Original Paper

MEASUREMENT AND CONTROL

Measurement and Control
2023, Vol. 56(9-10) 1764–1779
© The Author(s) 2023
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/00202940231180622
journals.sagepub.com/home/mac
Sage

A discrete whale optimization algorithm for the no-wait flow shop scheduling problem

Sujun Zhang[1] and Xingsheng Gu[2]

Abstract

A discrete whale optimization algorithm (DWOA) is presented to solve the no-wait flow shop scheduling problem (NWFSSP) with the optimization objective makespan. An effective combination of nearest neighbor (NN) and standard deviation heuristics (SDH) is used to acquire initial solutions of the population. After that, three crossover operators, the two-point crossover (TPX), multiple-point crossover (MPX) and job-based crossover (JBX) operators, are designed to mimics the humpback whales hunting process. Moreover, the dynamic transform mechanism of search process is designed to better balance the exploration and exploitation ability of DWOA. In order to further improve the optimization effect of DWOA, the parallel neighborhood search (PNS) and the serial neighborhood search (SNS) are selected to execute the local search and promoting global search scheme, respectively. Finally, to test the performance of DWOA, extensive experiments are conducted on benchmarks designed by Reeves and Taillard, which contain parameters tuning, effectiveness of the improvement strategies in the DWOA and comparison with several existing algorithms. From the experimental results, it is validated that the effectiveness of the improved mechanisms and the performance of the DWOA which can find better makespan values compared with other algorithms for solving NWFSSP.

Keywords

Discrete whale optimization algorithm, no-wait flow shop scheduling problem, parallel neighborhood search, serial neighborhood search

Date received: 23 December 2022; accepted: 11 May 2023

Introduction

Production scheduling is key link in manufacturing system. So, a good scheduling scheme can not only enhance productivity but also reduce cost. The no-wait flow shop scheduling problem (NWFSSP) has strong industrial application background and it widely applies to many industrial processes such as chemical processing, steel rolling, food industry, plastic molding, flexible manufacturing systems, robotic cells, aircraft landing problem, surgery scheduling and etc.[1] It is an essential branch of the flow shop scheduling problem (FSSP). Due to the no-wait requirement, the jobs have to be postponed on the first machine.[2] NWFSSP with the makespan criterion is a typical combinational optimization problem and has been proven to be NP-hard even when the number of machine is only two.[3] When the problem size increases, the problem becomes complicated, and it will be difficult to solve with an exact method in a reasonable time. However, heuristics and constructive heuristics algorithms are usually used to solve small-scale problems, or they are executed to get the initial solution of the population for meta-heuristic algorithms due to simplicity. Thus, many researchers are dedicated to finding various efficient meta-heuristic algorithms and to studying the more complex extensive model.

The branch and bound approach[4] is proposed to solve the aircraft manufacturing industry which can be modeled as NWFSSP, but it only can solve the small-scale problem. However, the new modeling technique and an enumeration algorithm are proposed to address the NWFSSP with the makespan criterion and due date constraints, which can shrink the feasible region and enhance the search speed.[5] The heuristics and constructive

[1]School of Mechanical and Electrical Engineering, Henan Institute of Science and Technology, Xinxiang, China
[2]Key Laboratory of Smart Manufacturing in Energy Chemical Process, Ministry of Education, East China University of Science and Technology, Shanghai, China

Corresponding author:
Xingsheng Gu, Key Laboratory of Smart Manufacturing in Energy Chemical Process, Ministry of Education, East China University of Science and Technology, Shanghai 200237, China.
Email: xsgu@ecust.edu.cn

heuristics[6–8] are proposed to resolve the NWFSSP with makespan or total flow-time minimization criterion. In addition, to solve the NWFSSP with large size, a jigsaw puzzle-inspired algorithm is presented.[9] However, meta-heuristics have a remarkable advantage over exact and heuristic methods in solving NWFSSP. Some meta-heuristics, containing genetic algorithm (GA),[10] discrete particle swarm optimization algorithm(DPSO$_{\mathrm{VND}}$)[11] and discrete particle swarm optimization mixed with the other algorithm (HDPSO),[12] discrete harmony search algorithm (DHS),[13] ant colony optimization algorithm,[14] a discrete[15] and an extended optimization algorithm[16] have been developed for resolving NWFSSP with the objective total flowtime or makespan. In addition, two efficient metaheuristics[17] are developed to solve NWFSSP for a set of large-scale scheduling problems with 2000-job and 20-machine, and can get better solutions. Depending on single individual evolution, two improved versions of an iterated greedy algorithm[18,19] also have been employed to NWFSSP because of its simplicity and effectiveness. However, the swarm-based intelligent optimization algorithm has the distinctive merit for tackling the optimization problem. A discrete wolf pack is addressed to solve the NWFSSP and gains better optimization.[20] In view of the merit of quantum algorithms, it is used to improve the optimal effect in cuckoo co-search algorithm[21] for solving NWFSSP. HMM-FPA[22] is provided to solve the NWFSSP, in which it takes advantage of the hormone modulation mechanism to improve the exploring ability of FPA. Recently, Some swarm-based intelligence optimization algorithms with better performance, such as a discrete version algorithm based on water wave optimization (DWWO),[23] a particle swarm optimization mixed the adaptation mechanism,[24] and a hybrid biogeography-based optimization[25] and an improved discrete version of migrating bird algorithm,[26] have presented to solve NWFSSP. It is worth noting that the variable neighborhood search (VNS) is embedded in the above four algorithms[23–26] directly and indirectly for generating neighborhood solutions and getting remarkable result. Besides the makespan and total flow time criterion, the total earliness and tardiness criterion are often considered when solving the NWFSSP. Some heuristics[27,28] are presented to address the NWFSSP, and designed the speed-up method and several forms of insertion to enhance the optimization effect of the algorithm. Moreover, as the extension, the scheduling problems which take into account dependent sequence setup times, energy-efficient and distributed features have attracted the attention of scholars and new soft computing strategies have proposed.[29–37]

The whale optimization algorithm (WOA)[38] is a population-based meta-heuristic algorithm, which mimics the behavior of humpback whales to prey. Due to few adjustment parameters and strong optimization ability, it has been applied directly or indirectly successfully to function optimization,[39] feature extraction,[40] image segmentation,[41] and other continuous optimization problems. Whereas, it can't be directly applied to the optimization problems with discrete property and has the slower convergence. Although there have been some research results, such as an enhanced WOA[42] and a multi-objective WOA[43] have applied to the production scheduling problems. However, they are not all directly discrete encoding, but individual positions (continuous variables) are converted into scheduling solutions (job permutation) through LPT or ROV rule, which will reduce the optimization performance of the algorithm. It has proven that discrete whale optimization algorithm (DWOA) encoding based on job-permutation representation can acquire the better performance, such as the WOA and an effective multi-objective WOA employed to solve the flexible job shop[44] and distributed welding flow shop scheduling problem,[45] respectively. To our best knowledge, it hasn't yet been studied that DWOA is employed to resolve the NWFSSP, and it is also rare to directly design DWOA for production scheduling. Hence, the DWOA is designed to solve the NWFSSP in this work due to the significance of researching NWFSSP in engineering applications, the excellent performance of whale optimization algorithm and the requirement of researching systemic methods in theory.

The rest contents are summarized as follows. Section 2 provides the model of NWFSSP. Section 3 and Section 4 present the WOA and DWOA designed in this paper, respectively. Section 5 conducts the simulation experiments and analyzes the results. However, Section 6 draws some conclusions and future research interests.

The no-wait flow shop scheduling problem (NWFSSP)

The NWFSSP can be depicted that n jobs are processed sequentially by m machines according to the same order; that is, each job has the same processing routes on machines. The processing time $p(j,k)$, which is the time job j ($j \in \{1,2,\cdots,n\}$) to be processed on machine k ($k \in \{1,2,\cdots,m\}$), is predefined and it contained the setup times (the time of machine cleaning or tool changing) or ignored. At any time, a job can't be operated by more than one machine, and one machine can't process more than one job at the same time. To satisfy the no-wait constraint, once the job starts to be processed, there can be no waiting times from one machine to the next machine. It implies that a job must be detained on the first machine before starting to process. In this paper, the optimization objective is to find a feasible schedule π with minimum processing completion time (i.e. makespan) for n jobs in a reasonable time.

The schedule $\pi = \{\pi_1, \pi_2, \cdots, \pi_n\}$ indicates a processing sequence of n jobs on all machines. Due to the no-wait restriction, the minimum detained (md) time before starting to be processed from jobs π_{j-1} and next job π_j is expressed by $md(\pi_{j-1}, \pi_j)$. The time of the job π_j processed by machine k is expressed by $p(\pi_j, k)$.$C_{\max}(\pi)$ is the makespan of scheduling permutation π, and it equals the completion time of the last job π_n processed by the last machine, ct_n. However, ct_1 is the completion time of the first job π_1 and it can be calculated by equation (1).

$$ct_1 = \sum_{k=1}^{m} p(\pi_1, k) \tag{1}$$

And the processing completion time of the job π_j can be calculated by equation (2), where the minimum detained time $md(\pi_{i-1}, \pi_i)$ is given by equation (3).

$$ct_j = \sum_{i=2}^{j} md(\pi_{i-1}, \pi_i) + \sum_{k=1}^{m} p(\pi_j, k) \quad j = 2, 3, \cdots, n \tag{2}$$

$$md(\pi_{i-1},\pi_i) = p(\pi_{i-1},1) + \max\left\{ \max_{2 \leqslant k \leqslant m} \left(\sum_{h=2}^{k} p(\pi_{i-1},h) - \sum_{h=1}^{k-1} p(\pi_i,h) \right), 0 \right\} \tag{3}$$

Firstly, the completion time of the first job π_1 is calculated by equation (1). Then, all others jobs except for π_1 processing completion time on machine m is given by equation (2), where $md(\pi_{i-1}, \pi_i)$ is associated with the operating time of the two sequential jobs (π_{i-1} and π_i). Finally, the makespan of the schedule $\pi = \{\pi_1, \pi_2, \cdots \pi_n\}$ is given by equation (4).

$$C_{\max}(\pi) = ct_n \tag{4}$$

$$C_{\max}(\pi^*) \leqslant C_{\max}(\pi), \forall \pi \in \Pi \tag{5}$$

Therefore, equation (5) shows that the purpose of scheduling is to find out a schedule π^* which has the minimum makespan from the set of all the feasible schedules Π.

The whale optimization algorithm

WOA[38] is proposed by Mirjalili which is inspired by the predation of the humpback whales. It is a novel swarm intelligent optimization algorithm and is usually employed to address optimization and classical engineering problem. When the whales found the prey, they swam in a spiral way toward the prey and encircled it, meanwhile they foraged the prey by emitting a bubble net. The process contains three kinds of hunting methods, namely, surrounding prey, shrinking and attacking by bubble-net, and searching prey randomly. The first two and the third completed the exploitation and the exploration of WOA algorithm, respectively. Thus, the mathematics model of the hunting process of humpback whales can be described as follows.

Encircling prey

Because the target location is unknown, the best individual of the population is selected as the target prey. Once other individuals discover the prey, they will surround to encircle prey. It can be expressed as follows.

$$D = |C \cdot XX^*(t) - XX(t)| \tag{6}$$

$$XX(t+1) = XX^*(t) - A \cdot D \tag{7}$$

$$A = 2a \cdot r - a \tag{8}$$

$$C = 2r \tag{9}$$

In equations (6) and (7), $XX(t)$ and $XX^*(t)$ denote the contemporary individual and the optimal individual at t iteration, respectively; the distance between two vectors is D. The coefficient vectors A and C which are expressed by equations (8) and (9), respectively. Where r is the random vector its value between 0 and 1, and a is linearly decreased from 2 to 0 expressed by equation (10), where t and $t_{\max}$ indicate the current and the maximum evolutionary algebras.

$$a = 2 - \frac{2t}{t_{\max}} \tag{10}$$

Shrinking encircling and in a spiral way updating position

The whales surround the prey along a spiral path simultaneously, and the movement in helix-shaped of the whale can be modeled as follows:

$$XX(t+1) = D' \cdot e^{bl} \cdot \cos(2\pi l) + XX^*(t) \tag{11}$$

$$D' = |XX^*(t) - XX(t)| \tag{12}$$

In equation (11), D' is a coefficient that denotes the distance between $XX(t)$ and $XX^*(t)$ given by equation (12). In addition, b is a constant value that indicates the shape of the logarithmic, and l is a random digit between -1 and 1.

Since the whales encircles the prey and moves along spiral path are conducted simultaneously, it can be described that the whales choose two movements with a 50% probability. As shown in equation (13), where the value of p is a random digit between 0 and 1.

$$XX(t+1) = \begin{cases} XX^*(t) - A \cdot D, & p < 0.5 \\ D' \cdot e^{bl} \cdot \cos(2\pi l) + XX^*(t), & p \geqslant 0.5 \end{cases} \tag{13}$$

Search randomly for prey

The whales can also search the prey randomly in addition to surrounding the prey $XX^*(t)$. The exploitation or exploration of WOA during evolution depends on the parameter A. When $|A| < 1$, the individual updates the position toward the optimal individual to perform exploitation. When $|A| \geqslant 1$, the individual searches randomly in a reasonable space to perform exploration. The search randomly can be expressed by equations (14) and (15).

$$D = |C \cdot XX_{rand} - XX(t)| \tag{14}$$

$$XX(t+1) = XX_{rand}(t) - A \cdot D \tag{15}$$

Where $XX_{rand}(t)$ is a random individual from the population, the meaning of the others symbol in equations (14) and (15) are the same as above mentioned.

The discrete whale optimization algorithm for the NWFSSP

Since the WOA can't directly be employed to resolve production scheduling problem with discrete characters, the DWOA is designed to resolve the NWFSSP in this work. A specific description of the DWOA is given as follows.

Encoding mechanism and initialization

The encoding and the decoding scheme play a critical roles when the algorithm is designed. Since permutation-based job representation is widely adopted in literatures for flow shop scheduling problem, it is employed to represent the individual (whale) of the population.

The diversity and quality of the initial population are essential guarantees for the performance of swarm intelligent optimization algorithm. The NEH[46] and the nearest neighbor (NN)[47] are effective constructive heuristics for generating the initial population. Considering no-wait constraint, the heuristics NEH is replaced by the standard deviation (SD) heuristic (SDH).[26] The difference between the two heuristics is the basis of priority given to jobs. The NEH follows that the highest priority should be given to the job with the longest total processing times on all machines. However, SDH follows that the highest priority should be given to the job with the largest SD because the job with the higher SD will need the longer detention time. So, the NN + SDH constructive heuristics is selected to get the initial population.

The partial schedules of the first two jobs of an individual in the initial population are obtained by NN heuristic, and the remaining n-2 jobs in schedules are generated with the SDH heuristic. The concrete steps of NN + SDH is shown as follows.

Step 1. Executing the NN heuristic
Step 1.1 randomly generate a set $S = \{J_1, J_2, \cdots, J_n\}$ for all jobs, and pick out NP jobs randomly from S to construct another set $F_s = \{F_s^1, F_s^2, \cdots, F_s^{NP}\}$;
Step 1.2 the first job in the jth individual is chosen from $F_s = \{F_s^1, F_s^2, \cdots, F_s^{NP}\}$;
Step 1.3 the second job is employed the NN heuristic to select from S with a minimum detained time to the first job;
Step 1.4 repeat Step1.2 and Step 1.3 NP times, and generate the NP partial scheduling sequences with the first two jobs for initial population.
Step 2. Executing SDH for the remaining n-2 jobs.
Step 2.1 acquire the initial schedule $\pi = \{\pi_1, \pi_2, \cdots, \pi_n\}$ which is arrayed in non-increasing order in terms of the SD value.
Step 2.2 Pick out the two jobs generated by NN heuristic algorithm from π, denoted as π_{rr}. The remaining partial of π contains n-2 jobs, denoted as π_{dd}.
Step 2.3 Evaluate π_{rr} with different jobs orders according to makespan criterion. The better one is as the current partial sequence π_{rr}^*.
Step 2.4 Take out job $\pi_j, j = 1, 2, \cdots, n-2$ from the partial schedule π_{dd} and insert it into all slots of π_{rr}^*. Then, find the best partial scheduling and update π_{rr}^*.
Step 2.5 Repeat execute the Step 2.4 until the new completion schedule π is constructed.
Step 2.6 Repeat Step 2.2 to Step 2.5 NP times for NP partial scheduling sequences to generate the initial population.

The discrete exploitation stage

In WOA, the whale hunts the prey by shrinking the encircling around the prey and bubble net attacking along the spiral way, as the exploitation stage of the algorithm. The individual updates its position according to the optimal individual position (the prey); that is, it evolves through the neighborhood individual. Thus, the quality of the neighboring solution has a heavily impact on the optimization ability of the algorithm. In this section, the crossover operator f_1 is designed as the discrete individual updating mechanism substituted for the individual encircling the prey, and the crossover operator f_2 is designed to substitute for the spiral way updating position. The two operators are selected with a 50% probability. If p is smaller than 0.5, f_1 will be conducted on the current individual and the optimal individual; otherwise, f_2 will be conducted on them, Where p is a random number between 0 and 1. f_1 and f_2 are described as follows.

The two-point crossover (TPX) is employed as crossover operator f_1, and the multiple-point crossover (MPX) is employed as crossover operator f_2, as follows in equation (16).

$$\pi(t+1) = \begin{cases} f_1(\pi(t), \pi^*(t)) & if\, p < 0.5 \\ f_2(\pi(t), \pi^*(t)) & otherwise \end{cases} \tag{16}$$

The detailed steps of the TPX scheme are as follows, and illustrated in Figure 1.

Step 1. Randomly pick two locations (r_1 and r_2);
Step 2. Duplicate the jobs in between r_1 and r_2 from P1 to C2, and remove the jobs in C1 from P1, then, the remaining jobs in P1 are copied to C1 in order;

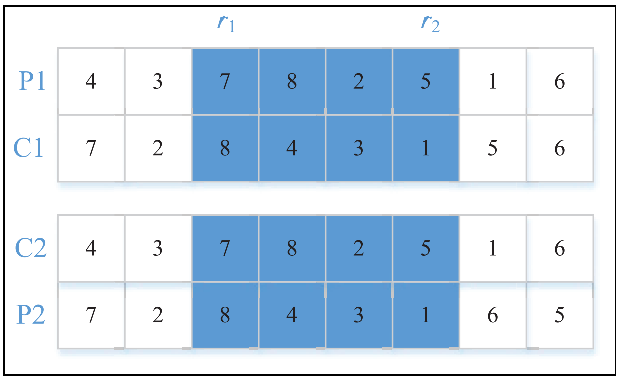

Figure 1. Two-point crossover (TPX) operation.

Step 3. Duplicate the jobs in between r_1 and r_2 from the P2 to C1, and remove the jobs in C2 from P2, then, the remaining jobs in P2 are copied to C2 in order;
Step 4. Choose the better between C1 and C2 according to makespan criterion as the crossover result.

The MPX scheme is depicted in Figure 2, and the steps as follows.

Step 1. Randomly create one set S where only contains 0 and 1.
Step 2. Copy the jobs with the positions corresponding with "1" in set S from P1 to C1 and from P2 to C2.
Step 3. Copy the rest jobs with the position corresponding with "0" in set S from P1 to C2 and from P2 to C1.
Step 4. Choose the better between C1 and C2 according to makespan criterion as the crossover result.

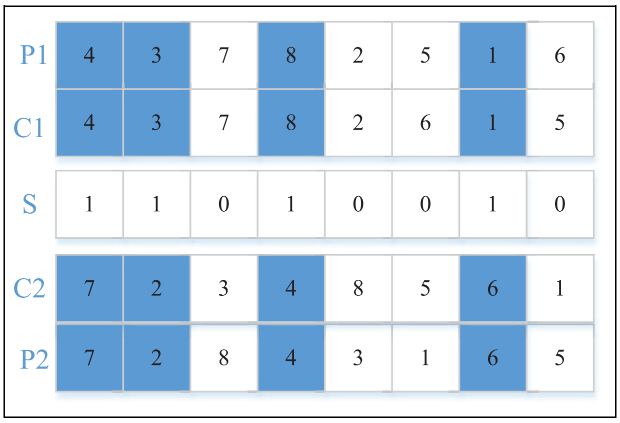

Figure 2. Multiple-point crossover (MPX) operation.

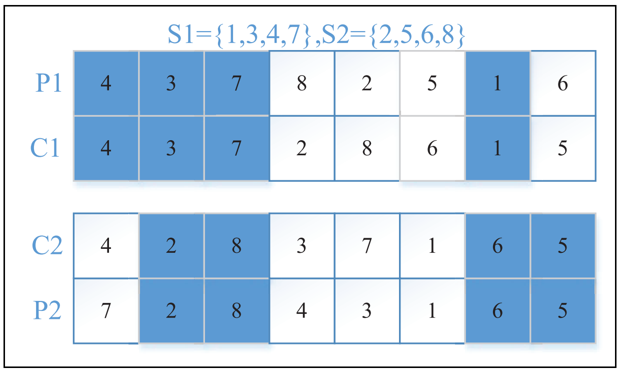

Figure 3. Job-based crossover (JBX) operation.

Step 4. Choose the better between C1 and C2 according to makespan criterion as the crossover result.

The discrete exploration stage

At the exploration stage, the algorithm makes the whale move far away from the prey to anywhere in the feasible solution space for searching the global optimum. Meanwhile, the whale individual updates its position by another individual randomly selected from the population, denoted $\pi_{rand}(t)$. In DWOA, the crossover operator f_3 between $\pi(t)$ and $\pi_{rand}(t)$, which selects the job-based crossover (JBX) operation given as equation (17), is employed for the discrete individual update.

$$\pi(t+1) = f_3(\pi(t), \pi_{rand}(t)) \tag{17}$$

The JBX operator is illustrated in Figure 3, and the steps are as follows.

Step 1. Randomly generate a job permutation S, then randomly choose several jobs from S and put into set S1, the others put into set S2;
Step 2. Copy the jobs belonging to S1 from P1 to C1, and copy the jobs belonging to S2 from P2 to C2;
Step 3. Copy the jobs belonging to S2 from P2 to C1, and copy the jobs belonging to S1 from P1 to C2;

Dynamic adjustment strategy

It is hoped that the algorithm has a better performance at any stage during evolution. That is, it can explore the optimum in an ample space at the early stage, and can exploit the potential space to quickly find out the global optimal solution at last. In the WOA, the transform between exploration and exploitation mainly depends on parameter a, which linearly decreases over the course of evolution, denoted by equation (10). To meet the above requirements, the expression of a is replaced by the non-linear dynamic adjustment expression of pb given by equation (18).

$$pb = pb_{\max} - (pb_{\max} - pb_{\min})\frac{t^2}{t_{\max}^2} \tag{18}$$

Where pb_{max} and $pb_{\min}$ are the upper and lower bound values of pb, respectively.

Promoting local search capability

In order to further improve the exploitation ability of DWOA, the following algorithm1 is conducted on all individuals. It employed the parallel neighborhood search (PNS), and it not only can find the better solutions but also can guarantee the diversity of solutions. The choice of neighborhood structures has a great influence on the performance of PNS. The job's position movement in schedule is usually employed to acquire neighborhood solutions. The insertion and pair swap movements are more effective and universal among the various movement ways used for solving the NWFSSP.[48] Moreover, the double-insert movement is the neighborhood structure removing two consecutive jobs and reinserting them into two adjacent positions. It is also adopted to generate the neighborhood solutions for the NWFSSP and is demonstrated to have a better performance.[19] Furthermore, the effectiveness of the iterated greedy (IG)[48] algorithm also has been demonstrated for production scheduling problems, especially the destruction and reconstruction (DC), as the critical step of IG is the important and effective strategy for searching neighborhood solution. So, the PNS contains four different neighborhood structures: the insertion structure, the pair exchange structure, the double-insert structure, and DC.

One of the four neighborhood structures is randomly selected for each individual of the population, and it is performed the local search to get its neighborhood individuals. The individual left by greedy selection between the original and the neighborhood. The concrete description of PNS is presented in algorithm 1.

Algorithm 1 pseudocode of parallel neighborhood search (PNS)

```
input: the original permutation $\pi$; output: the new permutation $\pi$
Randomly generate an integer $q$ between 1 and 4
  Switch ($q$)
    case 1 Randomly select a job $\pi_j$ from position $j$ in
           permutation $\pi$ and interchange with job $\pi_k$
           ($k \in \{1, 2, \cdots, n\}$ and $j \neq k$) to get the new
           permutation $\pi'$, then select the best permutation
           $\pi''$ from all of $\pi'$
             If $C_{\max}(\pi'') < C_{\max}(\pi)$
               $\pi = \pi''$;
             End If
    case 2 Randomly select a job $\pi_j$ from $\pi$ and reinsert it into
           all of the likely positions except for $j$ to get the new
           permutation $\pi'$, then select the best permutation
           $\pi''$ from all of $\pi'$
             If $C_{\max}(\pi'') < C_{\max}(\pi)$
               $\pi = \pi''$;
             End If
    case 3 Randomly select two adjacent jobs $\pi_j$ and $\pi_{j+1}$ from
           $\pi$ and reinsert them together into all the possible
           two adjacent positions $k$ and $k + 1$ to get the new
           permutation $\pi'$, then select the best permutation
           $\pi''$ from all of $\pi'$
             If $C_{\max}(\pi'') < C_{\max}(\pi)$
               $\pi = \pi''$;
             End If
    case 4 Employ DC operator to $\pi$, and get $\pi'$
             If $C_{\max}(\pi') < C_{\max}(\pi)$
               $\pi = \pi'$;
             End If
  End Switch
Return $\pi$
```

Promoting global search capability

The DWOA is prone to falling into local optimum after multiple iterations because of the neighborhood search mechanism. So, to find the better solution, the promoting global search mechanism is executed to produce a new individual when the optimal individual can't be improved and exceeded the given *limit*. The new individual generated must be far away but not too far away from the old. It has not been updated for many generations and exceeded the given *limit*, but it has gone through many iterations and carried good information. If it is directly discarded, new individuals randomly generate. The efficiency of the algorithm will be decreased. The age variable is chosen to represent the optimum update and set 0. If it can't be further improved, the age will be increased by 1 and compared with *limit*. When age exceeds to the *limit*, the serial neighborhood search (SNS) strategy is performed to it. On the contrary, if it is improved, then the age resets to 0. To find a better individual replaced the old one, the SNS strategy will run *u_max* times in a loop. In this way, the algorithm will jump out of the local optimum and proceed to search after a remarkable solution. The promoting global search mechanism can sufficiently and systematically exploit the neighborhood change. The specific steps of SNS is given in algorithm 2.

Methods to accelerate calculation makespan

In DWOA, the PNS and SNS all contain several neighborhood structures. To judge the neighborhood individual, its makespan needs to be calculated, so, it will expend large amount of computing time. However, the neighborhood structures, the insertion, swap, double-insert and DC neighborhoods, in PNS and SNS strategies only changed the part of the original permutation. When calculating the makespan of the neighborhood individual, considering only the impact of the changed part will greatly decrease lots of calculation time. The accelerate methods of insertion and swap have been described in detail,[11,26] and the accelerate methods of the first three mentioned above is given in increment.[19] Moreover, the specific description of the accelerate methods for above-mentioned structures except for DC has presented by Zhang et al.[26] The DC performs to generate the neighborhood solution, where the computing time mainly spends on the insert operation. Due to space limitation, specific description is omitted here.

The discrete whale optimization algorithm(DWOA)

The several important components of DWOA have been given a minute description above, and the complete

Algorithm 2 pseudocode of the serial neighborhood search (SNS)

```
input: the original permutation $\pi$, the maximum times of the cycle $u\_max$ ; output : the new permutation $\pi$
  $u = 1$;
While $u \leqslant u\_max$
  For $q=1$ to 3 do
    Switch ($q$)
      case 1 Randomly select a job $\pi_j$ from $\pi$ and swap with job $\pi_k$ ($k \in \{1, 2, \cdots, n\}$ and $j \neq k$) to get the new permutation $\pi'$, then select the best permutation $\pi''$ from all of $\pi'$
        If $C_{\max}(\pi'') < C_{\max}(\pi)$ then
          $\pi = \pi''$, $q = 1$;
        Else
          $q = q + 1$;
        End If
      case 2 Randomly select a job $\pi_j$ from $\pi$ and reinsert it into all of the likely positions except for $j$ to get the new permutation $\pi'$, then select the best permutation $\pi''$ from all of $\pi'$
        If $C_{\max}(\pi'') < C_{\max}(\pi)$ then
          $\pi = \pi''$, $q = 2$;
        Else
          $q = q + 1$;
        End If
      case 3 Randomly select two adjacent jobs $\pi_j$ and $\pi_{j+1}$ from $\pi$ and reinsert them together into all the possible two adjacent positions $k$ and $k + 1$ to get the new permutation $\pi'$, then select the best permutation $\pi''$ from all of $\pi'$
        If $C_{\max}(\pi'') < C_{\max}(\pi)$ then
          q=3; $\pi = \pi''$;
        Else
          $q = q + 1$;
        End If
    End Switch
  End for
  $u = u + 1$;
End While
Return $\pi$
```

algorithm framework of DWOA is illustrated by the flowchart, which is shown in Figure 4

The experiment and results analysis

Experiment settings

The following simulation experiments are executed on the MATLAB platform. The operating system of test computer is Windows 10, and the processor is the four-core Intel® Core™ i5-9600KF. In addition, it has 3.7 GHz CPU and 16.00 GB RAM.

In this section, the test instances contain 21 benchmarks by Reeves[49] and 120 benchmarks provided by Taillard[50] with different sizes, respectively.

Parameter setting

The performance of the intelligent optimization algorithm is heavily affected by parameters involved,[26] so it is crucial to finding a better set of parameters by executing the parameter calibration experiments. The DWOA has five main parameters, which are *NP* (population size), $t_{\max}$ (the maximum evolutionary algebras), *limit* (the given old enough age), *u_max* (maximum cycle of SNS) and *d* (the degree of destruction in DC), needed to set by preliminary testing. After the preliminary experiment, the ranges of the above five parameters are set as $\{n/2, n/3, n/4, n/5\}$, {300, 400, 500, 600},{5, 10, 15, 20}, $\{n/2, n/3, n/4, n/5\}$, and {6, 9, 12, 15}, respectively.

Five parameters are set to five factors, and each has four levels designed by orthogonal experiment to calibrate the parameters of the DWOA (Table 1). There are $4^5 = 1024$ parameters combination needed to be tested in the factorial experiments. Nevertheless, it requires only $4^2 = 16$ according to the orthogonal experiment. The amount of calculation is significantly reduced when the quality of parameter selection is not compromised.

Table 2 provides the orthogonal table L_{16} (4^5), which contains sixteen parameter combinations. To select the parameters for the DWOA that have good optimization effects on different scales instances, we test DWOA on nine instances with different scale, namely, Ta01, Ta11,Ta21 to Ta81, and each one is tested 10 times for each parameter combination. The average relative percentage deviation (ARPD) given by equation (19), is used to evaluate the quality of the experiments. The mean value of all instances (9×10) for the parameter combination are placed in Table 2 as the basis for

Table 1. Orthogonal experimental factors/level table.

Parameters	Levels			
	1	2	3	4
NP	*n*/2	*n*/3	*n*/4	*n*/5
$t_{\max}$	300	400	500	600
limit	5	10	15	20
u_max	*n*/2	*n*/3	*n*/4	*n*/5
d	6	9	12	15

Table 2. Orthogonal table of parameters $L_{16}(4^5)$ and average results.

Test	Factor					ARPD
	NP	$t_{\max}$	limit	*u_max*	*d*	
1	n/2(1)	300(1)	5(1)	n/2(1)	6(1)	1.81
2	n/2(1)	400(2)	10(2)	n/3(2)	9(2)	1.52
3	n/2(1)	500(3)	15(3)	n/4(3)	12(3)	1.28
4	n/2(1)	600(4)	20(4)	n/5(4)	15(4)	1.26
5	n/3(2)	300(1)	10(2)	n/4(3)	15(4)	1.49
6	n/3(2)	400(2)	5(1)	n/5(4)	12(3)	1.43
7	n/3(2)	500(3)	20(4)	n/2(1)	9(2)	1.53
8	n/3(2)	600(4)	15(3)	n/3(2)	6(1)	1.66
9	n/4(3)	300(1)	15(3)	n/5(4)	9(2)	1.64
10	n/4(3)	400(2)	20(4)	n/4(3)	6(1)	1.83
11	n/4(3)	500(3)	5(1)	n/3(2)	15(4)	1.44
12	n/4(3)	600(4)	10(2)	n/2(1)	12(3)	1.43
13	n/5(4)	300(1)	20(4)	n/3(2)	12(3)	1.67
14	n/5(4)	400(2)	15(3)	n/2(1)	15(4)	1.61
15	n/5(4)	500(3)	10(2)	n/5(4)	6(1)	1.95
16	n/5(4)	600(4)	5(1)	n/4(3)	9(2)	1.55

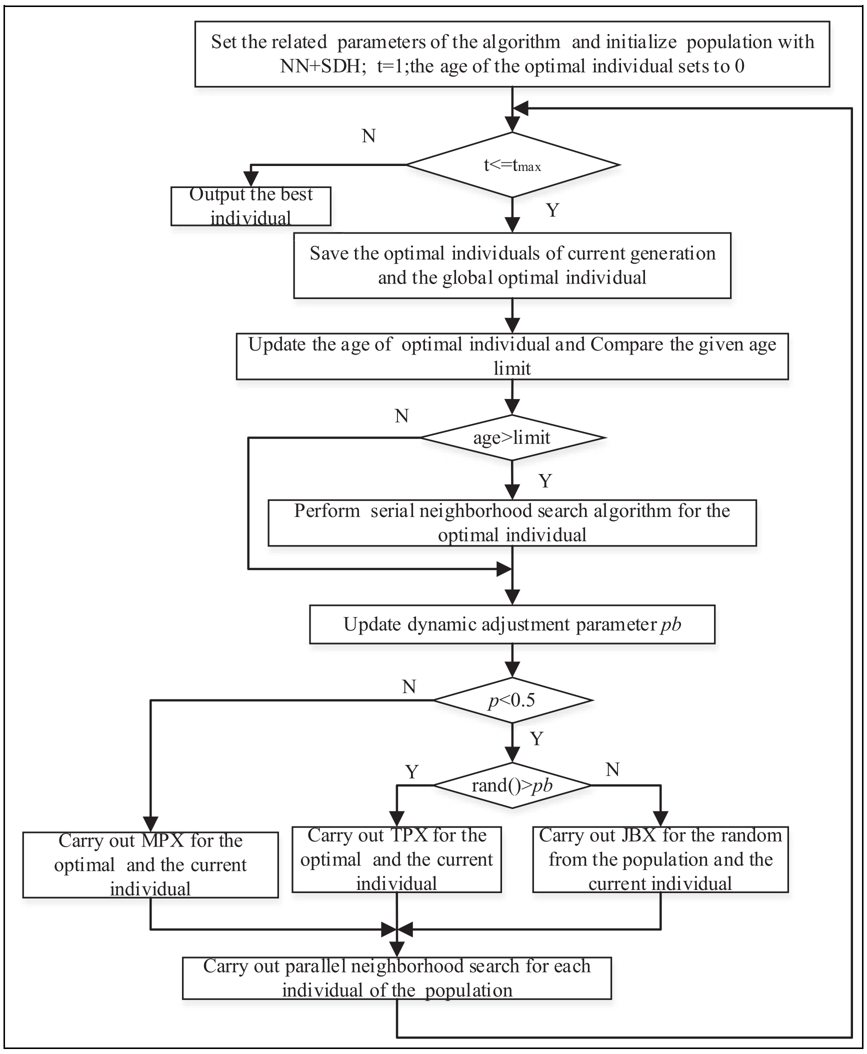
Set the related parameters of the algorithm and initialize population with NN+SDH; t=1;the age of the optimal individual sets to 0
t<=tmax
N
Y
Output the best individual
Save the optimal individuals of current generation and the global optimal individual
Update the age of optimal individual and Compare the given age limit
age>limit
N
Y
Perform serial neighborhood search algorithm for the optimal individual
Update dynamic adjustment parameter pb
p<0.5
N
Y
rand()>pb
Y
N
Carry out MPX for the optimal and the current individual
Carry out TPX for the optimal and the current individual
Carry out JBX for the random from the population and the current individual
Carry out parallel neighborhood search for each individual of the population

Figure 4. The flowchart of DWOA.

Table 3. Mean response values.

Level	Parameters				
	NP	$t_{\max}$	limit	*u_max*	*d*
1	**1.467**	1.652	1.556	1.592	1.814
2	1.528	1.595	1.598	1.572	1.557
3	1.583	1.551	**1.548**	**1.537**	1.451
4	1.693	**1.474**	1.570	1.570	**1.450**
SD	0.0831	0.0650	0.0190	0.0197	0.1485
rank	2	3	5	4	1

The bold entries in each column (for a certain parameter) indicate the minimum mean response.

selecting parameters. The mean response values is list in Table 3, where g_i is the average value of four groups experiments at level i corresponding to a certain parameter. The smallest g_i values for each parameter mean that selecting the parameter at level i can make DWOA have better performance, which is given in bold. The standard deviation (SD) values calculated by equation (20) and made appropriate modifications to the meaning of parameters are provided on the penultimate line. The rank according to descending order of SD are listed in the last row. The larger SD values have greatly influenced on DWOA. On the contrary, the smaller SD values have slight impact on the optimization. So, the influence of parameters on DWOA is d, NP, $t_{\max}$, u_max $limit$ in descending order. The parameter factor levels curve is depicted in Figure 5. As previously analyzed, Table 3 and Figure 5 show that the DWOA has obviously better optimization when NP, $t_{\max}$, $limit$, u_max and d are set to $n/2$, 600, 15, $n/4$ and 15 respectively. So, the parameter combination in DWOA is suggested in the latter experiments.

$$ARPD = \frac{1}{N}\sum_{i=1}^{N}\frac{C_i - C_{opt}}{C_{opt}}\times 100 \qquad (19)$$

Where

C_{opt}: The best makespan.

C_i: The makespan of the ith experiment executing a specific algorithm

N: The number of run times (N is set to 10 in this section and the following compared experiments).

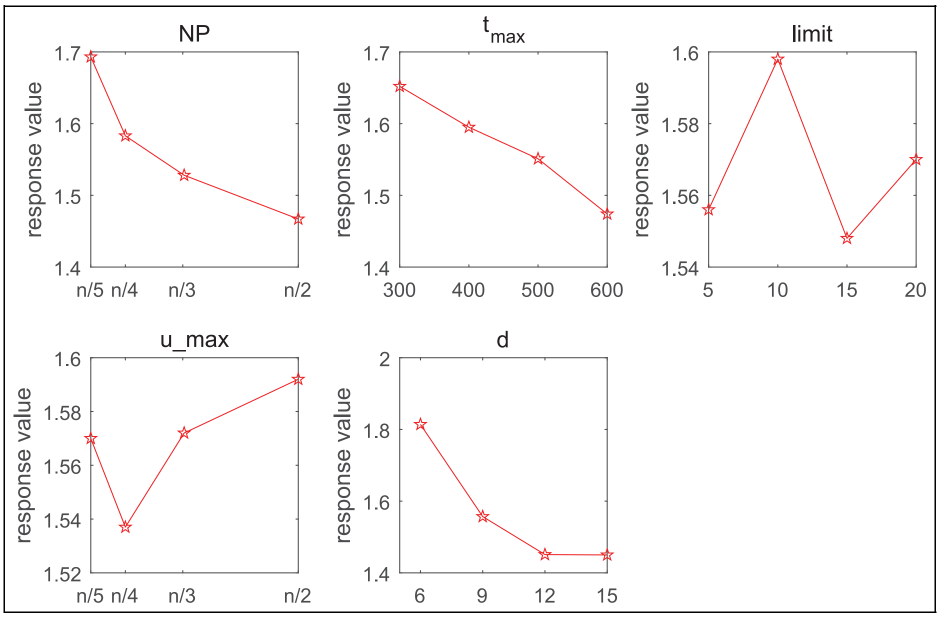

Figure 5. The curves of the parameters factor level.

Table 4. Comparison the mean of initial solution with three heuristics algorithms.

n × m	NEH	SDH	NN + SDH
20 × 5	1539.6	1541.4	**1537.84**
20 × 10	2056.4	2062.8	**2052.02**
20 × 20	3066.1	3079.8	**3062.86**
50 × 5	3525.3	3477.6	**3472.99**
50 × 10	4527.5	4504.7	**4500.27**
50 × 20	6234.6	6163.6	**6163.55**
100 × 5	6710.9	6684.8	**6684.75**
100 × 10	8537.4	8511.2	**8510.1**
100 × 20	**11215**	11269	11231.89
200 × 10	16311.7	16277.7	**16274.65**
200 × 20	21147.3	21056.4	**21040.14**
500 × 20	49705.4	49688.6	**49655.33**

Bolded black values indicate superiority over other comparison heuristics algorithms in the quality of the initial solution.

Compared with different algorithms, the algorithm that can obtain smaller APRD value is better algorithm.

$$SD = \sqrt{\frac{\sum_{i=1}^{N} (C_i - C_{AVG})}{N}} \tag{20}$$

Where C_{AVG} denotes the average makespan value of N times independent experiments. The smaller SD is, the stronger robustness of the algorithm.

Effectiveness analysis of improvement schemes

In this section, the experiment contains initialization algorithm performance and the variants of DWOA tests. The performance the initialization algorithm NN + SDH will be compared with NEH and SDH by testing the instances Ta01 to Ta120 by Talliard, and the makespan generated by them are listed in Table 4. For the sake of fairness, NN + SDH is independently tested for ten times and averaged. It can be seen that the NN + SDH can generate better initial solution because of the adaptability of NN and the effectiveness of SDH. NN + SDH can generate many initial solutions with high-quality, but the NEH and SDH are only generate one initial solution and the other initial solutions will be generated randomly.

The effectiveness of DWOA algorithm improvement mechanism will be tested by comparing the different variants of the DWOA. In the variants DWOA_cos and DWOA_sin, the expression of the parameter *pb* in DWOA is replaced by equations (21) and (22), respectively. The variants DWOA_nls and DWOA_nr are the DWOA without the local search and without promoting global search scheme respectively. Each considered algorithm is executed 10 independent replications for all instances. Furthermore, the experiment results obtained by the DWOA and the variants are listed in Table 5 as three indicators BRPD, ARPD and SD, Where the lower bound C_{opt} found in the literature.[23] In addition, time consumption is also considered and listed in Table 5.

$$pb = pb_{\max} - (pb_{\max} - pb_{\min}) \sin \frac{t\pi}{2t_{\max}} \tag{21}$$

$$pb = pb_{\max} - (pb_{\max} - pb_{\min}) \cos \frac{t\pi}{2t_{\max}} \tag{22}$$

$$BRPD = \frac{C_{\min} - C_{opt}}{C_{opt}} \times 100 \tag{23}$$

For the makespan criterion, $C_{\min}$ is the minimum makespan value in 10 independent runs for the relevant instance.

Table 5. Computational results of DWOA and the variants.

n × m	DWOA_nr				DWOA_nls				DWOA_sin				DWOA_cos				DWOA			
	BRPD	ARPD	SD	time	BRPD	ARPD	SD	time	BRPD	ARPD	SD	time	BRPD	ARPD	SD	time	BRPD	ARPD	SD	time
20 × 5	0.03	0.19	1.89	1.9	0.50	1.25	8.74	**1**	0.04	0.18	1.34	2.6	**0.01**	0.17	1.74	2.6	**0.01**	**0.15**	**1.33**	2.6
20 × 10	0.06	**0.18**	2.60	3.4	0.64	1.44	12.50	**1.9**	0.03	**0.18**	2.84	4.7	**0.00**	**0.18**	2.49	4.7	**0.00**	**0.18**	2.21	4.7
20 × 20	**0.03**	0.07	2.21	6.5	0.28	1.12	21.84	**3.5**	**0.03**	0.06	1.62	9.1	**0.03**	0.06	1.38	9	**0.03**	**0.05**	**0.88**	9.1
50 × 5	0.85	1.50	12.87	13.8	2.00	2.99	19.97	**5.4**	0.94	1.52	13.68	17.1	1.15	1.70	12.42	17.3	**0.79**	**1.44**	**12.39**	17.5
50 × 10	0.65	1.44	13.47	26.6	1.59	2.51	23.83	**11**	0.63	1.14	**12.02**	34.1	0.67	1.21	14.10	33.7	**0.54**	**1.06**	12.88	34.4
50 × 20	0.44	0.89	17.46	52.9	1.52	2.31	28.77	**21.9**	0.49	0.94	**15.43**	68.2	0.51	0.90	17.89	66.5	**0.40**	**0.85**	16.69	68.9
100 × 5	2.20	2.71	21.74	62	3.37	4.12	28.04	**20.6**	2.49	3.22	25.65	71.8	2.81	3.34	24.32	73.1	**2.13**	**2.70**	**19.03**	74.6
100 × 10	1.50	1.97	22.67	118.8	2.57	3.18	27.91	**39.4**	1.71	2.22	22.96	137.4	1.84	2.32	25.94	140.8	**1.44**	**1.90**	**21.54**	144.4
100 × 20	1.44	1.84	29.57	232.2	2.46	3.03	32.78	**84.7**	1.63	2.08	29.88	270.2	1.75	2.23	**29.36**	271	**1.26**	**1.71**	31.37	290.2
200 × 10	2.98	3.32	31.07	497.4	3.94	4.39	41.91	**161.1**	3.42	3.83	43.39	577.7	3.78	3.90	**13.36**	574.2	**2.85**	**3.23**	32.19	588.6
200 × 20	2.46	2.76	**33.46**	934.8	3.31	3.74	52.92	**321.2**	3.05	3.63	61.74	1140.2	2.77	3.59	69.71	1085.3	**2.37**	**2.73**	51.08	1101.3
500 × 20	3.95	4.14	84.12	7340.8	4.506	4.96	82.86	**1798.1**	4.52	4.86	94.35	8734.6	4.43	4.76	90.65	8844.2	**3.88**	**4.11**	**68.34**	8454.7
Average	1.38	1.75	22.76	774.26	2.22	2.92	31.84	**205.82**	1.58	1.99	27.08	922.31	1.65	2.03	25.28	926.87	**1.31**	**1.68**	**22.49**	899.25

Bolded black values indicate superiority over other variants of the DWOA.

From Table 5, it is obviously that the DWOA has better optimization effect compared with other variants. Specifically, the DWOA significantly outperforms DWOA_nls, and it can illustrate that the local search can greatly improve the performance with more significant time cost. In addition, the DWOA is slightly superior to DWOA_nr, and the reason may be that the promoting global search scheme in DWOA can improve the exploration ability of the algorithm and continue to search the potential space. Since the scheme is conducted on the optimal individual, the improvement effect is not particularly obvious. Moreover, the performance of DWOA_sin and DWOA_cos are slightly worse than DWOA_nr but significantly worse than DWOA, which explains that the expression of *pb* has a greater influence on the performance. According to the above analysis, the various optimization mechanisms can effectively improve the optimization ability of the proposed DWOA.

Results and comparisons for Reeves's and Taillard's benchmark

The designed DWOA is compared with four existing algorithms, DPSO$_{\mathrm{VND}}$,[11] IIGA,[18] TMMIG[19] and DWWO[23] which are used to resolve the same problem NWFSSP, and compared with IWOA[44] and DGWO[51] at the same. Among the compared algorithms, the DPSO$_{\mathrm{VND}}$, DWWO, IWOA, DGWO and DWOA are population-based algorithm and job-permutation-based representation. However, the algorithms IIGA and TMMIG are adaptable algorithms, and the comparisons indicates the difference between two kinds of algorithm above-mentioned. For fairness, the compared algorithms are re-executed as all the flow and parameters setting given in the original paper. As we know that there are no DGWO and IWOA to solve the NWFSSP problem in the literature, the algorithm which is used to solve the production scheduling problem in the literature, is appropriately modified to compare with the DWOA algorithm. The NN + SDH is also employed to initialize the population for DGWO and IWOA. Meanwhile, all the algorithms are evolved the same iterations, and each algorithm is carried out independently 10 times on each instance of Reeves's and Taillard's benchmarks. The computational results in following Tables 6 to 8 contain four indicators: ARPD, BRPD, SD and time. They are listed in Table 7 for Reeves's and Tables 6 and 8 for Taillard's instances, where the best values for the same optimization index are given in boldface.

For small-scale instances, it is obvious from Tables 6 and 7 that the results generated by IIGA, TMMIIG and DWOA are superior to the results by other algorithms. Specifically, the results generated by IIGA are mainly better than the other compared algorithms at ARPD and SD aspect; and the TMMIG algorithm has a better performance in BRPD respect. However, the

Table 6. Results comparison based on Reeve's benchmarks.

n × m	IIGA			TMIIG			DPSO$_{VND}$			DWWO			IWOA			DGWO			DWOA		
	BRPD	ARPD	SD	BRPD	ARPD	SD	BRPD	ARPD	SD	BRPD	ARPD	SD	BRPD	ARPD	SD	BRPD	ARPD	SD	BRPD	ARPD	SD
20 × 5	0.00	**0.05**	**0.89**	−0.48	0.53	8.55	0.80	1.50	5.71	0.01	0.13	1.60	0.72	1.89	14.03	0.75	1.61	8.60	0.00	0.13	1.33
20 × 10	0.00	**0.06**	**1.21**	−0.55	0.50	11.03	0.67	1.28	7.78	0.02	0.13	1.81	1.31	2.17	13.66	0.74	1.62	12.82	0.00	0.18	2.65
20 × 20	0.03	**0.03**	**0.29**	−0.70	0.20	17.42	0.70	1.03	6.59	0.03	0.08	2.75	1.13	1.84	19.08	0.65	1.26	14.20	0.03	0.05	0.88
50 × 5	1.36	1.76	**6.40**	2.18	2.94	15.09	4.03	4.62	8.98	**0.68**	**1.04**	6.64	3.27	4.30	17.61	3.31	4.39	22.20	0.79	1.44	11.85
50 × 10	0.60	0.92	**7.13**	1.42	2.42	22.75	3.04	3.42	8.19	**0.22**	**0.53**	7.80	2.79	3.50	23.95	2.84	3.72	24.39	0.54	1.06	12.02
50 × 20	0.50	0.81	11.61	1.29	2.09	26.16	2.31	2.68	**10.14**	**0.15**	**0.47**	10.66	2.30	2.94	23.33	2.35	3.33	35.02	0.39	0.85	15.43
100 × 5	3.13	3.50	14.68	3.57	4.20	23.79	7.03	7.19	**6.19**	3.13	3.84	22.10	5.21	5.91	24.44	5.03	6.06	36.24	**2.13**	**2.70**	23.32
100 × 10	1.79	2.28	20.95	2.83	3.33	22.79	4.94	5.11	**6.97**	1.47	**1.83**	17.69	3.97	4.61	31.97	4.85	5.48	32.20	**1.44**	1.90	18.07
100 × 20	1.60	1.96	29.04	2.58	3.31	39.77	4.19	4.38	**12.29**	**0.94**	**1.23**	21.40	3.72	4.41	45.76	4.44	5.07	34.44	1.26	1.71	29.36
200 × 10	3.07	3.68	43.30	4.31	4.70	29.07	6.53	6.80	**14.15**	3.93	4.33	47.36	4.69	5.03	25.32	6.19	6.54	36.46	**2.81**	**3.22**	28.13
200 × 20	2.97	3.19	23.17	3.86	4.11	33.65	5.21	5.25	**4.72**	2.90	3.21	37.95	3.96	4.66	65.51	5.08	5.89	64.91	**2.29**	**2.71**	59.36
500 × 20	4.72	5.06	93.75	5.57	5.78	57.01	6.74	6.79	**7.42**	5.99	6.23	102.1	5.83	6.36	85.27	6.04	6.57	89.42	**3.88**	**4.09**	81.12
Average	1.65	1.94	21.04	2.16	2.84	25.59	3.85	4.17	**8.26**	1.62	1.92	23.32	3.24	3.97	32.49	3.52	4.30	34.24	**1.30**	**1.67**	25.65

Bolded black values indicate superiority over other comparison algorithms.

proposed DWOA has similar results and has less time cost compared to TMMIG and DPSO$_{VND}$. For medium-scale instances, the DWWO has a better performance than other compared algorithms. Even though DWOA can generate slightly worse solutions than DWWO, it should be noted that it will take considerably less computing time. However, for large-scale instances (Ta91-Ta120), Table 6 shows that the results obtained by DWOA are significantly exceeds to those of the other compared algorithms in accordance with BRPD and ARPD.

The overall results for all Tailliard's instances are given at the last row of Table 6. The average BRPD and ARPD obtained by DWOA are 1.3 and 1.67, which smaller than the corresponding values of 1.62 and1.92 launched in DWWO algorithm and 1.65 and 1.94 obtained by IIGA, respectively. Meanwhile, those far exceeds the remaining four algorithms (TMIIG, DPSO$_{VND}$, IWOA and DGWO) according to BRPD and ARPD. The average SD acquired by DPSO$_{VND}$ algorithm correspond to 8.26 is less than the SD generated by other six compared algorithms. Even though DPSO$_{VND}$ algorithm has stronger robustness, it should be noted that it can't find the better makespan values. As for time index, the times consumption by seven compared algorithms abovementioned are listed in Table 8. From it, time required of executing TMMIG algorithm is the smallest of the compared algorithms. However, the time-cost of DWOA is only much than that of TMMIG but magnificent less than time required of executing the algorithms DWWO, IWOA and DGWO. The diversity of neighborhood structure of DWOA can promote the global search capability, and the strategy of randomly selecting neighborhood structure can increase the robustness.

The Gantt charts of Rec35 and Ta29 are shown in Figures 6 and 7. The schedule is provided by carrying out the DWOA. The convergence curves of compared algorithms on Ta69, Ta79, Ta100 and Ta110 are shown in Figures 8 to 11, respectively. Regardless of the size of instance, the convergence curves of DWOA has the smallest convergence value among the compared algorithms. Meanwhile, the DWOA can converge rapidly compared with others algorithms at the early stage of iteration.

Conclusion

In this paper, the DWOA is designed to address the NWFSSP with the objective makespan. Firstly, the permutation job-based encoding scheme is adopted to stand for the individual, and the NN + SDH constructive heuristics is employed to initialize population. Secondly, the individual updating mechanism in DWOA is designed. Specifically, The TPX operation is adopted to simulate the shrinking encircling the prey mechanism of the whales; the MPX operation is employed to simulate the spiral updating mechanism;

Table 7. Results comparison based on Taillard's benchmarks.

Problem	n,m	C_{opt}	IIGA			time	TMIIG			time	$DPSO_{VND}$			time	DWWO			time	DWOA			time
			BRPD	ARPD	SD		BRPD	ARPD	SD		BRPD	ARPD	SD		BRPD	ARPD	SD		BRPD	ARPD	SD	
Rec01	20,5	1526	0.00	**0.00**	**0.00**	24.9	−0.66	0.45	12.77	4.22	0.46	0.46	**0.00**	5.0	0.00	0.16	2.33	21.7	0.00	0.10	2.33	**2.6**
Rec03	20,5	1361	**0.00**	**0.00**	**0.00**	24.7	**0.00**	1.79	17.96	4.1	1.25	1.83	6.46	5.0	**0.00**	**0.00**	**0.00**	21.7	**0.00**	**0.00**	**0.00**	**2.6**
Rec05	20,5	1511	0.00	0.03	**0.92**	25.7	−1.26	−0.62	7.14	4.1	0.60	1.16	7.51	5.0	0.00	0.08	1.72	21.5	0.00	0.10	2.24	**2.6**
Rec07	20,10	2042	0.00	0.00	**0.00**	44.3	−2.11	−0.51	21.20	7.3	0.05	0.05	**0.00**	8.9	0.00	0.04	0.40	40.7	0.00	0.00	**0.00**	**4.9**
Rec09	20,10	2042	0.00	**0.00**	**0.30**	46.3	−1.52	0.22	18.88	7.2	0.05	0.66	8.18	8.9	0.00	0.11	1.10	40.3	0.00	0.01	0.45	**4.7**
Rec11	20,10	1881	**0.00**	**0.00**	**0.00**	46.6	0.11	0.84	6.81	7.2	0.58	0.58	**0.00**	8.9	**0.00**	0.14	4.12	40.4	**0.00**	**0.00**	**0.00**	**4.8**
Rec13	20,15	2545	**0.00**	**0.00**	**0.00**	66.8	0.12	1.05	15.02	10.4	0.55	0.78	3.49	12.9	**0.00**	0.23	2.98	59.8	**0.00**	0.10	3.42	**7**
Rec15	20,15	2529	0.00	0.00	**0.00**	63.4	−2.93	−1.35	19.47	10.5	0.00	0.17	4.98	12.8	0.00	0.00	**0.00**	59.4	0.00	0.00	**0.00**	**7.1**
Rec17	20,15	2587	0.00	**0.00**	**0.00**	73.3	−0.15	0.07	3.70	10.3	0.23	0.90	9.73	12.9	0.00	**0.00**	**0.00**	59.4	0.00	**0.00**	**0.00**	**7.0**
Rec19	30,10	2850	**0.00**	**0.22**	6.52	126.1	0.81	1.65	14.59	**11.0**	2.25	2.78	7.51	17.4	0.39	0.48	**0.90**	122.0	**0.00**	0.27	4.78	11.9
Rec21	30,10	2821	0.18	**0.29**	2.93	125.1	0.25	0.85	8.76	**11.0**	1.24	1.71	8.39	17.3	0.21	0.37	**2.80**	121.1	**0.00**	0.35	3.43	11.9
Rec23	30,10	2700	**0.00**	0.23	4.49	121.4	0.85	1.53	14.53	**11.0**	1.22	2.09	7.77	17.2	**0.00**	**0.07**	**2.10**	121.6	**0.00**	0.31	9.04	11.8
Rec25	30,15	3593	**0.00**	0.08	**2.24**	173.4	1.31	1.80	10.92	**15.9**	0.92	1.19	8.10	25.4	**0.00**	**0.06**	3.96	180.6	**0.00**	0.09	5.04	17.4
Rec27	30,15	3431	0.09	0.45	6.96	179.3	0.29	0.88	11.32	**16.0**	2.07	2.59	10.07	25.6	**0.00**	0.40	12.19	179.8	**0.00**	**0.25**	**6.39**	17.6
Rec29	30,15	3291	**0.00**	**0.03**	**3.00**	172.1	0.27	0.70	4.87	**15.9**	0.79	1.06	9.00	25.5	**0.00**	0.54	9.62	180	**0.00**	0.57	5.10	17.9
Rec31	50,10	4307	0.67	1.06	9.17	341.2	0.91	2.19	37.66	**18.6**	2.14	3.26	23.80	44.5	**0.37**	**0.54**	**3.85**	583.7	0.44	0.84	10.34	34.3
Rec33	50,10	4424	0.95	1.49	12.13	336.6	2.10	3.11	24.94	**18.6**	3.73	4.17	10.02	44.3	**0.66**	**0.92**	**6.41**	581.4	0.84	0.93	9.04	34.1
Rec35	50,10	4397	0.73	1.00	7.53	313.3	1.77	2.31	17.95	**18.5**	3.27	3.51	**4.49**	44.3	**0.13**	0.39	11.72	580.3	**0.13**	**0.38**	9.88	34.4
Rec37	75,20	8007	1.31	1.62	17.61	1346	3.03	3.37	18.72	**55.8**	3.56	3.69	**6.25**	204.9	**0.85**	**1.00**	16.77	3814	0.89	1.02	19.68	162.2
Rec39	75,20	8419	1.38	1.60	15.08	1378	2.84	4.50	53.08	**55.8**	2.80	2.95	**9.78**	204.4	**0.27**	**0.73**	18.61	3807	0.28	0.82	13.23	151.4
Rec41	75,20	8437	1.20	1.50	14.82	1277	1.86	2.77	47.57	**55.9**	3.37	3.59	**7.45**	202.6	**0.47**	**0.74**	12.60	3793	0.53	0.75	14.61	158.3
Average			0.31	0.46	**4.94**	300.2	0.38	1.31	18.47	**17.6**	1.48	1.87	7.28	45.4	0.16	**0.33**	5.44	687.1	**0.15**	**0.33**	5.67	33.6

Bolded black values indicate superiority over other comparison algorithms.

Table 8. Times consumption comparison based on Taillard's benchmarks.

n × m	IIGA	TMIIG	DPSO$_{VND}$	DWWO	IWOA	DGWO	DWOA
20 × 5	19.3	3.1	3.9	22.37	14.63	102.73	**2.6**
20 × 10	47.7	7.4	9.1	41.36	28.38	198.15	**4.9**
20 × 20	88.6	13.4	16.6	78.43	55.80	397.85	**9.3**
50 × 5	189.6	**9.7**	22.3	306.70	178.05	390.28	17.8
50 × 10	334.5	**18.7**	44.9	590.97	374.69	791.70	35.4
50 × 20	635.5	**36.9**	91	1171.6	895.77	1862.80	66.5
100 × 5	790.9	**20.2**	93.2	2579	1196.95	1484.74	75
100 × 10	1268.8	**39.4**	191.5	4954.1	3030.33	3603.19	151.1
100 × 20	2306.5	**77.6**	386.1	9596.8	5325.86	6184.31	286.8
200 × 10	4714	**77**	890.1	29726	17864.05	15036.88	648.7
200 × 20	7996.7	**150.8**	1785.2	55286	39781.42	22726.05	1110.6
500 × 20	51658	**424.8**	18002	535668	828693.6	977985.63	8314.3
Average	5837.5	**73.3**	1794.7	53335	74786.63	85897.03	968.8

Bolded black values indicate superiority over other comparison algorithms in term of the times consumption.

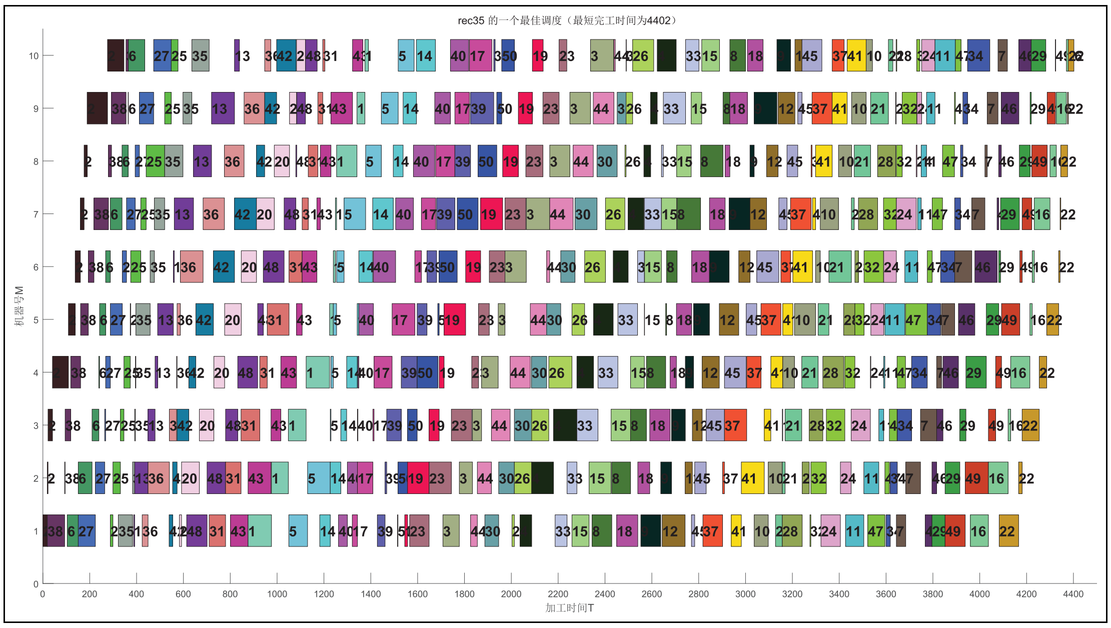

Figure 6. The Gantt chart of Rec35.

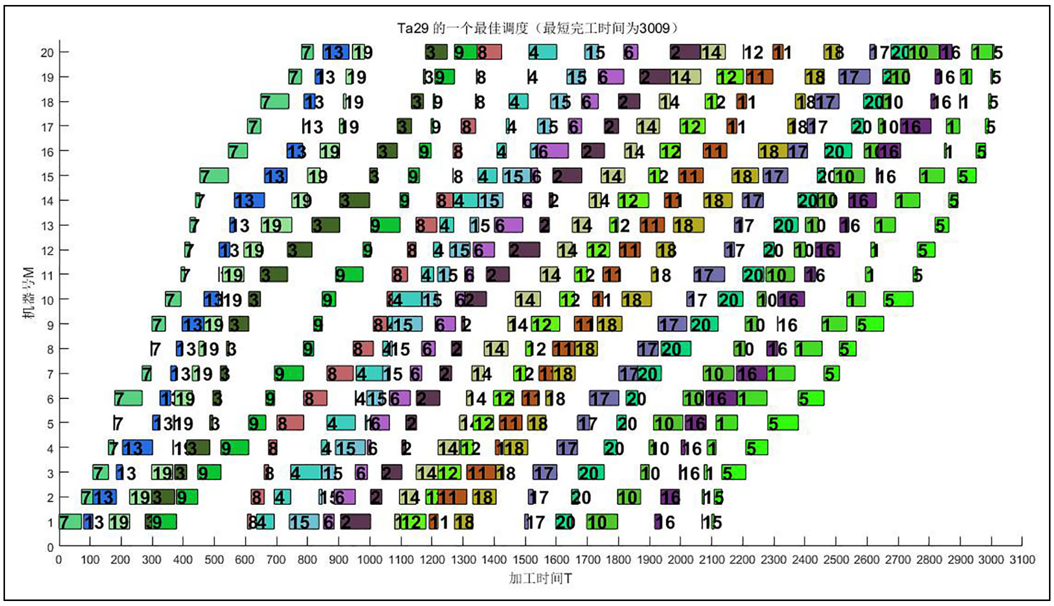

Figure 7. The Gantt chart of Ta29.

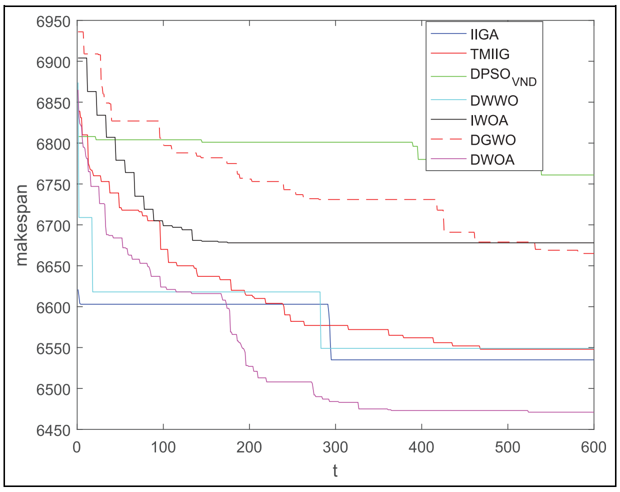

Figure 8. Convergence curves comparison for the instance of Ta69.

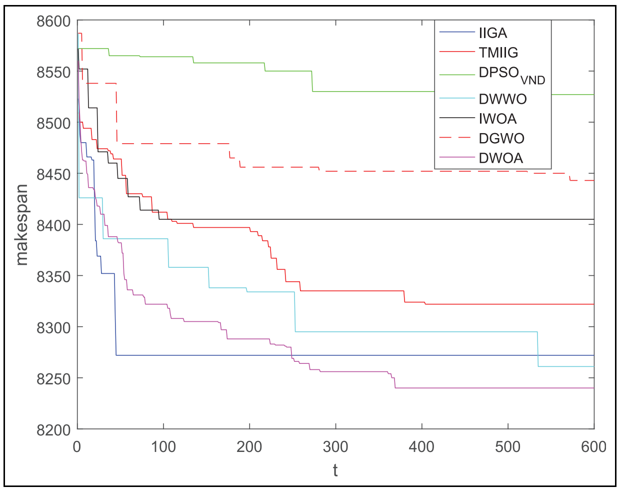

Figure 9. Convergence curves comparison for the instance of Ta79.

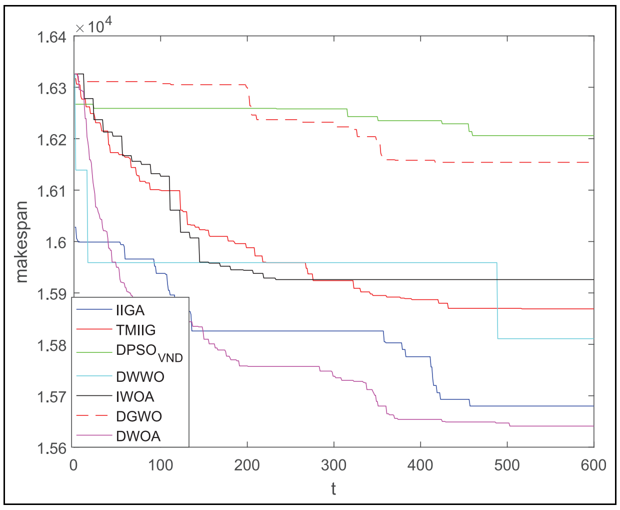

Figure 10. Convergence curves comparison for the instance of Ta100.

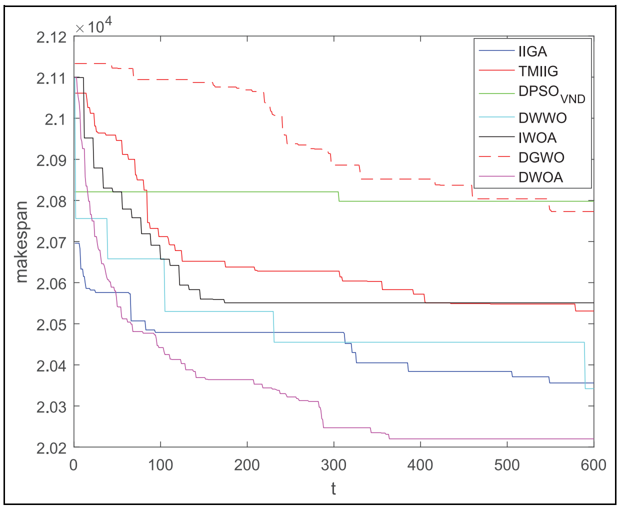

Figure 11. Convergence curves comparison for the instance of Ta110.

the JBX operation is applied to simulate the process of randomly searching for prey. Thirdly, a nonlinear function is selected as adjustment mechanism of exploitation and exploration of the DWOA. Fourthly, the local search and promoting global search ability schemes are designed to further enhance the performance of DWOA. In the end, the performance of DWOA is tested by experiments. The simulation experimental results show that the DWOA can solve the NWFSSP effectively, in particular, it has the remarkable performance when processes the larger scale instances of Ta. In the future, the energy-efficient NWFSSP and the distributed NWFSSP will be studied and the more practical constraints will be taken into account.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work is supported by the National Nature Science Foundation of China (61973120, 61973209, and 62273132) and Henan Provincial Science and Technology Research Project (202102110282, 222102110095).

ORCID iD

Sujun Zhang https://orcid.org/0000-0002-5198-1406

References

1. Allahverdi A. A survey of scheduling problems with no-wait in process. *Eur J Oper* 2016; Res.255: 665–686.
2. Samarghandi H and Elmekkawy TY. A meta-heuristic approach for solving the no-wait flow-shop problem. *Int J Prod Res* 2012; 50: 7313–7326.

3. Aldowaisan T and Allahverdi A. New heuristics for no-wait flowshops to minimize makespan. *Comput Oper Res* 2003; 30: 1219–1231.
4. Urgo M. A branch-and-bound approach to schedule a no-wait flow shop to minimize the CVaR of the residual work content. *Comput Ind Eng* 2019; 129: 67–75.
5. Samarghandi H and Behroozi M. On the exact solution of the no-wait flow shop problem with due date constraints. *Comput Oper Res* 2017; 81: 141–159.
6. Ye H, Li W, Abedini A, et al. An effective and efficient heuristic for no-wait flow shop production to minimize total completion time. *Comput Ind Eng* 2017; 108: 57–69.
7. Miyata HH, Nagano MS and Gupta JND. Incorporating preventive maintenance into the m-machine no-wait flow-shop scheduling problem with total flow-time minimization: a computational study. *Eng, Optimiz* 2019; 51: 680–698.
8. Ye H, Li W and Abedini A. An improved heuristic for no-wait flow shop to minimize makespan. *J Manuf Syst* 2017; 44: 273–279.
9. Zhao F, He X, Zhang Y, et al. A jigsaw puzzle inspired algorithm for solving large-scale no-wait flow shop scheduling problems. *Appl Intell* 2020; 50: 87–100.
10. Chaudhry IA, Elbadawi IA, Usman M, et al. Minimising total flowtime in a no-wait flow shop (NWFS) using genetic algorithms. *Rev Ing Invest* 2018; 38: 68–79.
11. Pan QK, Fatih Tasgetiren M and Liang YC. A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. *Comput Oper Res* 2008; 35: 2807–2839.
12. Pan QK, Wang L, Tasgetiren MF, et al. A hybrid discrete particle swarm optimization algorithm for the no-wait flow shop scheduling problem with makespan criterion. *Int J Adv Manuf Technol* 2008; 38: 337–347.
13. Gao KZ, Pan QK and Li JQ. Discrete harmony search algorithm for the no-wait flow shop scheduling problem with total flow time criterion. *Int J Adv Manuf Technol* 2011; 56: 683–692.
14. Orhan E and Abdullah G. A new hybrid ant colony optimization algorithm for solving the no-wait flow shop scheduling problems. *Appl Soft Comput* 2018; 18: 55–74.
15. Shao W, Pi D and Shao Z. A hybrid discrete optimization algorithm based on teaching–probabilistic learning mechanism for no-wait flow shop scheduling. *Knowl-Based Sys* 2016; 107: 219–234.
16. Shao W, Pi D and Shao Z. An extended teaching-learning based optimization algorithm for solving no-wait flow shop scheduling problem. *Appl Soft Comput* 2017; 61: 193–210.
17. Lin SW and Ying KC. Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics. *Omega* 2016; 64: 115–125.
18. Pan QK, Wang L and Zhao BH. An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion. *Int J Adv Manuf Technol* 2008; 38: 778–786.
19. Ding J-Y, Song S, Gupta JND, et al. An improved iterated greedy algorithm with a tabu-based reconstruction strategy for the no-wait flowshop scheduling problem. *Appl Soft Comput* 2015; 30: 604–613.
20. Lai R, Gao B and Lin W. Solving no-wait flow shop scheduling problem based on discrete wolf pack algorithm. *Sci Program* 2021; 2021: 1–6.
21. Zhu H, Qi X, Chen F, et al. Quantum-inspired cuckoo co-search algorithm for no-wait flow shop scheduling. *Appl Intell* 2019; 49: 791–803.
22. Qu C, Fu Y, Yi Z, et al. Solutions to no-wait flow shop scheduling problem using the flower pollination algorithm based on the hormone modulation mechanism. *Complexity* 2018; 2018: 1–18.
23. Zhao F, Liu H, Zhang Y, et al. A discrete water wave optimization algorithm for no-wait flow shop scheduling problem. *Expert Syst Appl* 2018; 91: 347–363.
24. Zhao F, Qin S, Yang G, et al. A factorial based particle swarm optimization with a population adaptation mechanism for the no-wait flow shop scheduling problem with the makespan objective. *Expert Syst Appl* 2019; 126: 41–53.
25. Zhao F, Qin S, Zhang Y, et al. A hybrid biogeography-based optimization with variable neighborhood search mechanism for no-wait flow shop scheduling problem. *Expert Syst Appl* 2019; 126: 321–339.
26. Zhang S, Gu X and Zhou F. An improved discrete migrating birds optimization algorithm for the no-wait flow shop scheduling problem. *IEEE Access* 2020; 8: 99380–99392.
27. Schaller J and Valente JMS. Minimizing total earliness and tardiness in a nowait flow shop. *Int J Prod Econ* 2020; 224: 107542.
28. Schaller J and Valente JMS. Scheduling in a no-wait flow shop to minimise total earliness and tardiness with additional idle time allowed. *Int J Prod Res* 2022; 60: 5488–5504.
29. Miyata HH, Nagano MS and Gupta JND. Integrating preventive maintenance activities to the no-wait flow shop scheduling problem with dependent-sequence setup times and makespan minimization. *Comput Ind Eng* 2019; 135: 79–104.
30. Wu X and Che A. Energy-efficient no-wait permutation flow shop scheduling by adaptive multi-objective variable neighborhood search. *Omega* 2020; 94: 102117.
31. Zhao F, He X and Wang L. A two-stage cooperative evolutionary algorithm with problem-specific knowledge for energy-efficient scheduling of no-wait flow-shop problem. *IEEE Trans Cybern* 2021; 51: 5291–5303.
32. Zhao F, Jiang T and Wang L. A reinforcement learning driven cooperative meta-heuristic algorithm for energy-efficient distributed no-wait flow-shop scheduling with sequence-dependent setup time. *IEEE Trans Ind Inform* 2022; 1–12. (in press). DOI: 10.1109/TII.2022.3218645
33. Wang JJ and Wang L. A cooperative memetic algorithm with feedback for the energy-aware distributed flow-shops with flexible assembly scheduling. *Comput Ind Eng* 2022; 168: 108126.
34. Zhao F, Zhang H and Wang L. A pareto-based discrete jaya algorithm for multiobjective carbon-efficient distributed blocking flow shop scheduling problem. *IEEE Trans Ind Inform* 2022; 1–10. (in press). DOI: 10.1109/TII.2022.3220860
35. Pan Z, Lei D and Wang L. A knowledge-based two-population optimization algorithm for distributed energy-efficient parallel machines scheduling. *IEEE Trans Cybern* 2022; 52: 5051–5063.
36. Zhao F, Zhang L, Cao J, et al. A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem. *Comput Ind Eng* 2021; 153: 107082.

37. Zhao F, Di S and Wang L. A hyperheuristic with q-learning for the multiobjective energy-efficient distributed blocking flow shop scheduling problem. *IEEE Trans Cybern* 2023; 53: 3337–3350.
38. Mirjalili S and Lewis A. The whale optimization algorithm. *Adv Eng Softw* 2016; 95: 51–67.
39. Luo J and Shi B. A hybrid whale optimization algorithm based on modified differential evolution for global optimization problems. *Appl Intell* 2019; 49: 1982–2000.
40. Agrawal RK, Kaur B and Sharma S. Quantum based whale optimization algorithm for wrapper feature selection. *Appl Soft Comput* 2020; 89: 106092.
41. Aziz MAE, Ewees AA and Hassanien AE. Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation. *Expert Syst Appl* 2017; 83: 242–256.
42. Li J, Guo L, Li Y, et al. Enhancing whale optimization algorithm with chaotic theory for permutation flow shop scheduling problem. *Int J Comput Intell Syst* 2021; 14: 651–675.
43. Wang YK, Wang SL, Li D, et al. An improved multiobjective whale optimization algorithm for the hybrid flow shop scheduling problem considering device dynamic reconfiguration processes. *Expert Syst Appl* 2021; 174: 114793.
44. Luan F, Cai Z, Wu S, et al. Optimizing the low-carbon flexible job shop scheduling problem with discrete whale optimization algorithm. *Mathematics* 2019; 7: 688.
45. Wang G, Li X, Gao L, et al. An effective multi-objective whale swarm algorithm for energy-efficient scheduling of distributed welding flow shop. *Oper Res* 2022; 310: 223–255.
46. Nawaz M, Enscore EE and Ham I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega* 1983; 11: 91–95.
47. Laha D and Gupta JND. A Hungarian penalty-based construction algorithm to minimize makespan and total flow time in no-wait flow shops. *Comput Ind Eng* 2016; 98: 373–383.
48. Ruiz R and Stützle T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *Eur J Oper Res* 2007; 177: 2033–2049.
49. Reeves CR. A genetic algorithm for flowshop sequencing. *Comput Oper Res* 1995; 22(1): 5–13.
50. Taillard E. Benchmarks for basic scheduling problems. *Eur J Oper Res* 1993; 64: 278–285.
51. Jiang T and Zhang C. Application of grey wolf optimization for solving combinatorial problems: job shop and flexible job shop scheduling cases. *IEEE Access* 2018; 6: 26231–26240.