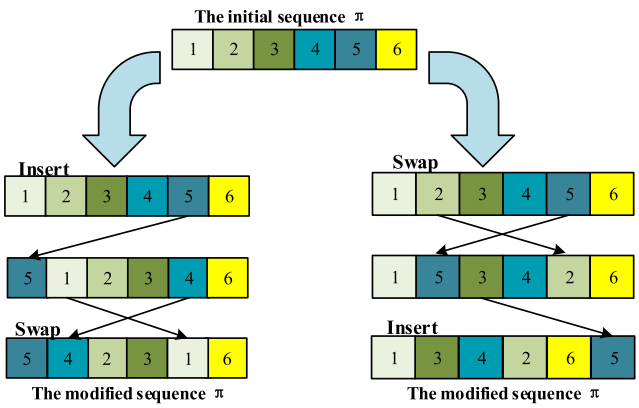


Fig. 4 The description of the two operations

Algorithm 3 Propagation

Input: $Solution_i$, $Fitness_i$ λ_i
Output: $Solution_P$ $Fitmess_P$

```
// destruction
Obtain partial sequence ϑ by selecting λ_i jobs from the Solution_i;
// reorganization
for k = 1: λ_i do
    Obtain π by inserting ϑ(k) at the position resulting in the smallest C^k_max;
end for
Calculate the C_max of π;
Solution_p = π;
Fitness_p = C_max;
return Solution_P, Fitmess_P
```

2.6.3 Breaking operation based on optimal block-shifting

In WWO, when a new current optimal is found, the breaking behavior is performed on the wave to split it into several isolated waves, and each wave moves a short distance from

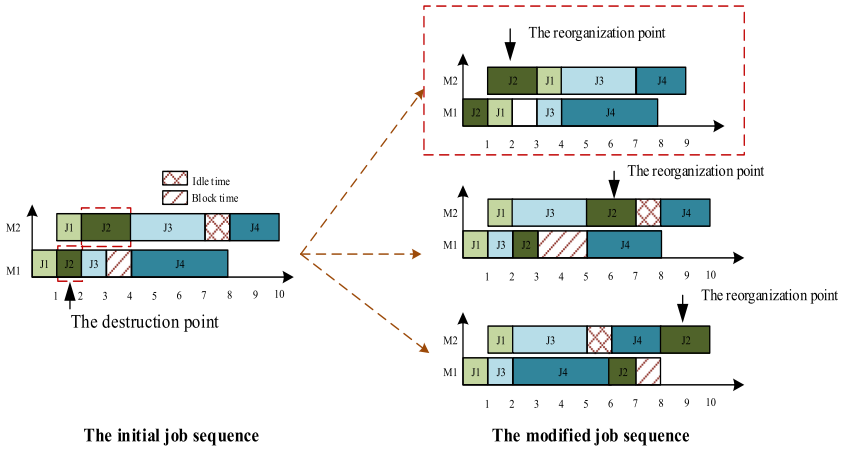


Fig. 5 The procedure of the destruct-reorganization strategy

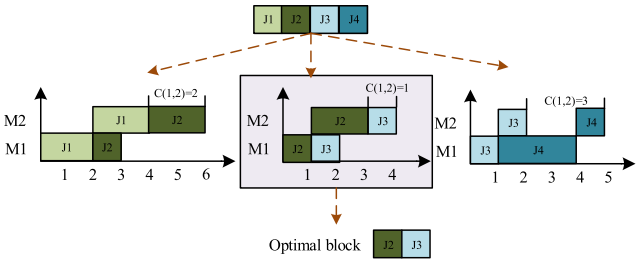


Fig. 6 The diagram of an optimal block

the new optimal wave in a random direction to strengthen the local search capacity of the WWO algorithm. However, the local search capability of the breaking operation is relatively weak. Thus, a breaking operation based on block-shifting with variable neighborhood search as a framework is proposed to further search around the current optimal to find a new solution.

The block-shifting operator is proposed to overcome the limitations of the insert or swap operator, which bring small variations to the current sequence especially in large-scale problems [60]. Inspired by the core of the block-shifting operator, the block-shifting operator is introduced in breaking operation to improve the intensive search. A diagram of an optimal block is given in Fig. 6. The process of the modified breaking operation is as follows. Firstly, according to the block size recommendations, which were provided by Ding et al. [61], the optimal current solution is determined to perform the block-shifting operator. Then, a series of insert operators and swap operators are executed between the current solution and the optimal block. The procedure of the breaking operation based on optimal block-shifting with the variable neighborhood search as a framework is described in Algorithm 4.

Algorithm 4 Breaking

Input: $Solution_i$, $Fitness_i$
Output: $Solution_b$ $Fitness_b$

```
Determine the optimal block γ;
Solution_0 = Solution_i, Fitness_0 = Fitness_i;
k_max = 2, k = 1;
while k ≤ k_max do
    switch k
        Case 1: Solution_1 = inset(γ, Solution_0);
        Case 2: Solution_1 = swap(γ, Solution_0);
    end switch
    if Fitness_1 < Fitness_0
        Fitness_0 = Fitness_1, Solution_0 = Solution_1;
        k = 1;
    else k = k + 1;
    end if
end while
Solution_b = Solution_0;
Fitness_b = Fitness_0;
return Solution_b, Fitness_b
```

2.6.4 Perturbation of refraction operation

The refraction operation in WWO is designed as a local search method as well as the breaking operation to avoid algorithm stagnation. The individual, which is unable to find a desirable individual after a few iterations, is removed from the current population. Then, a new individual is generated by perturbing the optimal individual. Consequently, a perturbation technology is introduced in the refractor operator to learn the search information stored in the excellent solutions. The procedure of the refractor operator is shown in Algorithm 5. In the proposed method, the individual with the best performance $Solution_{best}$ is selected as the guiding solution to help the current individual $Solution_i$ to generate a new worthful individual. To transform $Solution_i$ into $Solution_{best}$, the jobs and positions are recorded when the jobs of the two individuals in the corresponding positions are the same. Meanwhile, the recorded jobs are inserted in the corresponding positions of the new

individual $Solution_{new}$. The remaining jobs in $Solution_i$ are perturbed and inserted in the empty positions in S_{new}. An example is considered with two solutions, $Solution_{best}$= {1,2,3,4,5}, $Solution_i$= {5,4,3,2,1}. The $Solution_{new}$= {_, _, 3, _, _} is obtained, and the empty positions p= [1, 2, 4, 5]. Supposing a job sequence {4,1,2,5} is generated after performing a random perturbation method on the remaining jobs 1,2,4,5, then, according to the procedure of refraction, the jobs are inserted into the corresponding empty positions in $Solution_{new}$. As a result, the final solution {4,1,3,2,5} is obtained.

Algorithm 5 Refraction

Input: $Solution_i$, $Solution_{best}$
Output: $Solution_r$, $Fitmess_r$

```
ϵ = ∅, p = 1;
for k = 1:|Solution_i| do
    if S_i(k) = S_best(k)
        S_temp(k) = S_i(k);
    else ϵ(p) = S_i(k), p = p + 1;
    end if
end for
Obtain ϵ_new by perturbating ϵ, k = 1;
for P = 1:p do
    if S_temp(k) = ∅
        S_temp(k) = ϵ_new(P);
    else k = k + 1;
    end if
end for
Calculate the C_max of S_temp;
Solution_r = S_temp;
Fitmess_r = C_max;
return Solution_r, Fitmess_r
```

2.6.5 The framework of ensemble discrete water wave optimization algorithm

According to the discussions on each component of the EDWWO, the general framework of EDWWO is given in Algorithm 6. In EDWWO, the population with high quality and diversity is generated based on SNEH in the initial phase. A new priority rule that considers the front delay time and processing time is designed. Meanwhile, a reinsertion operator is utilized to improve the quality of the solution.

After the population is initialized, the propagation operation of EDWWO is executed on each solution to obtain an offering solution. The propagation operation is modified by the destruction-reorganization strategy based on a data-driven mechanism. To determine the search range in the next iteration, the wavelength is reinforcement controlled by introducing the data-driven strategy, which utilizes the reward of the comparison between the solution and offspring solution. The next search behavior is determined by the offspring solution. If the offspring solution is better than the current solution and the historical optimal solution, then the breaking operation is performed on the offspring solution; Otherwise, the wave height is reduced. Once the wave height is reduced to 0, the refraction operation is executed on the current solution. In the breaking operation phase, the optimal block-shifting operator, which replaces the simple single job-shifting, is utilized in the framework of variable neighborhood search to enhance the search near the current optimal solution. In the refraction phase, the substructure of the historical optimal solution is preserved to the maximum extent possible. The performance of EDWWO in achieving a desirable solution is improved by the coordination of the initialization, propagation, breaking, and refraction. A flowchart of EDWWO is described in Fig. 7.

2.7 The computational complexity of ensemble discrete water wave optimization algorithm

In this section, the computational complexity of EDWWO is analyzed in detail. Suppose that a problem is proposed with $n$jobs and m machines, and the population size of EDWWO is set to N_p. In the initial phase, the individuals are generated based on Algorithm 1. The computational complexity in this phase is $O(mn + n^2)$. In the propagation phase, the wavelength is generated based on the data-driven mechanism. The computational complexity of the search strategy is decided by the wavelength. Hence, the computational complexity of propagation operation is $O(N_p \times \lambda \times mn^2)$. The computational complexity in the breaking phase is decided by the selection of optimal block, and the framework of VNS. The computational complexity in this phase is $O(N_p(k \times m(n-1)^2)$. In the last phase, the computational complexity for the worst case is $O(N_p \times mn^2)$. Therefore, the computational complexity of EDWWO is calculated as follows:

$$\begin{aligned} &O(mn+n^2)+0(N_p\times)+O(N_p\times\lambda\times mn^2)\\ &\quad+O\left(N_pk\times m(n-1)^2\right)+O(mn+n^2)\\ &=O(N_p\times\lambda\times mn^2) \end{aligned}$$

2.8 The engineering application in smart factories

Due to the requirements of material temperature or other characteristics in some production processes, the production environment with blocking constraint appears in many practical process applications, such as the production of steel. Molten steel goes through a series of operations, such as ingot casting, demoulding, reheating, soaking,

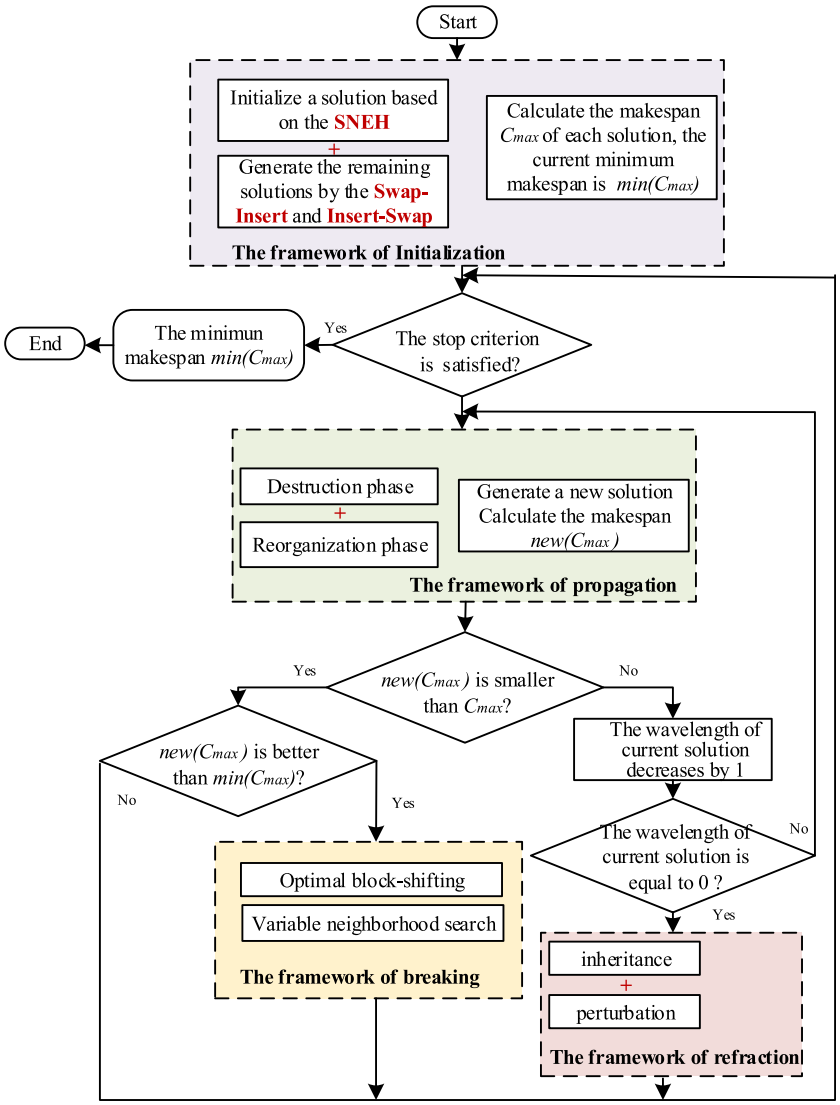


Fig. 7 The flowchart of EDWWO

and rolling. A series of processes are conducted at a certain temperature to ensure smooth production. Molten steel stays on the current machine to maintain temperature when there is no space buffer between machines. Therefore, the coordinated manufacturing process is modeled as the BFSP to meet the requirements of customers.

In this section, the proposed method is applied to solve the scheduling problem in the production environment of steel. Supposing there are 3 materials and 5 production processes. According to the requirements of the decision-maker, each material completes the production of steel in the shortest time. The processing time of materials in different production processes is listed in Table 2. When the materials are processed in the order of {1,2,3}, the makespan is C_{max}=388. However, the optimal sequence of processing materials is found by utilizing EDWWO to optimize the machining time greatly. The optimal sequence obtained by EDWWO is {3,1,2}, the makespan is C_{max}=372. Therefore, the proposed method is an effective technology for the production scheduling problem in smart factories.

Table 2 The processing time for different materials

materials	Production processes				
	ingot casting	demoulding	heating	soaking	rolling
1	83	3	89	58	56
2	15	11	49	31	20
3	54	79	16	66	58

Algorithm 6 EDWWO

Input: N_p, PT, λ_{max}, λ_{min}, H_{max}, σ
Output: $Solution_{best}$, $Fitness_{best}$

```
//initialization
Obtain Solution_1 and Fitness_1 based on Algorithm 1;
Initialize the population;
Record Solution_best, and Fitness_best;
while the stop criterion is not satisfied do
  for i = 1:N_p do
//propagation
    Obtain Solution_p and Fitness_p based on Algorithm 3;
    if Fitness_p < Fitness_i
      Fitness_p = Solution_p, Fitness_i = Fitness_p;
      if Fitness_p < Fitness_best
        Fitness_best = Solution_p , Fitness_best = Fitness_p;
//breaking
        Obtain Solution_b and Fitness_b based on Algorithm 3;
        if Fitness_b < Fitness_best
          Fitness_best = Solution_b , Fitness_best = Fitness_b;
        end if
        h(i) = h_max;
    else h(i) = h(i) − 1;
        if h(i) == 0
//refraction
          Obtain Solution_r and Fitness_r based on Algorithm 4;
          if Fitness_r < Fitness_best
            Fitness_best = Solution_r , Fitness_best = Fitness_r;
          else
            Fitness_i = Solution_r , Fitness_i = Fitness_r;
          end if
        end if
      end if
    Update λ(i) based on Eq. (12);
  end for
end while
return Solution_best, Fitness_best
```

3 Experiments and analysis

The performance of EDWWO in solving the considered problem is evaluated based on its performance on the well-known Taillard benchmark problem in this section. According to the description by Rossi et al. [62], the instance is generated with $n \in \{20,50,100,200,500\}$, and $m \in \{5,10,20\}$. The concrete combinations in the instance are given in Table 3. Each combination has ten different instances. Therefore, the benchmark problem consists of 120 instances, where there are 12 combinations of n and m.

Table 3 The combinations of n and m

n	20	50	100	200	500
m	{5,10,20}	{5,10,20}	{5,10,20}	{10,20}	{20}

To make a fair comparison for the performance of the algorithms in solving the BFSP, 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 of RAM in the Windows 10 operation system. Meanwhile, the EDWWO and compared algorithms are coded with Java. The MATLAB language is applied to perform the non-parametric tests, such as Friedman-test and Wilcoxon-test. The average relative percentage deviation (ARPD) [63] over the solutions found in the experiment is adopted as the performance measure.

$$ARPD = (1/runs) \times \sum_{i=1}^{runs} \left(C_{alg.} - C_{best}\right)/C_{best}\Big) \times 100 \quad (13)$$

where $C_{alg.}$ is the makespan obtained by the particular algorithm, C_{best} is the lowest makespan obtained in the experiment, and *runs* is the time to run the algorithm on each instance.

3.1 Parameters tuning

Design of experiments [64] is adopted to test the impact of the control parameters of EDWWO. The special control parameters in the EDWWO included population size (N_p), weight parameter (σ), and destruction rate (D_s). The wavelength λ is self-adapting based on the data-driven mechanism, thus, the wavelength does not need to be analyzed in this section. The levels of each control parameter are listed as follows: $N_p \in \{10,30,50\}$, $\sigma \in \{0.4,0.5,0.6\}$, and $D_s \in \{0.2,0.4,0.6\}$. The three parameters lead to a total of $3 \times 3 \times 3 = 27$ configurations. The implemented test set consists of 48 instances, where there are 12 combinations of n and m with each combination having 4 instances. Each configuration is tested five times with the same termination criterion, which is $t_{max} = 10 \times n \times m$ milliseconds.

The analysis of variance technique (ANOVA) [65] is utilized to analyze the effect of the parameters. The ANOVA results are given in Table 4. As shown, the F-ratio of the population size N_p is larger than σ and D_s, which indicates that the population size has the most significant effect on the performance of EDWWO. The main effects plot of the control parameters in EDWWO is presented in Fig. 8 to determine the value of the control parameters. From Fig. 8 (N_p), it is seen that EDWWO performs better at N_p=30 than N_p=10 and N_p=50. Fig. 8 (σ) shows that the performance of EDWWO is better at σ=0.6, and the results become worse as the value of σ increased. Additionally, Fig. 8 (D_s) indicates that the performance of EDWWO is better when D_s=0.6 than any of the other levels. Hence, the value of the control parameters in EDWWO is set as N_p=30, σ=0.6, and D_s=0.6.

Table 4 Results of ANOVA for parameters calibration of EDWWO

Source	Sum of squares	Degrees of freedom	Mean square	F-ratio	P-value
N_p	0.13142	2	0.06571	810.29	0
σ	0.00105	2	0.00053	6.5	0.0211
D_s	0.0004	2	0.0002	2.45	0.1475
$N_p * \sigma$	0.00478	4	0.00119	14.73	0.0009
$N_p * D_s$	0.00028	4	0.00007	0.87	0.5234
$\sigma * D_s$	0.00051	4	0.00013	1.56	0.2747
Error	0.0065	8	0.00008		
Total	0.13908	26			

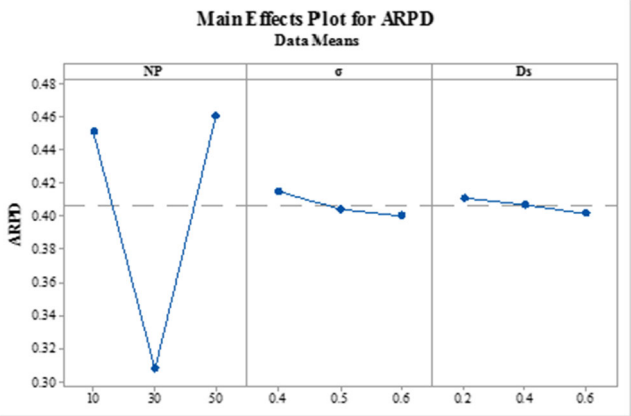


Fig. 8 Main effects plot of the control parameters in EDWWO

3.2 Heuristic algorithm analysis

To verify the performance of SNEH, three other heuristic algorithms, including NEH [66], DNEH [67], and TR [41], are adopted for comparison. All the algorithms are tested against 120 instances in the Taillard benchmark problem with the same termination criterion $t_{max} = 10 \times n \times m$ milliseconds. The value of ARPD is computed as the variant to evaluate the performance of the algorithms in obtaining the initial solution. The experimental results are given in Table 5. Meanwhile, the point plot for the compared initial methods is presented in Fig. 9. As shown in Table 5 and Fig. 9, the proposed SNEH is superior to DNEH and TR on all the scales of instances, but it is inferior to NEH on five scales, including 50×5, 100×5, 100×10, 200×10, and 500×20. Additionally, the probability plot with a 95% confidence interval for all the compared algorithms is shown in Fig. 10 to evaluate the compliance of data points to the fitted lines. Each algorithm in the plot has three lines, and the middle line is named fitted line. The data points are distributed on both sides of the lines. According to Fig. 10, the points of SNEH tightly distributed along a straight around the fitted line demonstrates that the stability and compactness of SNEH are better than the other compared algorithms.

3.3 Effectiveness analysis for each component

The dominant components in EDWWO are divided into four parts, including the new heuristic SNEH, the propagation phase with the data-driven mechanism, the breaking phase with the block-shifting operator, and the refraction phase with the perturbation operator. To analyze the effectiveness of each component of EDWWO, simulation experiments are designed based on the benchmark set of Taillard. In the simulation experiments, EDWWO is conducted to compare with the ED-S algorithm (EDWWO without the initial method SNEH), ED-P-DD algorithm (EDWWO without the data-driven mechanism in the propagation phase), ED-B-BS algorithm (EDWWO without the block-shifting operator in the breaking phase), and ED-R-P algorithm (EDWWO without the perturbation in the refraction phase). The Taillard benchmark set consists of 12 problems with ten instances for each combination of jobs and machines. The details of the combination are given in Table 3. All the algorithms performed five

Table 5 ARPD for the compared initial methods

n	m	ARPD			
		NEH	DNEH	TR	SNEH
20	5	3.0374	5.7270	4.7801	1.6409
	10	10.8608	14.7344	9.9695	3.0144
	20	8.6729	10.4052	7.7050	2.6465
50	5	4.2161	6.3035	8.2685	5.4045
	10	3.1498	7.5245	3.9550	1.5108
	20	6.6674	9.6494	9.5558	1.2662
100	5	0.0000	4.4530	2.8074	1.3804
	10	1.5032	3.8450	4.8100	2.0961
	20	3.9796	5.6185	9.4235	2.6467
200	10	0.0000	2.5843	6.7718	3.0183
	20	0.9280	3.1425	1.0647	0.5455
500	20	1.1658	1.8420	2.2663	1.8013

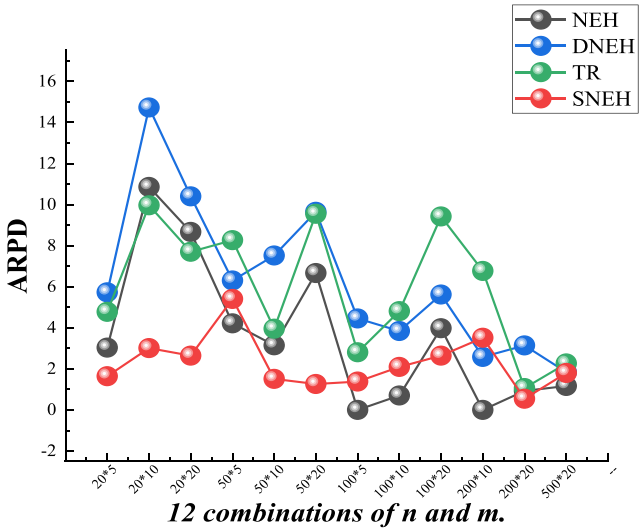


Fig. 9 Point plot for NEH, DNEH, TR, and SNEH

Table 6 The results for EDWWO and four designed algorithms

n	*m*	ED-S	ED-P-DD	ED-B-BS	ED-R-P	EDWWO
20	5	2.875	2.178	3.035	2.026	**0.344**
	10	2.248	2.617	2.446	2.449	**0.260**
	20	1.929	1.375	1.708	1.866	**0.224**
50	5	5.911	4.933	5.809	6.286	**1.432**
	10	4.402	4.779	4.345	4.590	**1.593**
	20	3.431	3.341	3.363	3.413	**0.983**
100	5	6.598	6.447	6.688	6.630	**0.401**
	10	5.028	5.110	5.055	5.239	**0.429**
	20	4.516	4.629	4.646	4.523	**0.415**
200	10	5.768	5.623	5.883	5.237	**1.246**
	20	5.583	5.999	5.699	5.859	**0.574**
500	20	5.866	7.398	5.945	5.945	**0.182**
Average		4.513	4.536	4.552	4.505	**0.673**

independent runs with the same termination criterion as $t_{max} = 10 \times n \times m$ milliseconds. The experimental results for EDWWO and four designed algorithms are given in Table 6. The optimal ARPD value for each instance is marked in black.

As seen from Table 6, the ARPD value obtained by EDWWO is the lower than the ARPD value of ED-S, ED-P-DD, ED-B-BS, and ED-R-P on each instance. Besides, ED-S obtains the highest ARPD value than the other compared algorithms when the number of jobs is 20 and 50, and the number of machines is 20. The number of the higher ARPD value obtained by ED-P-DD and ED-B-BS than the ARPD value obtained by other compared algorithms among all the instances is 4. Moreover, the ARPD value of ED-R-P is higher than the other compared algorithms in four instances, including 20×10, 50×5, 100×10, and 200×10.

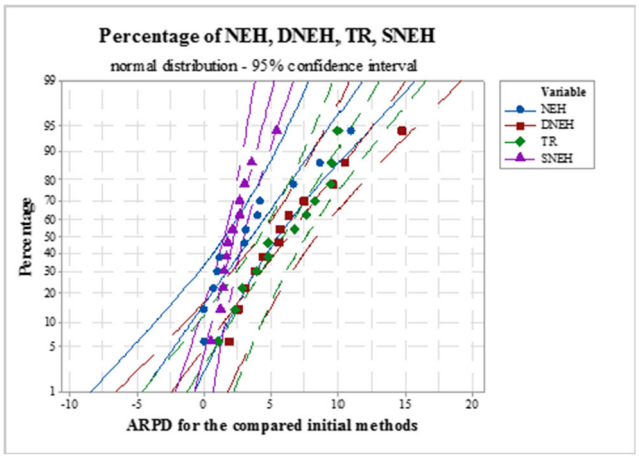
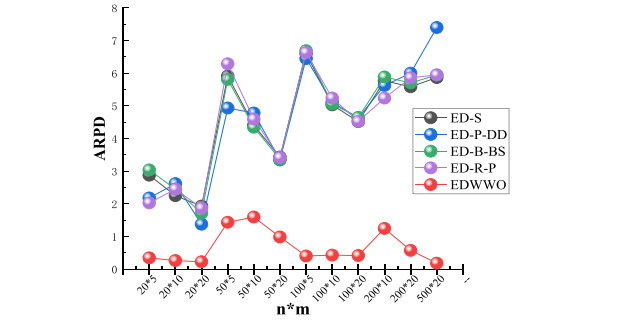


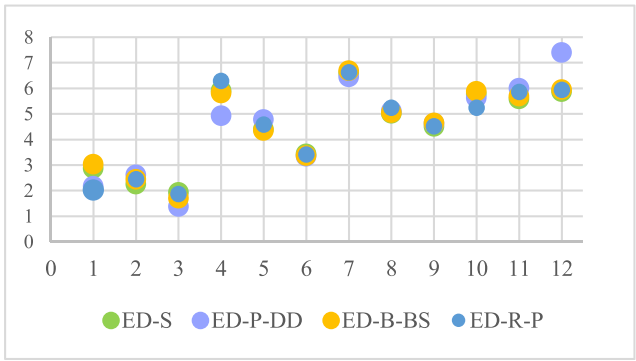
Fig. 10 Probability plot for NEH, DNEH, TR, and SNEH

Since the overall average ARPD value of ED-B-BS is higher than the overall average ARPD value of the compared algorithms, the breaking operation with the block-shifting operator has the most significant impact on EDWWO. The breaking operation plays a vital role in exploitation to find the optimum. Once the block-shifting operator with the framework of variable neighborhood search is removed from the breaking operation, the ability for ED-B-BS in local search is diminished. Additionally, the average ARPD value of ED-P-DD is lower than the average ARPD value of ED-B-BS but higher than the other three compared algorithms. The reason for this is that ED-B-BS is unable to acquire an effective balance between exploration and exploitation without the data-driven mechanism in the propagation operation. According to the results of the average ARPD value among ED-S, ED-P-DD, ED-B-BS, and ED-R-P, there is an significant difference between them.

Figure 11 presents scatter diagrams of the experimental results. The ARPD value among all the compared algorithms is shown in Fig. 11(a). Moreover, the results of ED-S, ED-P-DD, ED-B-BS, and ED-R-P are given in Fig. 11 (b) independently to show the comparison clearly. Furthermore, a box plot of ED-S, ED-P-DD, ED-B-BS, ED-R-P, and EDWWO is utilized to analyze the stability of the algorithms. As shown in Fig. 12, the stability of EDWWO is superior to the other algorithms. The middle line of ED-P-DD is higher than the middle line of ED-S, ED-B-BS, ED-R-P, and EDWWO. When the enhanced propagation is removed from EDWWO, the impact on the stability of EDWWO is clear. In summary, the experimental results confirm that each dominant component has a significant effect on improving the performance of the presented EDWWO algorithm.



(a) Scatter diagrams for the average ARPD value



(b) Scatter diagrams for the average ARPD value

Fig. 11 Scatter diagrams among the compared algorithms. (**a**) Scatter diagrams for the average ARPD value. (**b**) Scatter diagrams for the average ARPD value

3.4 Comparative experimental analysis

The proposed EDWWO algorithm is compared against five other metaheuristics to evaluate the effectiveness of the algorithm when solving the BFSP with the makespan criterion. The computational experiments are executed based on the well-known Taillard benchmark set, which is described clearly in Table 3. The compared algorithms are listed below:

- Hybrid iterated greedy (HIG) [68]
- Iterated greedy algorithm (IG2S) [69]
- Iterated greedy algorithm (IGA) [70]
- Iterated local search algorithm (ILS) [34]
- Scatter search algorithm (SS) [71]

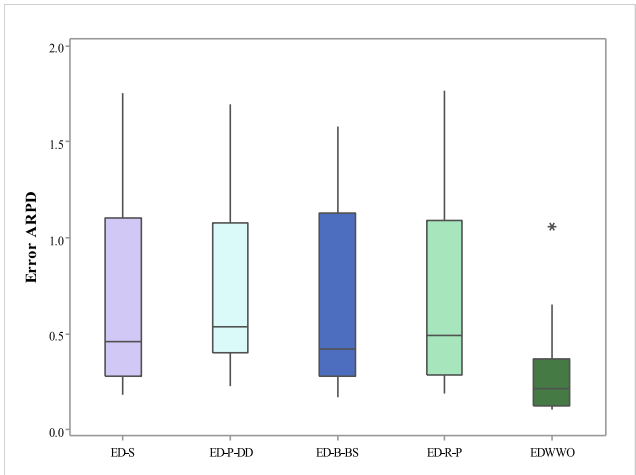


Fig. 12 Box plots among the compared algorithms

The mentioned metaheuristics are competitive algorithms, representing outstanding performance against different scheduling problems. The algorithms are reimplemented carefully, and the values of the adjustable parameters for the compared algorithms are set according to the original literature. All the algorithms performed five independent runs with the same termination criterion as $t_{max} = 10 \times n \times m$ milliseconds. Analogously, the ARPD is adopted to be the method to measure the performance of the results obtained by the algorithms for an unbiased assessment. The ARPD results of the 120 instances are arranged into 12 groups based on the combination of n and m. Each group has ten instances. The average ARPD value of each group is given in Table 7. The smallest value is indicated in bold.

(1) Visualization of the experimental results

In this section, the experimental results are visualized to make the comparison results clearer. Besides, the analyses are given in the end. Firstly, the scatter diagrams, which show the ARPD values of the 120 instances grouped by every 20 instances, are illustrated in Fig.13. Each line represents an algorithm, and it is noted that the line in the scatter diagram closer to the bottom means there is better performance of the algorithm compared to other algorithms. Moreover, the pie chart in Fig. 14 explains the proportion of the number of optimal solutions obtained by each algorithm based on the benchmark set.

Table 7 The results of the compared algorithm

n	m	HIG	IG2S	IGA	ILS	SS	EDWWO
20	5	1.911	1.106	1.120	1.068	2.768	**0.674**
	10	1.089	0.803	0.759	0.732	2.176	**0.462**
	20	0.851	0.642	0.643	0.674	1.728	**0.259**
50	5	3.172	2.464	2.395	2.469	5.319	**2.390**
	10	2.712	**1.767**	2.064	1.844	4.572	2.588
	20	2.611	1.827	2.154	1.966	4.018	**1.359**
100	5	1.210	1.104	1.095	1.106	3.523	**1.046**
	10	1.357	0.985	1.160	**0.910**	3.277	1.373
	20	2.095	1.440	2.083	1.874	3.434	**1.056**
200	10	1.562	0.557	0.798	**0.530**	1.824	0.731
	20	2.135	1.653	2.182	1.986	2.936	**0.636**
500	20	3.072	0.656	0.839	0.731	1.029	**0.651**
Average		1.981	1.250	1.441	1.324	3.050	**1.102**

Following the results from the experimental results from Table 7 and Figs. 13 and 14, the concrete analyses are listed as follows.

1. As seen from Table 7, EDWWO obtains smaller average ARPD values than the other five compared algorithms. The ARPD value of EDWWO is 1.102, which is 1.8 times smaller than HIG, 1.1 times smaller than IG2S, 1.4 times smaller than IGA, 1.2 times smaller than ILS, and 2.8 times smaller than SS. In each of the combinations of n × m, EDWWO achieves the smallest ARPD values among almost all the compared algorithms except for 50×10, 100×10, and 200×10. The results characterize that EDWWO is superior to the compared algorithms.
2. According to Fig. 13, the ARPD values obtained by the algorithm are represented by clarity. From Fig. 13(a), 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. This shows that the ARPD values obtained by EDWWO for most of the instances are smaller than those of the other algorithms. In combination with Figs. 13 and 14, EDWWO obtains a total of 51% (61) optimal solutions, IG2S obtains a total of 26% (31) optimal solutions, ILS obtains a total of 19% (23) optimal solutions, IGA obtains a total of 3% (4) optimal solutions, and HIG obtains 1% (1) optimal solution respectively. Thus, EDWWO is an effective and competitive algorithm in solving the BFSP with the criterion to minimize the makespan.

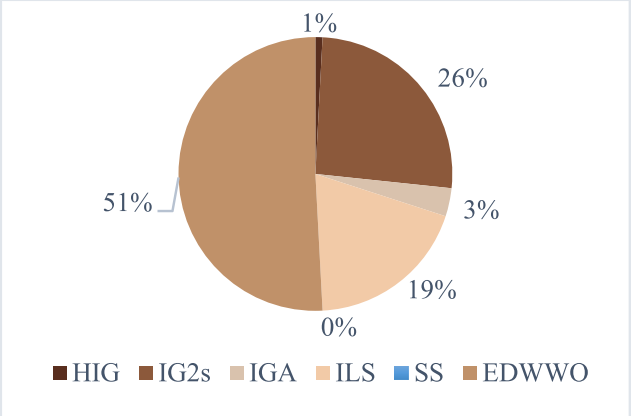
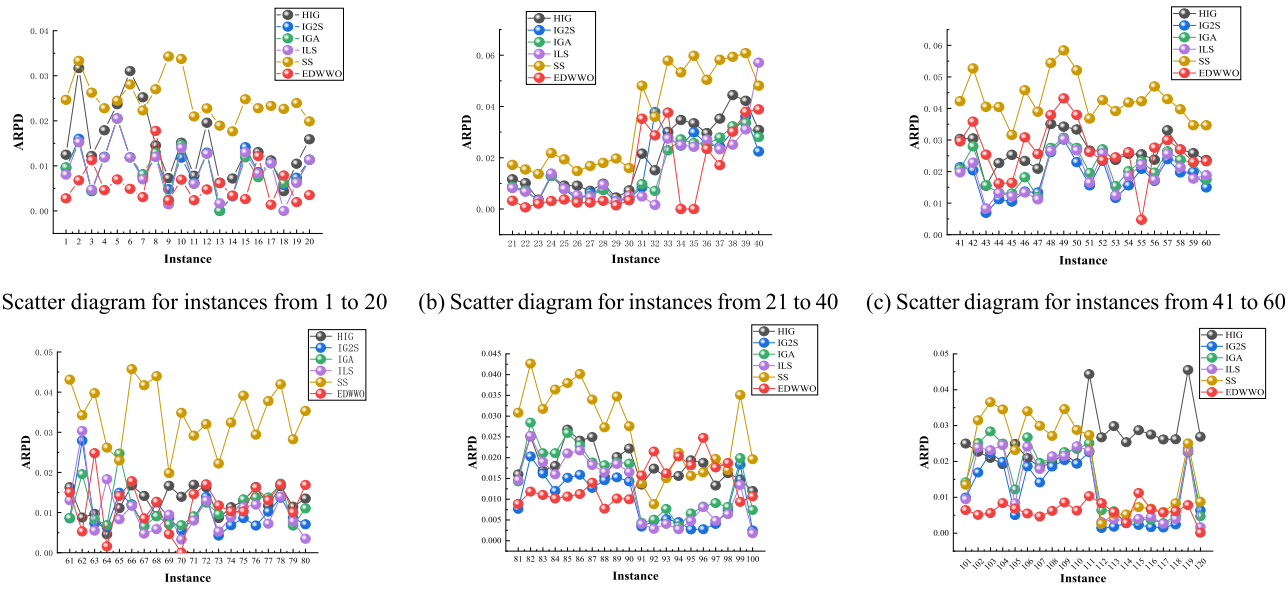


Fig. 14 Pie chart of the proportion of the algorithms with optimum. (**a**) Scatter diagram for instances from 1 to 20. (**b**) Scatter diagram for instances from 21 to 40. (**c**) Scatter diagram for instances from 41 to 60. (**d**) Scatter diagram for instances from 61 to 80. (**e**) Scatter diagram for instances from 81 to 100. (**f**) Scatter diagram for instances from 101 to 120

(2) Non-parametric test

Afterward, the Friedman-test and Wilcoxon-test are conducted as the non-parametric post hoc tests to investigate the statistical validity of the results. The procedure of the two tests, the results of the experimental comparisons, and the analysis are given in this section.

The Friedman-test is a statistical test of the homogeneity of multiple related samples. In the Friedman test, all the compared algorithms are considered as different factors. The mean rank with 95% confidence intervals of the interactions



(a) Scatter diagram for instances from 1 to 20

(b) Scatter diagram for instances from 21 to 40

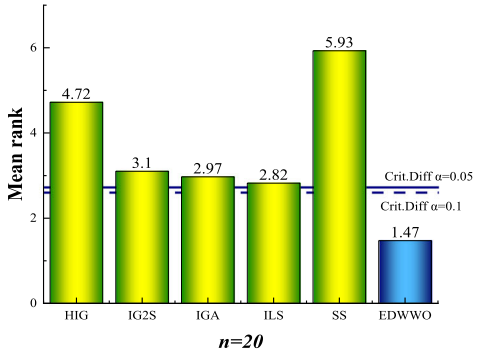
(c) Scatter diagram for instances from 41 to 60

(d) Scatter diagram for instances from 61 to 80

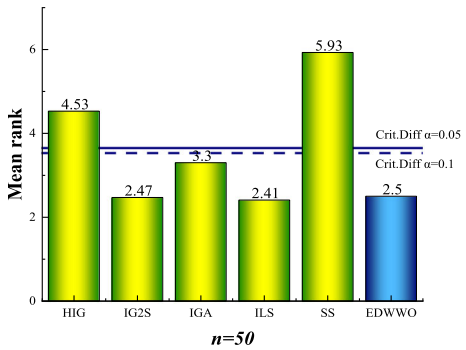
(e) Scatter diagram for instances from 81 to 100

(f) Scatter diagram for instances from 101 to 120

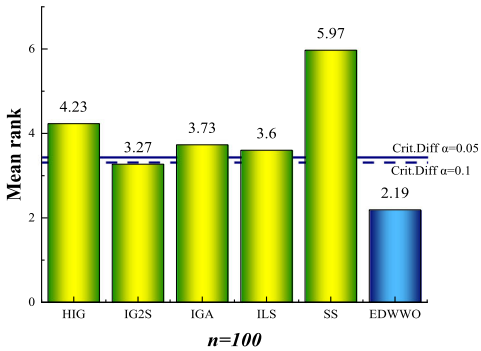
Fig. 13 Scatter diagrams of the experimental results



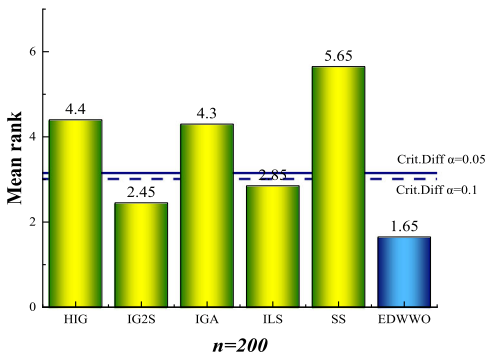
(a) Bonferroni–Dunn's graphic for n=20



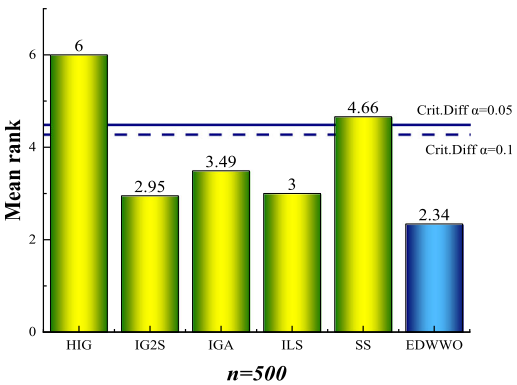
(b) Bonferroni–Dunn's graphic for n=50



(c) Bonferroni–Dunn's graphic for n=100



(d) Bonferroni–Dunn's graphic for n=200



(e) Bonferroni–Dunn's graphic for n=500

Fig. 15 The Bonferroni–Dunn's graphic of the mean rank obtained by the algorithms. (**a**) Bonferroni–Dunn's graphic for n=20. (**b**) Bonferroni–Dunn's graphic for n=50. (**c**) Bonferroni–Dunn's graphic for n=100. (**d**) Bonferroni–Dunn's graphic for n=200. (**e**) Bonferroni–Dunn's graphic for n=500

between compared algorithms for the ARPD value is illustrated in Table 8. Note that the results of the Friedman-test are grouped into five groups based on the size of n. Compared to the other algorithms, the lower the mean rank value of an algorithm, the better the performance of the algorithm.

Additionally, the Bonferroni–Dunn experiment is carried out to evaluate whether the differences of the experimental results are significant or not. The critical differences at the 95% (α=0.05) and 90% (α=0.1) confidence intervals are utilized as the test thresholds. The values of the critical differences are also listed in Table 8. The results of the Bonferroni–Dunn experiment are shown in Fig. 15. The solid line and dotted line represent 95% confidence interval and 90% confidence interval, respectively. The cylinders that represent the algorithms being below the thresholds means that the statistical difference between them is not significant. On the contrary, there are significant differences between them. As shown in Table 8, the mean rank of EDWWO is the smallest

Table 8 Friedman-test results of the ARPD obtained by the algorithms

Algorithm	Mean rank				
	n=20	n=50	n=100	n=200	n=500
HIG	4.72	4.53	4.23	4.40	6.00
IG2S	3.10	**2.47**	3.27	2.45	2.95
IGA	2.97	3.30	3.73	4.30	3.49
ILS	2.82	2.41	3.60	2.85	3.00
SS	5.93	5.93	5.97	5.65	4.66
EDWWO	**1.47**	2.50	**2.19**	**1.65**	**2.34**
Crit.Diff α=0.05	1.24	1.24	1.24	1.52	2.16
Crit.Diff α=0.1	1.12	1.12	1.12	1.38	1.95

in the six compared algorithms when the number of jobs is 20, 100, 20, and 500, which demonstrates that the EDWWO significantly outperforms other compared algorithms. From Fig. 15(a), it is clear that only EDWWO is lower than the thresholds. Thus, the EDWWO algorithm has a significant difference from the compared algorithms. According to Fig. 15(b), all the compared algorithms are divided into two different groups with no significant differences within each group. The two groups, from worst to best are {SS, HIG} {IGA, IG2S, ILS, EDWWO}. The results in Fig. 15(c) to (e) are almost unchanged.

It is not affirmed that there is a significant difference among the algorithms; Therefore, an additional test based on the Wilcoxon test is applied to further observe the differences between EDWWO and the well-established algorithms. The Wilcoxon-test is a parametric method of non-parametric testing for paired observations [72]. Due to the reasons of space, the ARPD values obtained by the algorithms on the 120 instances are summarized according to the scale of $n \times m$, and the results of the Wilcoxon-test are shown in Table 9. It is noted that the symbols of (+), and (−) indicate that the result of EDWWO is better than and worse than its competitor on the corresponding quantity of problems, respectively. Meanwhile, $CS_{(+)}$ and $CS_{(-)}$ return the numbers plus sign (+) and minus sign (−) on each of the scales of $n \times m$. As shown in Table 9, $RS_{(+)}$ and $CS_{(+)}$ are always greater than $RS_{(-)}$ and $CS_{(-)}$, which means that EDWWO is better than the other compared algorithms. The P-values in the rows are 1.97E-17, 3.17E-04, 1.68E-04, 4.75E-08, and 2.60E-21 with a 95% confidence interval. All the P-values are less than 0.05, which indicates that there are significant differences between EDWWO and the corresponding compared algorithms.

Following the above comparisons and analysis, the conclusions are safely established that the proposed EDWWO algorithm is more effective and efficient in solving the BFSP with the makespan criterion. The superiority of the proposed EDWWO is mainly attributed to the following aspects: (1) the proposed SNEH heuristic generates promising initial solutions; (2) the performance of EDWWO is effectively improved by technologies in terms of the analysis of the components.

4 Conclusions and future work

In this paper, the blocking flow-shop scheduling problem (BFSP) with the objective of minimizing the makespan is investigated. As consequence, an ensemble discrete water wave optimization algorithm (EDWWO) is proposed to solve the considered problem. In the initialization, an effective constructive heuristic, named SNEH, is developed based on a new dispatching rule and the classical NEH to generate solutions with high quality. The data-driven mechanism with the destruction-reorganization strategy in the propagation phase enhances the balance between exploration and exploitation. Meanwhile, the optimal block-shifting operator under the framework of variable neighborhood search intensifies exploitation in the local region for finding more promising solutions. Subsequently, the perturbation strategy in the refraction phase effectively avoids algorithm trapping into the local optimum.

Table 9 Wilcoxon-test results of the ARPD between one compared algorithm and EDWWO

EDWWO *VS*	n=20			n=50			n=100			n=200		n=500	$RS_{(+)}$	$RS_{(-)}$	P-value
	m=5	m=10	m=20	m=5	m=10	m=20	m=5	m=10	m=20	m=10	m=20	m=20			
HIG	+	+	+	+	+	+	+	+	+	+	+	+	12	0	*1.97E-17*
IG2S	+	+	+	+	+	-	+	+	+	+	+	-	10	2	*3.17E-04*
IGA	+	+	+	+	+	+	+	+	+	+	+	+	12	0	*1.68E-04*
ILS	+	+	+	-	+	+	-	+	+	+	+	+	10	2	*4.75E-08*
SS	+	+	+	+	+	+	+	+	+	+	+	+	12	0	*2.60E-21*
$CS_{(+)}$	5	5	5	4	5	4	4	5	5	4	5	4	*NULL*	*NULL*	*NULL*
$CS_{(-)}$	0	0	0	1	0	1	1	0	0	0	0	1	*NULL*	*NULL*	*NULL*

Design of experiment and ANOVA are utilized to analyze and calibrate the main parameters of EDWWO. Moreover, the effectiveness of the SNEH is evaluated by comparing it with three efficient initial methods. The impact of each improvement strategy in EDWWO is analyzed through several numerical experiments. The proposed EDWWO is verified based on 120 benchmark instances and five other competitive algorithms. The statistical results confirm that EDWWO is an effective algorithm for the considered problem.

In the future, it is a worthwhile direction to consider the total energy consumption in the production process. Furthermore, it is significant to apply EDWWO to solve complex scheduling problems, such as the distributed flow-shop scheduling problems and the flow-shop scheduling problems with sequence-dependent setup times.

Funding 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.

Declarations

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

Conflict of interest All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

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

References

1. Wikarek J, Sitek P (2021) Proactive and reactive approach to employee competence configuration problem in planning and scheduling processes. Appl Intell 2021:1–20. https://doi.org/10.1007/S10489-021-02594-X
2. Ivanov D, Dolgui A, Sokolov B et al (2016) A dynamic model and an algorithm for short-term supply chain scheduling in the smart factory industry 4.0. Int J Prod Res 54:386–402. https://doi.org/10.1080/00207543.2014.999958
3. Zhou G, Zhou Y, Zhao R (2021) Hybrid Social Spider Optimization Algorithm with Differential Mutation Operator for the Job-Shop Scheduling Problem. J Ind Manag Optim 17:533–548. https://doi.org/10.3934/JIMO.2019122
4. Wan J, Chen B, Wang S et al (2018) Fog Computing for Energy-Aware Load Balancing and Scheduling in Smart Factory. IEEE Trans Ind Informatics 14:4548–4556. https://doi.org/10.1109/TII.2018.2818932
5. Fang CJ, Wang L, Ping PZ (2019) A collaborative optimization algorithm for energy-efficient multi-objective distributed no-idle flow-shop scheduling. Swarm Evol Comput:50. https://doi.org/10.1016/J.SWEVO.2019.100557
6. Zhao F, Ma R, Wang L (2021) A Self-Learning Discrete Jaya Algorithm for Multiobjective Energy-Efficient Distributed No-Idle Flow-Shop Scheduling Problem in Heterogeneous Factory System. IEEE Trans Cybern. https://doi.org/10.1109/TCYB.2021.3086181
7. Lu C, Huang Y, Meng L et al (2022) A Pareto-based collaborative multi-objective optimization algorithm for energy-efficient scheduling of distributed permutation flow-shop with limited buffers. Robot Comput Integr Manuf 74:102277. https://doi.org/10.1016/J.RCIM.2021.102277
8. Shao Z, Shao W, Pi D (2020) Effective Constructive Heuristic and Metaheuristic for the Distributed Assembly Blocking Flow-shop Scheduling Problem. Appl Intell 50(12):4647–4669. https://doi.org/10.1007/S10489-020-01809-X
9. Hamzadayı A, Arvas MA, Elmi A (2021) Distributed assembly permutation flow shop problem; Single seekers society algorithm. J Manuf Syst 61:613–631. https://doi.org/10.1016/J.JMSY.2021.10.012
10. Chen S, Pan QK, Gao L (2021) Production scheduling for blocking flowshop in distributed environment using effective heuristics and iterated greedy algorithm. Robot Comput Integr Manuf:71. https://doi.org/10.1016/J.RCIM.2021.102155
11. Zhao F, He X, Wang L (2020) A Two-Stage Cooperative Evolutionary Algorithm With Problem-Specific Knowledge for Energy-Efficient Scheduling of No-Wait Flow-Shop Problem. IEEE Trans Cybern:1–13. https://doi.org/10.1109/tcyb.2020.3025662
12. Rossi FL, Nagano MS (2021) Heuristics and iterated greedy algorithms for the distributed mixed no-idle flowshop with sequence-dependent setup times. Comput Ind Eng:157. https://doi.org/10.1016/J.CIE.2021.107337
13. Shao Z, Pi D, Shao W (2019) A novel multi-objective discrete water wave optimization for solving multi-objective blocking flow-shop scheduling problem. Knowledge-Based Syst 165:110–131. https://doi.org/10.1016/j.knosys.2018.11.021
14. Chen Q, Pan Q, Zhang B, et al. (2019) Algorithms in Compact Strip Production 16:1933–1951
15. Merchan AF, Maravelias CT (2016) Preprocessing and tightening methods for time-indexed MIP chemical production scheduling models. Comput Chem Eng 84:516–535. https://doi.org/10.1016/J.COMPCHEMENG.2015.10.003
16. Riahi V, Khorramizadeh M, Newton MAH, Sattar A (2017) Scatter search for mixed blocking flowshop scheduling. Expert Syst Appl 79:20–32. https://doi.org/10.1016/j.eswa.2017.02.027
17. Miyata HH, Nagano MS (2019) The blocking flow shop scheduling problem: A comprehensive and conceptual review. Expert Syst. Appl. 137:130–156
18. Zhao F, Xue F, Zhang Y et al (2019) A discrete gravitational search algorithm for the blocking flow shop problem with total flow time minimization. Appl Intell 499(49):3362–3382. https://doi.org/10.1007/S10489-019-01457-W
19. He X, Pan Q, Gao L et al (2021) A Greedy Cooperative Co-evolution ary Algorithm with Problem-specific Knowledge for Multi-objective Flowshop Group Scheduling Problems. IEEE Trans Evol Comput:1–1. https://doi.org/10.1109/TEVC.2021.3115795
20. Nawaz M, Enscore EE, Ham I (1983) A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega 11:91–95. https://doi.org/10.1016/0305-0483(83)90088-9
21. McCormick ST, Pinedo ML, Shenker S, Wolf B (1989) Sequencing in an assembly line with blocking to minimize cycle time. Oper Res 37:925–935. https://doi.org/10.1287/OPRE.37.6.925
22. Wang L, Pan QK, Suganthan PN et al (2010) A novel hybrid discrete differential evolution algorithm for blocking flow shop

scheduling problems. Comput Oper Res 37:509–520. https://doi.org/10.1016/j.cor.2008.12.004
23. Liang JJ, Pan QK, Tiejun C, Wang L (2011) Solving the blocking flow shop scheduling problem by a dynamic multi-swarm particle swarm optimizer. Int J Adv Manuf Technol 55:755–762. https://doi.org/10.1007/S00170-010-3111-7
24. Wang L, Pan QK, Tasgetiren MF (2011) A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem. Comput Ind Eng 61:76–83. https://doi.org/10.1016/J.CIE.2011.02.013
25. Han YY, Gong D, Sun X (2015) A discrete artificial bee colony algorithm incorporating differential evolution for the flow-shop scheduling problem with blocking. Eng Optim 47:927–946. https://doi.org/10.1080/0305215X.2014.928817
26. Han Y, Gong D, Li J, Zhang Y (2016) Solving the blocking flow shop scheduling problem with makespan using a modified fruit fly optimisation algorithm. Int J Prod Res 54:6782–6797. https://doi.org/10.1080/00207543.2016.1177671
27. Eddaly M, Jarboui B, Siarry P (2016) Combinatorial particle swarm optimization for solving blocking flowshop scheduling problem. J Comput Des Eng 3:295–311. https://doi.org/10.1016/j.jcde.2016.05.001
28. Shao Z, Pi D, Shao W (2018) Estimation of distribution algorithm with path relinking for the blocking flow-shop scheduling problem. Eng Optim 50:894–916. https://doi.org/10.1080/0305215X.2017.1353090
29. Shao Z, Pi D, Shao W (2018) A multi-objective discrete invasive weed optimization for multi-objective blocking flow-shop scheduling problem. Expert Syst Appl 113:77–99. https://doi.org/10.1016/j.eswa.2018.06.020
30. Abu Doush I, Al-Betar MA, Awadallah MA et al (2019) Flow shop scheduling with blocking using modified harmony search algorithm with neighboring heuristics methods. Appl Soft Comput J. https://doi.org/10.1016/J.ASOC.2019.105861
31. Shao Z, Pi D, Shao W (2017) Self-adaptive discrete invasive weed optimization for the blocking flow-shop scheduling problem to minimize total tardiness. Comput Ind Eng 111:331–351. https://doi.org/10.1016/j.cie.2017.07.037
32. Shao Z, Pi D, Shao W (2018) A novel discrete water wave optimization algorithm for blocking flow-shop scheduling problem with sequence-dependent setup times. Swarm Evol Comput 40:53–75. https://doi.org/10.1016/j.swevo.2017.12.005
33. Han Y, Li J, Sang H et al (2020) Discrete evolutionary multi-objective optimization for energy-efficient blocking flow shop scheduling with setup time. Appl Soft Comput J 93. https://doi.org/10.1016/J.ASOC.2020.106343
34. Ribas I, Companys R, Tort-Martorell X (2017) Efficient heuristics for the parallel blocking flow shop scheduling problem. Expert Syst Appl 74:41–54. https://doi.org/10.1016/j.eswa.2017.01.006
35. Ribas I, Companys R, Tort-Martorell X (2019) An iterated greedy algorithm for solving the total tardiness parallel blocking flow shop scheduling problem. Expert Syst Appl 121:347–361. https://doi.org/10.1016/j.eswa.2018.12.039
36. Shao Z, Pi D, Shao W (2020) Hybrid enhanced discrete fruit fly optimization algorithm for scheduling blocking flow-shop in distributed environment. Expert Syst Appl 145:113147. https://doi.org/10.1016/j.eswa.2019.113147
37. Shao Z, Shao W, Pi D (2020) Effective heuristics and metaheuristics for the distributed fuzzy blocking flow-shop scheduling problem. Swarm Evol Comput:59. https://doi.org/10.1016/J.SWEVO.2020.100747
38. Shao Z, Shao W, Pi D (2021) Effective constructive heuristic and iterated greedy algorithm for distributed mixed blocking permutation flow-shop scheduling problem. Knowledge-Based Syst:221. https://doi.org/10.1016/J.KNOSYS.2021.106959
39. Ribas I, Companys R (2021) A computational evaluation of constructive heuristics for the parallel blocking flow shop problem with sequence-dependent setup times. Int J Ind Eng Comput 12:321–328. https://doi.org/10.5267/J.IJIEC.2021.1.004
40. Ribas I, Companys R, Tort-Martorell X (2021) An iterated greedy algorithm for the parallel blocking flow shop scheduling problem and sequence-dependent setup times. Expert Syst Appl:184. https://doi.org/10.1016/J.ESWA.2021.115535
41. Zhao F, Zhao L, Wang L, Song H (2020) An ensemble discrete differential evolution for the distributed blocking flowshop scheduling with minimizing makespan criterion. Expert Syst Appl 160:113678. https://doi.org/10.1016/j.eswa.2020.113678
42. Li YY, Lin J, Wang ZJ (2021) Multi-skill resource constrained project scheduling using a multi-objective discrete Jaya algorithm. Appl Intell. https://doi.org/10.1007/S10489-021-02608-8
43. Tasgetiren MF, Kizilay D, Pan QK, Suganthan PN (2017) Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion. Comput Oper Res 77:111–126. https://doi.org/10.1016/j.cor.2016.07.002
44. Sang H, Pan Q, Li J et al (2019) Effective invasive weed optimization algorithms for distributed assembly permutation flowshop problem with total flowtime criterion. Swarm Evol Comput 44:64–73. https://doi.org/10.1016/j.swevo.2018.12.001
45. Zhou Y, Zhang J, Yang X, Ling Y (2019) Optimization of PID controller based on water wave optimization for an automatic voltage regulator system. Inf Technol Control 48:160–171. https://doi.org/10.5755/J01.ITC.48.1.20296
46. Zheng Y (2015) Water wave optimization: A new nature-inspired metaheuristic. Comput Oper Res 55:1–11. https://doi.org/10.1016/j.cor.2014.10.008
47. Zhang J, Zhou Y, Luo Q (2018) An improved sine cosine water wave optimization algorithm for global optimization. J Intell Fuzzy Syst 34:2129–2141. https://doi.org/10.3233/JIFS-171001
48. Zhang J, Zhou Y, Luo Q (2019) Nature-inspired approach: a wind-driven water wave optimization algorithm. Appl Intell 49:233–252. https://doi.org/10.1007/S10489-018-1265-4
49. Medara R, Singh RS, Amit (2021) Energy-aware workflow task scheduling in clouds with virtual machine consolidation using discrete water wave optimization. Simul Model Pract Theory 110:102323. https://doi.org/10.1016/j.simpat.2021.102323
50. Yan Z, Zhang J, Tang J (2021) Path planning for autonomous underwater vehicle based on an enhanced water wave optimization algorithm. Math Comput Simul 181:192–241. https://doi.org/10.1016/j.matcom.2020.09.019
51. Lu XQ, Yan HF, Su ZL et al (2021) Metaheuristics for homogeneous and heterogeneous machine utilization planning under reliability-centered maintenance. Comput Ind Eng 151:106934. https://doi.org/10.1016/j.cie.2020.106934
52. Zhou Y, Zhang J, Yang X, Ling Y (2020) Optimal reactive power dispatch using water wave optimization algorithm. Oper Res 20:2537–2553. https://doi.org/10.1007/S12351-018-0420-3
53. Yun X, Feng X, Lyu X et al (2016) A novel water wave optimization based memetic algorithm for flow-shop scheduling. IEEE Congr Evol Comput 2016:1971–1976
54. Zhao F, Liu H, Zhang Y et al (2018) A discrete Water Wave Optimization algorithm for no-wait flow shop scheduling problem. Expert Syst Appl 91:347–363. https://doi.org/10.1016/j.eswa.2017.09.028
55. Zhao F, Zhang L, Zhang Y et al (2020) A hybrid discrete water wave optimization algorithm for the no-idle flowshop scheduling problem with total tardiness criterion. Expert Syst Appl 146. https://doi.org/10.1016/j.eswa.2019.113166
56. Zhao F, Zhang L, Cao J, Tang J (2021) A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem. Comput Ind Eng 153:107082. https://doi.org/10.1016/j.cie.2020.107082

57. Zhao F, Zhang L, Liu H et al (2019) An improved water wave optimization algorithm with the single wave mechanism for the no-wait flow-shop scheduling problem. Eng Optim 51:1727–1742. https://doi.org/10.1080/0305215X.2018.1542693
58. Lawler EL, Lenstra JK, Rinnooy Kan AHG (1982) Recent Developments in Deterministic Sequencing and Scheduling: A Survey. Determ Stoch Sched:35–73. https://doi.org/10.1007/978-94-009-7801-0_3
59. Bhatt KC, Malav PK, Gore PG et al (2021) A note on distribution and potential of Japanese wild adzuki bean [Vigna angularis var. nipponensis (Ohwi) Ohwi and H. Ohashi] in India. Genet Resour Crop Evol 68:2157–2166. https://doi.org/10.1007/s10722-021-01130-7
60. Zhao F, Zhao J, Wang L, Tang J (2021) An optimal block knowledge driven backtracking search algorithm for distributed assembly No-wait flow shop scheduling problem. Appl Soft Comput 112:107750. https://doi.org/10.1016/j.asoc.2021.107750
61. Ding JY, Song S, Zhang R et al (2015) A novel Block-shifting simulated annealing algorithm for the no-wait flowshop scheduling problem. IEEE Congr Evol Comput 2015:2768–2774. https://doi.org/10.1109/CEC.2015.7257232
62. Rossi FL, Nagano MS (2020) Heuristics and metaheuristics for the mixed no-idle flowshop with sequence-dependent setup times and total tardiness minimisation. Swarm Evol Comput 55:100689. https://doi.org/10.1016/j.swevo.2020.100689
63. Khare A, Agrawal S (2020) Effective heuristics and metaheuristics to minimise total tardiness for the distributed permutation flowshop scheduling problem. Int J Prod Res 0:1–17. https://doi.org/10.1080/00207543.2020.1837982
64. Smith JR, Larson C (2019) Statistical approaches in surface finishing. Part 3. Design-of-experiments. Trans Inst Met Finish 97:289–294. https://doi.org/10.1080/00202967.2019.1673530
65. Jaeger TF (2008) Categorical data analysis: Away from ANOVAs (transformation or not) and towards logit mixed models. J Mem Lang 59:434–446. https://doi.org/10.1016/j.jml.2007.11.007
66. Pan QK, Gao L, Xin-Yu L, Framinan JM (2019) Effective constructive heuristics and meta-heuristics for the distributed assembly permutation flowshop scheduling problem. Appl Soft Comput 81:105492. https://doi.org/10.1016/j.asoc.2019.105492
67. Zhang G, Xing K, Cao F (2018) Discrete differential evolution algorithm for distributed blocking flowshop scheduling with makespan criterion. Eng Appl Artif Intell 76:96–107. https://doi.org/10.1016/j.engappai.2018.09.005
68. Ying KC, Lin SW, Cheng CY, He CD (2017) Iterated reference greedy algorithm for solving distributed no-idle permutation flowshop scheduling problems. Comput Ind Eng 110:413–423. https://doi.org/10.1016/j.cie.2017.06.025
69. Ruiz R, Pan QK, Naderi B (2019) Iterated Greedy methods for the distributed permutation flowshop scheduling problem. Omega (United Kingdom) 83:213–222. https://doi.org/10.1016/j.omega.2018.03.004
70. Ribas I, Companys R, Tort-Martorell X (2017) Efficient heuristics for the parallel blocking flow shop scheduling problem. Expert Syst Appl 74:41–54. https://doi.org/10.1016/j.eswa.2017.01.006
71. Naderi B, Ruiz R (2014) A scatter search algorithm for the distributed permutation flowshop scheduling problem. Eur J Oper Res 239:323–334. https://doi.org/10.1016/j.ejor.2014.05.024
72. Garcia S, Molina D, Lozano M, Herrera F (2009) 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 15:617–644. https://doi.org/10.1007/s10732-008-9080-4

Fuqing Zhao received the B.Sc. and Ph.D. degrees from the Lanzhou University of Technology, Lanzhou, China, in 1994 and 2006, respectively. Since 1998, he has been with the School of Computer Science Department, Lanzhou University of Technology, Lanzhou, China, where he became a Full Professor in 2012. He has been as the post Doctor with the State Key Laboratory of Manufacturing System Engineering, Xi'an Jiaotong University, Xi'an, China in 2009. He has been as a visiting scholar in Exeter Manufacturing Enterprise Center in Exeter University and Georgia Tech Manufacturing Institute in Georgia Institute of Technology from 2008-2019 and 2014-2015 respectively. He has authored two academic book and over 50 refereed papers. His current research interests include intelligent optimization and scheduling.