Expert Systems With Applications 91 (2018) 347–363

Contents lists available at ScienceDirect

Expert Systems With Applications

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

A discrete Water Wave Optimization algorithm for no-wait flow shop scheduling problem

Fuqing Zhao[a,*], Huan Liu[a], Yi Zhang[b], Weimin Ma[c], Chuck Zhang[d]

[a] School of Computer and Communication Technology, Lanzhou University of Technology, Lanzhou 730050, China
[b] School of Mechnical Engineering, Xijin University, Xi'an 710123, China
[c] School of Economics and Management, Tongji University, Shanghai 200092, China
[d] H. Milton Stewart School of Industrial & Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA

ARTICLE INFO

Article history:
Received 26 May 2017
Revised 9 September 2017
Accepted 10 September 2017
Available online 11 September 2017

Keywords:
Water Wave Optimization (WWO)
Iterated greedy algorithm
No-wait flow shop scheduling problem
Makespan

ABSTRACT

In this paper, a discrete Water Wave Optimization algorithm (DWWO) is proposed to solve the no-wait flowshop scheduling problem (NWFSP) with respect to the makespan criterion. Inspired by the shallow water wave theory, the original Water Wave Optimization (WWO) is constructed for global optimization problems with propagation, refraction and breaking operators. The operators to adapt to the combinatorial optimization problems are redefined. A dynamic iterated greedy algorithm with a changing removing size is employed as the propagation operator to enhance the exploration ability. In refraction operator, a crossover strategy is employed by DWWO to avoid the algorithm falling into local optima. To improve the exploitation ability of local search, an insertion-based local search scheme which is utilized as breaking operator, is applied to search for a better solution around the current optimal solution. A ruling out inferior solution operator is also introduced to improve the convergence speed. The global convergence performance of the DWWO is analyzed with the Markov model. In addition, the computational results based on well-known benchmarks and statistical performance comparisons are presented. Experimental results demonstrate the effectiveness and efficiency of the proposed DWWO algorithm for solving NWFSP.

1. Introduction

Shop scheduling is an indispensable process in manufacturing plant to provide schedule planning to produce a job sequence of the product to improve productivity for the company. Shop scheduling can be classified into Single machine scheduling with single processor, Single machine scheduling problem with parallel processors (machines), Flow shop scheduling, Job shop scheduling (Zhao, Zhang, Zhang, & Wang, 2015), Open shop scheduling (Huang & Lin, 2011) and Batch scheduling (Potts & Kovalyov, 2000). Flowshop scheduling problem (FSP) is a very active research area and has been extensively studied since it was proposed by Johnson (1954). In the last decade, FSP is also a hot issue in academic research. With the introduction of the concept of industrial Internet, the processing of products in the traditional manufacturing industry has become more complex. As a result, a variety of complex flow shop scheduling models have emerged, such as non-permutation FSP (Mehravaran & Logendran, 2013), mixed-blocking FSP (Riahi, Khorramizadeh, Newton, & Sattar, 2017) and hybrid-stage-shop-scheduling (Rossi, Soldani, & Lanzetta, 2015). FSP has been extended to many branches: permutation FSP (PFSP) (Shao & Pi, 2016; Zhao, Zhang, Wang, & Zhang, 2015), no-wait FSP (NWFSP), blocking FSP (BFSP) (Fernandez-Viagas, Leisten, & Framinan, 2016), no-idle FSP (NIFSP) (Shao, Pi, & Shao, 2017), flexible Flow Shop (Shahvari, Salmasi, Logendran, & Abbasi, 2012), FSP with buffers (Zhao, Tang, Wang, & Jonrinaldi, 2014) and hybrid FSP (HFSP) (Bozorgirad & Logendran, 2013; Shahvari & Logendran, 2016), etc.

The NWFSP, as an extension of the FSP, assumes that each job should be processed without waiting time between consecutive machines. In other words, in order to satisfy the no-wait constraints, the starting time of a job on a certain machine may have to be postponed. The main reason for the research of NWFSP is the technology requirement in some manufacturing environment, such as in conventional industries including steel rolling (Aldowaisan & Allahverdi, 2012), food processing (Hall & Sriskandarajah, 1996), chemical industry (Rajendran, 1994) and pharmaceutical industry (Raaymakersa, 2000) and in some advanced manufacturing systems including just-in-time production systems (Shabtay, 2012), flexible manufacturing systems(Z. Wang, Xing, & Bai, 2005) and robotic cells (Agnetis, 2000). The previous research (Röck, 1984)

* Corresponding author.
E-mail addresses: fzhao2000@hotmail.com (F. Zhao), 799151253@qq.com (H. Liu), 1049440546@qq.com (Y. Zhang), mawm@tongji.edu.cn (W. Ma), chuck.zhang@isye.gatech.edu (C. Zhang).

http://dx.doi.org/10.1016/j.eswa.2017.09.028
0957-4174/

has proved that the NWFSP is a NP-hard problem with minimizing the makespan when the number of machines is more than two. It is significant to try more efficient algorithms to solve the NWFSP. NWFSP with the criterion to minimize the makespan is denoted as $Fm|no\text{-}wait|C_{\max}$ (Graham, Lawler, Lenstra, & Rinnooy Kan, 1979). In recent years, various heuristics and meta-heuristics are proposed for solving NWFSP. The algorithms for solving NWFSP can be classified into three categories: basic local search, basic population-based and hybrid algorithms between local search and population-based. A brief overview of these algorithms is given below.

The basic local search algorithm can perform a depth search within a certain range in the solution space. Ding et al. (2015) proposed a tabu-mechanism improved iterated greedy algorithm (TMIIG) to solve NWFSP. A speed-up method was adopted to expedite calculation speed and a tabu-mechanism was introduced to avoid numerous repeated-searching. As a result, three neighborhood searching methods were applied to get better solutions. Ding et al. (2015) proposed a block-shifting simulated annealing (BSA) algorithm. The BSA algorithm embeds a block-shifting operator based on k-insertion moves into the algorithm framework of simulated annealing. Wang, Li, and Wang (2010) proposed a tabu search algorithm, which incorporated speed-up operations, and compared its performance with few other existing heuristics. Wang (2014) presented a fast iterated local search (FILS) algorithm with high order (polynomial size) neighborhood. In the FILS algorithm, a new neighborhood along with the insertion neighborhood is used in variable neighborhood.

Population-based algorithms have good performance in global search. Gao, Pan, and Li (2011) presented a discrete harmony search algorithm (DHS) to solve the NWFSP with the objective to minimize total flow time. A novel pitch adjustment rule is employed in the improvisation to produce a new harmony. Nagano, Silva, and Lorena (2014) proposed an Evolutionary Clustering Search algorithm which divided the search space into clusters and picked out some promising search space areas from the clusters.

Hybrid algorithm combining local search with population-based global search is a more popular optimization framework. Jarboui, Eddaly, and Siarry (2011) proposed a new crossover mechanism in GA for solving NWFSP, and the variable neighborhood search (VNS) was applied to improve the quality of solutions. Samarghandi and ElMekkawy (2012) presented a hybrid algorithm combining PSO with TS algorithm. They showed that the hybrid algorithm performs better than other hybrid PSO algorithms. Akrout, Jarboui, Rebaï, and Siarry (2013) proposed a hybrid greedy randomized adaptive search procedure (GRASP) with DE for solving NWFSP. The parameters were adjusted by using DE algorithm. Davendra, Zelinka, Senkerik, and Jasek (2011) presented a novel discrete self-organizing migrating algorithm (DSOMA) for solving NWFSP, in which the 2-OPT local search was introduced to improve the quality of solutions.

Based on the shallow water wave models, the WWO algorithm was first proposed by Zheng (2015). These models derive three effective mechanisms: propagation, refraction and breaking. The WWO algorithm framework composed of the three operators is a good way to balance the capabilities of its local search and global search. In numerical tests and practical applications, the WWO algorithm shows efficient performance. At present, the WWO algorithm and its modifications have been utilized to solve many problems, such as economic dispatch problems (Siva, Balamurugan, & Lakshminarasimman, 2016), selection of key components of software formal development (Zheng, Zhang, & Xue, 2016) and other fields. Thereafter, some attempts to employ the WWO to solve the combinatorial optimization problem have emerged. Wu, Liao, and Wang (2015) redefined the WWO algorithm to solving TSP. Yun, Feng, Xin, Wang, and Liu (2016) first applied WWO algorithm to solve the flow-shop scheduling problem.

Currently, there is no reported study on the WWO algorithm to solve the NWFSP. The contributions of this study can be summarized as follows:

- We retain the original framework of the algorithm and redefine the three operators of WWO algorithm to solve $Fm|no\text{-}wait|C_{\max}$. As a simple and efficient local search algorithm, iterated greedy (IG) algorithm (Ruizab, 2007) is introduced as the propagation operator. The crossover operator and the insert-based local search are applied as the refraction and breaking operators respectively.
- In order to improve the quality of the initial population, a modified initialization strategy which combined nearest neighbor heuristic (NN) (Fink & Voß, 2003) with MNEH (Gao et al., 2011) algorithm is proposed. A ruling out inferior solution operator is introduced into the framework of WWO to enhance the convergence capability.
- At the same time, the convergence performance of the DWWO algorithm is analyzed. The convergence process of the candidate solution is mapped to the state transition process in the Markov chain and the convergence performance of the algorithm is proved.

The remainder of the paper is organized as follows: Section 2 gives the problem formulation on $Fm|no\text{-}wait|C_{\max}$. Section 3 provides a brief introduction of the original WWO algorithm. In Section 4, the proposed DWWO algorithm is described in detail for $Fm|no\text{-}wait|C_{\max}$. Section 5 makes a global convergence analysis of DWWO based on the Markov Chains. In Section 6, the main parameters are set. Then several experiments are given to verify the performance of the proposed algorithm. Section 7 summarizes conclusions and future work.

Some symbols used mostly through this paper are summarized as follows:

n	number of jobs
m	number of machines
π	the sequence of all jobs
π_i	the ith sequence (or individual) in population
$\pi(i)$	the ith job in sequence π
$\pi_i(j)$	the jth job in the ith sequence
$C_{max}(\pi)$	the make span of π
$p(\pi(i),k)$	the processing time for the $\pi(i)$ on the kth machine
$D(\pi(i\text{-}1), \pi(i))$	the processing delay time between job $\pi(i\text{-}1)$ and job $\pi(i)$
NP	the population size
λ	the wave length
λ_i	the wave length of the ith sequence
h	the wave height
h_i	the wave height of the ith sequence
α	the coefficient of ruling out inferior solution
π_D	sub-sequence containing d jobs
π_R	sub-sequence containing $(n\text{-}d)$ jobs
Π	the set of all possible permutations

2. No-wait flow shop scheduling problem(NWFSP)

NWFSP can be described as follows: n jobs are processed on m machines, and the processing routes of all jobs on m machines are the same. Several restrictions are set: A job can be only processed on a single machine at a certain moment; a machine can only process a job at a time; each job must be processed without waiting time between consecutive machines. The processing time of each job on each machine is given before. The target of scheduling is to find the best sequence which has the minimum makespan. To

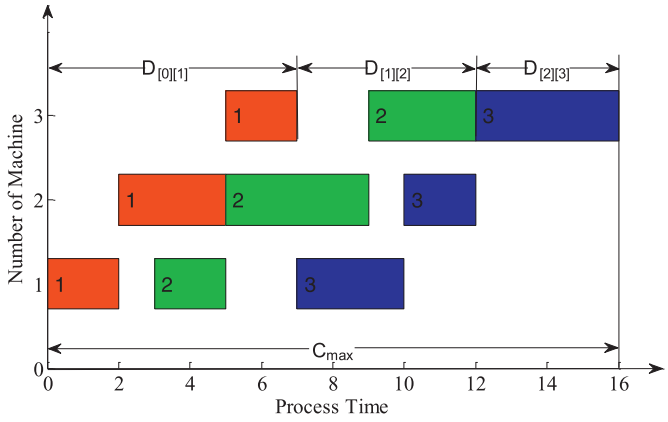

Fig. 1. A Gantt chart of a no-wait flowshop.

provide a better understanding of the problem, Fig. 1 shows the Gantt chart of a no-wait flowshop with three machines and three jobs. The first machine cannot process the second job immediately when the first machine has finished processing the first job due to the no wait requirement. Therefore the starting time of the second job is delayed. It is obvious that the makespan can be easily obtained by summing up the three completion time distances.

Assume that $\pi = [\pi(1), \pi(2), \ldots, \pi(k), \ldots, \pi(n)]$ represents a sequence to be scheduled. Let $C_{max}(\pi)$ denote the makespan of π, and $p(\pi(i),k)$ be the processing time of the ith job $\pi(i)$ on the kth machine. Let $D(\pi(i-1),\pi(i))$ represent minimum processing delay time between adjacent jobs ($\pi(i-1)$ and $\pi(i)$), which can be calculated as follows:

$$D(\pi(i-1),\pi(i)) = \max_{k=1,\cdots,m}\left\{\sum_{h=k}^{m}(p(\pi(i),h)-p(\pi(i-1),h))+p(\pi(i-1),k)\right\}. \tag{1}$$

The makespan of sequence π can be obtained as Eq. (2).

$$C_{\max}(\pi) = \sum_{i=2}^{n} D(\pi(i-1),\pi(i)) + \sum_{k=1}^{m} p(\pi(1),k). \tag{2}$$

In order to simplify the calculation process, a virtual job is introduced, and its processing time is set to zero. Sequence π is replaced by $\pi' = [\pi(0),\pi(1),\ldots,\pi(k),\ldots\pi(n)]$, and set $D(\pi(0),\pi(1)) = \sum_{k=1}^{m} p(\pi(1),k)$. Therefore, the computational formula of makespan is redefined as follows:

$$\begin{aligned} C_{\max}(\pi) &= \sum_{i=2}^{n} D(\pi(i-1),\pi(i)) + \sum_{k=1}^{m} p(\pi(1),k) \\ &= \sum_{i=2}^{n} D(\pi(i-1),\pi(i)) + D(\pi(0),\pi(1)) \\ &= \sum_{i=1}^{n} D(\pi(i-1),\pi(i)). \end{aligned} \tag{3}$$

Let Π denote the set of all possible permutations. The minimum makespan aiming to obtain can be calculated by the following Eq. (4).

$$C_{\max}(\pi^*) = \min\{C_{\max}(\pi)|\pi \in \Pi\}. \tag{4}$$

The mixed integer programming(MIP) model of the NWFSP can be formulated as follows: objective function:

$$C_{\max}(\pi^*) = \min\{C_{\max}(\pi)|\pi \in \Pi\} = \min\{C_{n,m}(\pi)|\pi \in \Pi\} \quad . \tag{5}$$

subject to:

$$\sum_{k=1}^{n} X_{j,k} = 1, \quad j \in \{1,2,\cdots,n\}. \tag{6}$$

$$\sum_{j=1}^{n} X_{j,k} = 1, \quad k \in \{1,2,\cdots,n\}. \tag{7}$$

$$C_{k+1,i} \geq C_{k,i} + \sum_{j=1}^{n} X_{j,k+1} p_{j,i}, \quad k \in \{1,2,\cdots,n-1\}, \; i \in \{1,2,\cdots,m\}. \tag{8}$$

$$C_{k,i+1} = C_{k,i} + \sum_{j=1}^{n} X_{j,k} p_{j,i+1}, \quad k \in \{1,2,\cdots,n\}, \; i \in \{1,2,\cdots,m-1\}. \tag{9}$$

$$C_{k,i} \geq 0, \quad k \in \{1,2,\cdots,n\}, \quad i \in \{1,2,\cdots,m\}. \tag{10}$$

$$X_{j,k} \in \{0,1\}, \quad j,k \in \{1,2,\cdots,n\}. \tag{11}$$

Eqs. (6) and (7) ensure that all jobs can only appear once in permutation π. Eq. (8) presents the relationship between the completion time of two consecutive jobs on each machine, which ensures that a machine cannot process multiple jobs at the same time. Eq. (9) guarantees no wait between two processing sequences. Eq. (10) specifies that the completion time of each job is nonnegative. Eq. (11) defines the range of decision variables.

3. Brief introduction to WWO algorithm

As a novel optimization algorithm, WWO algorithm (Zheng, 2015) takes inspiration from the shallow water wave theory for solving continuous optimization problems. In WWO algorithm, the solution space is analogous to the seabed area and the fitness of a point in the space is measured by its seabed depth. Each wave in the population has two attributes: wave height h and wave length λ. According to the theory of the shallow water wave, the shorter the distance between the seabed and the wave is, the higher the fitness is, the shorter the wave length λ is, and the higher the wave height h is. Fig. 2 illustrates the changes of λ and h. Let X_i^t denote ith wave after tth iteration. Firstly, a population $X^0 = [X_1^0, X_2^0, \ldots, X_{NP}^0]$ is initialized randomly. Secondly, the wave X_i^t is updated to a new wave X_i^{t+1} by searching the optimal solution in its neighborhood space. Thirdly, the wavelength of the water wave is updated, and the updated wave will influence the next propagation operation in the next iteration. During the problem-solving process, the operations of propagation, refraction and breaking are applied to evolve the population. Fig. 3 shows the framework of WWO algorithm.

In propagation operator, the process of propagating can be regarded as the process of moving from deep water to shallow water. Each dimension $d(1 \leq d \leq D)$ of a water wave $X_i^t = [x_{i1}^t, x_{i2}^t, \cdots, x_{iD}^t]$ needs to be propagated according to the Eq. (12).

$$x_{id}^t{}' = x_{id}^t + rand(-1,1) \cdot \lambda L_d. \tag{12}$$

where $rand(-1, 1)$ is a random number uniformly distributed in the range $[-1, 1]$, and L_d is the length of the dth dimension of the search space. If the new position is outside the feasible range, it will be reset to a random position in the range.

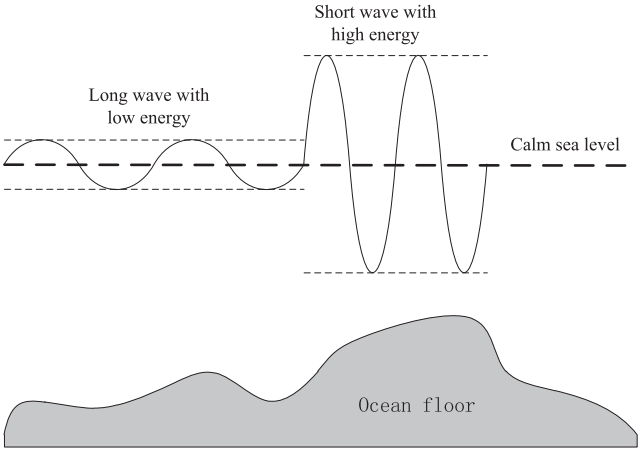

Fig. 2. Water Wave models in deep and shallow water.

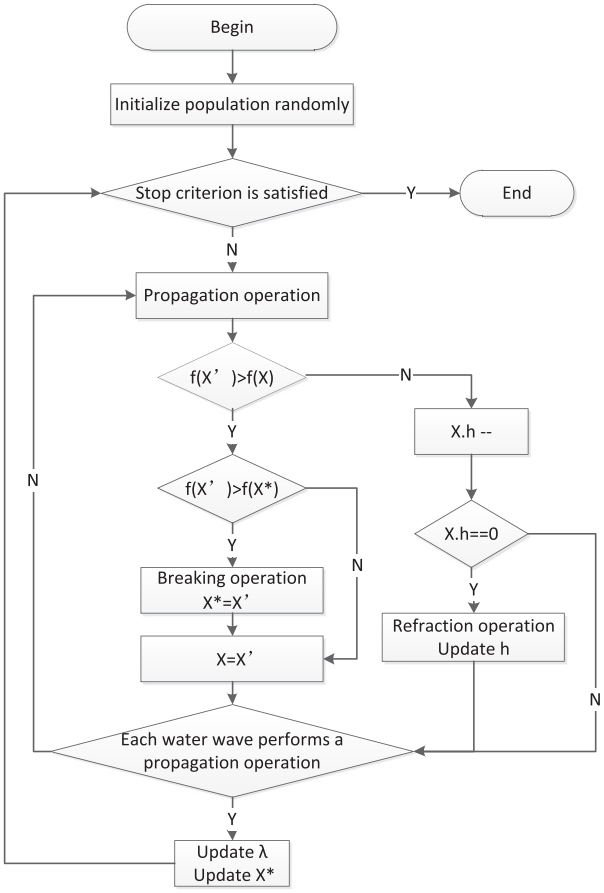

Fig. 3. The framework of WWO algorithm.

After each iteration, the wavelength of each wave is updated as Eq. (13).

$$\lambda = \lambda \cdot \alpha^{-(f(x)-f_{\min}+\varepsilon)/(f_{\max}-f_{\min}+\varepsilon)}. \tag{13}$$

where $f_{\max}$ and $f_{\min}$ are the maximum and minimum fitness values in the current population respectively, $\alpha$is the wavelength reduction coefficient, and$\varepsilon$is a small positive number to avoid division-by-zero.

In refraction operator, if a wave has not been improved after many times and its height is decreased to zero, the refraction operation is activated to update this wave. Its calculation formula is given in Eq. (14).

$$x_{id}^{t}{}' = N\left(\frac{x_d^* + x_{id}^t}{2}, \frac{\left|x_d^* - x_{id}^t\right|}{2}\right). \tag{14}$$

where $N(\mu, \sigma)$is a Gaussian random number with mean μ and standard deviation σ. Actually this mechanism makes X_i^t learn from the current optimal wave X^*. After refraction, the wave height of wave $X_i^t$is reset to$h_{\max}$, and its wavelength is set as

$$\lambda_i{}' = \lambda_i \frac{f(X_i^t)}{f(X_i^t{}')}. \tag{15}$$

In breaking operator, we perform the breaking operation only on a wave that finds a new best solution. The specific operation is to choose k dimensions randomly ($k$is a random number between 1 and $k_{\max}$) and reset them to X' as

$$x_d{}' = x_d + N(0,1) \cdot \beta L_d. \tag{16}$$

where β is the breaking coefficient.

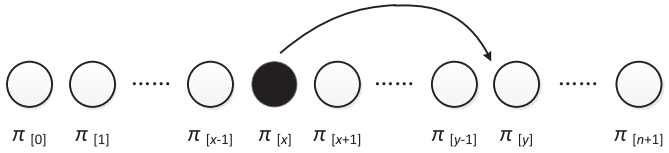

Fig. 4. Insert move.

4. The discrete Water Wave Optimization (DWWO) algorithm

In this section, the DWWO algorithm for solving the NWFSP with a makespan criterion is presented. The proposed algorithm includes five phases: a modified algorithm based on NN and MNEH to generate the initial population, a propagation operator, a refraction operator to avoid falling into local optimum, a breaking operator to do a deep local search around the best solution, and a ruling out inferior solution operator to improve the convergence. Propagation operation and breaking operation are based on basic insertion operation. Therefore, the incremental property of the insertion operation, which has been well studied in Li, Wang, and Wu (2008), is applied to calculate the fitness values of the sequences. Fig. 4 illustrates the process of inserting operation. An insert move is defined as removing job $\pi(x)$ from position x and re-inserting it into position y. According to Eq. (17), the makespan increments can be calculated in 0(1) time.

Based on the descriptions above, the differences between the original and proposed WWO are listed as follows:

- Since the original WWO algorithm is to solve the continuous function optimization problem, the goal of this paper is aiming to solve the combinatorial optimization problem. The propagation, refraction, and breaking operators in WWO are reformulated to make WWO suitable for solving NWFSP.
- In the original WWO, the population is initialized randomly, but in DWWO the population is initialized according to the MNEH + NN algorithm.
- The DWWO algorithm adopts the same algorithm framework as the original WWO. In order to speed up the algorithm convergence, the ruling out inferior solution operator is added to the DWWO after each iteration.

In the following sub-sections, the details of each phase are fully described. Thereafter, the steps of the overall framework are presented.

$$\begin{aligned}\Delta_{\pi(x,y)} &= D_{[y-1][x]} + D_{[x][y]} - D_{[y-1][y]} + D_{[x-1][x+1]} \\ &\quad - D_{[x-1][x]} - D_{[x][x+1]}.\end{aligned} \tag{17}$$

Algorithm 1 Pseudo code of initial population.

Input: the set of all jobs $S_0=\{J_1,J_2,\cdots,J_n\}$, the population size NP
Pick out NP jobs from set S_0 and construct these jobs as an array $JFs=[JF_1,JF_2,\ldots,JF_{NP}]$.
For $k=1{:}NP$
  $\pi_k(1)=JF_k$.
  $\pi_k(2)=NN(\pi_k(1))$.
  $S_1=S_0-\{\pi_k(1),\pi_k(2)\}$.
  $\pi_k=[\pi_k(1),\pi_k(2)]+MNEH(S_1)$.
  $pops(k)=\pi_k$.
End For
Output: the population: $pops$

Table 1
Comparison of the NN + NEH and NN + MNEH.

$n\times m$	Average makespan		Average improvement ratio (%)
	NN + NEH	NN + MNEH	
20 × 5	1642.3	1639.1	0.20
20 × 10	2229.8	2225.8	0.18
20 × 20	3309.3	3302.7	0.20
50 × 5	3605.1	3593.7	0.32
50 × 10	4701.3	4696.7	0.10
50 × 20	6477.7	6460.7	0.26
100 × 5	6874.1	6812.2	0.90
100 × 10	8754.9	8732.7	0.25
100 × 20	11,586.3	11,574.5	0.10
200 × 10	16,556.8	16,510.7	0.28
200 × 20	21,428.6	21,392.8	0.17
AVG	NA	NA	0.27

4.1. Initial population

The nearest neighbor NN (Fink & Voß, 2003) and the MNEH algorithm (Gao et al., 2011) are two popular heuristics. The NN heuristic appends an unscheduled job with a minimal delay time to the last job of the partial scheduled sequence at each step. The MNEH heuristic consists of three steps. In step 1, sort the jobs according to the descending order of the standard deviation of processing times of a job on different machines and generate a sequence π. In step 2, take the first two jobs of π and evaluate the two possible sub-sequences. Then, the better sub-sequence is chosen as the current sequence. In step 3, take job three and find the best sub-sequence by placing it in all possible positions of the subsequence that have been already scheduled. Thereafter, repeat the third step with $\pi(j)$ ($j=4, 5, \ldots, n$) until all jobs are sequenced.

The idea of using NN heuristic and NEH heuristic to construct initial population is presented by Pan, Fatih Tasgetiren, and Liang (2008) in the discrete particle swarm optimization algorithm (DPSO). In this paper, a new method of initial population is presented by slightly modifying. The detailed steps can be described as follows: Firstly, pick out NP jobs randomly from a set $S_0=\{J_1,J_2,\ldots, J_n\}$ which contains all the jobs with no order and a sequence $JFs=[JF_1,JF_2,\ldots, JF_{NP}]$ is constructed with these NP jobs as the first-jobs for every sequence in the initial population. Secondly, the ith job JF_i ($i=1,2,\ldots,NP$) from JFs is taken as the first job in the ith candidate sequence of initial population which is denoted as $\pi_i(1)$ and apply NN heuristic with $\pi_i(1)$ to find the second job $\pi_i(2)$. Thirdly, apply MNEH heuristic with the other (n-2) jobs to build a partial sequence. The final permutation π_i is constructed by appending the partial sequence to the first two scheduled jobs ($\pi_i(1)$ and $\pi_i(2)$). In the end, repeat the second and third steps with the elements from JFs to generate the population. The pseudo code of the initial population is given in Algorithm 1.

The effectiveness of initial population was tested by using the Taillard's 110 problem instances. Table 1 shows the results of the new initial population approach. NN + NEH was presented in Pan, Faith Tasgetiren, et al. (2008). NN + MNEH is a modified initial population method. The performance of the algorithm is improved by 0.27% on average.

4.2. Propagation operator

In each iteration, all the waves need to propagate exactly once. Let π_i denote the ith wave (sequence) and λ_i denote the wavelength of π_i. The range of propagation of π_i is determined by λ_i. The better the fitness value of water wave π_i, the smaller the corresponding wavelength λ_i, and the smaller the neighborhood search space of the propagation operation.

For solving combinatorial optimization problems, iterated greedy algorithm (IG) (Ruizab, 2007) as a simple, effective and easily adaptable algorithm is highly desirable. Based on the basic principle of NEH, the IG algorithm can obtain a better solution by doing destruction operation and construction operation constantly. In destruction operation, d jobs are randomly selected and removed from the candidate solution π containing all n jobs. π is divided into two parts, π_D containing d jobs and π_R containing (n-d) jobs. In construction operation, each job in π_D is inserted into π_R sequentially.

Here, the idea of the IG algorithm is applied to the propagation operation. In propagation operator, the parameter d which is represented as the size of removing jobs in IG algorithm, is regarded as the wavelength λ_i. Therefore, an iterative process of IG algorithm is regarded as a propagation operation of water wave. After each iteration, all the wavelengths are updated as follows:

$$\lambda_i=\left\lfloor \lambda_{\max}-(\lambda_{\max}-\lambda_{\min})\times\frac{C(\pi_i)-C(\pi_i^*)}{(C(\pi_i)-C(\pi^*))\times(1+\varepsilon)}\right\rfloor .(i=1,2,\cdots,NP) \quad (18)$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are respectively the maximum and minimum wavelength; λ_i denotes the wavelength of the ith wave ($i=1,2,\ldots,NP$); $C_{max}(\pi_i)$ is the makespan of current wave π_i; $C_{\max}(\pi_i^*)$ denotes the best makespan that the ith wave has ever propagate to; $C_{\max}(\pi^*)$is the best makespan in the population found so far in the current program. $\varepsilon=0.001$ to avoid division-by-zero. The pseudo code of the propagation operator is given in Algorithm 2.

4.3. Refraction operator

When a wave π_i has not been improved after several propagations and its wave height h_i has decreased to zero, a refraction operation needs to be performed on π_i. A crossover operator between the current wave π_i and the best wave π^* found in the program acts as the refraction operation. In details, we generate two positions P_1 and P_2 randomly and the jobs between P_1 and P_2 (including P_1, P_2) in π_i are taken out as a sub-sequence sub_1. Set $sub_2=\pi^*$-sub_1. A uniform random number r is generated between 0 and 1. If r is less than 0.5, the sequence [sub_1, sub_2] is

Algorithm 2
Pseudo code of Propagation operator.

Input: current water wave: sequence π; wavelength of π: λ
1 $\pi_R = \pi$
2 **For** $i = 1$: λ
3 $\pi(i)$ = remove one job from π_R randomly.
4 insert job $\pi(i)$ into π_D.
5 **End For**
6 **For** $i = 1$: λ
7 insert job $\pi_D(i)$ into the optimal location which gives the minimum makespan in all possible positions of π_R.
8 **End For**
Output: sequence π_R

Algorithm 3
Pseudo code of Refraction operator.

Input: current water wave: sequence $\pi_i = [\pi_i(1) \quad \pi_i(2) \quad \cdots \quad \pi_i(n)]$; the best wave π^*.
1 generate two positions P1 and P2 randomly, $1 \le P1 \le P2 \le n$.
2 $sub_1 = [\pi_i(P1) \quad \pi_i(P1+1) \quad \cdots \quad \pi_i(P2)]$.
3 $sub_2 = \pi^* - sub_1$.
4 **If** (rand < 0.5)
5 $\pi_i = [sub_1 \quad sub_2]$.
6 **Else**
7 $\pi_i = [sub_2 \quad sub_1]$.
8 **End If**
Output: sequence π_i

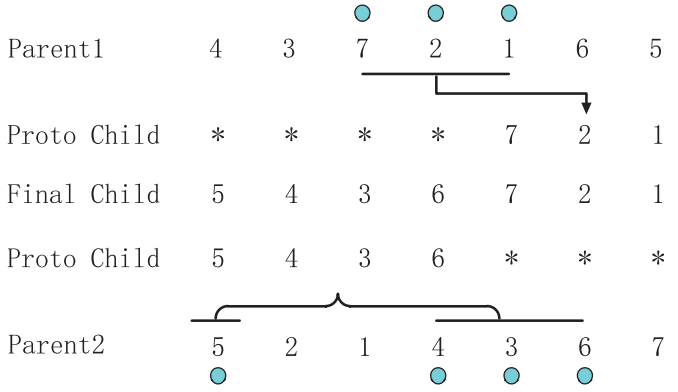

Fig. 5. crossover operation.

used as the final offspring to place sequence π_i; otherwise, the sequence $[sub_2, sub_1]$ is employed. The crossover operation is depicted in Fig. 5. The pseudo code of the refraction operator is given in Algorithm 3.

4.4. Breaking operator

If the fitness value of π_i is smaller than the best fitness value after propagating, a breaking operation needs to be executed on π_i. Its purpose is to enhance the intensive search near the relative optimal solution. In details, an insertion-based local search algorithm is utilized as the breaking operation to generate better solutions. Firstly, generate a sequence π'' with n jobs randomly. Secondly, remove the job $\pi''(i)$, $i = 1$ from the current sequence π and obtain the subsequence π'. The best schedule is found by placing $\pi''(i)$, $i = 1$ in all possible positions of the sub-sequence π'. If the fitness value of π' is better than the fitness value of π, π is replaced by π'. Thirdly, repeat the second step where $i = 2, 3, \ldots, n$ until all jobs of π'' are reinserted and evaluated. The pseudo code is outlined in Algorithm 4.

4.5. Ruling out inferior solution operator

The ruling out inferior solution operation is a probability-based acceptance strategy for inferior solutions. If an inferior child sequence is obtained after the propagation operation and its wave height h_i is reduced to 0, the algorithm can judge whether to perform the ruling out inferior solution operation based on the coefficient of ruling out inferior solution α or not. From Fig. 6, when $rand < \alpha$ ($rand$ is a random number between 0 and 1), remove the worst sequence from the population and insert the child sequence into the population; Otherwise, abandon the child sequence obtained by this iteration.

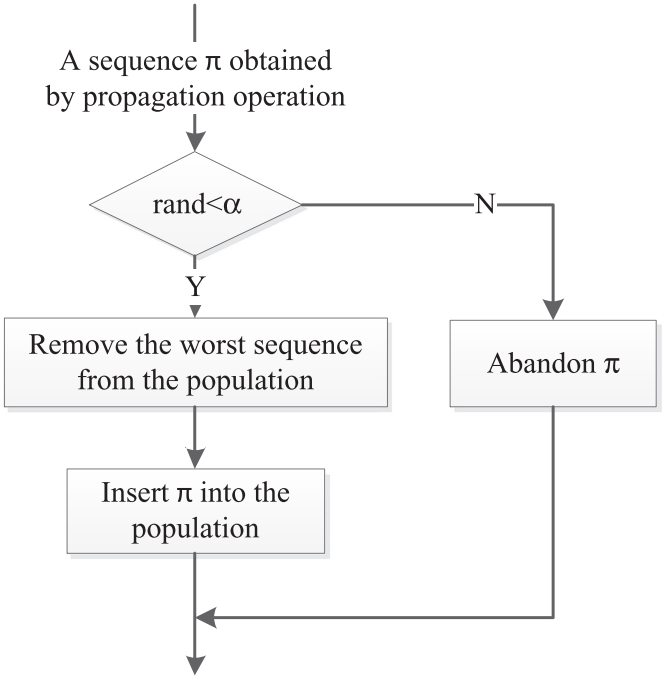

Fig. 6. The ruling out inferior solution strategy.

The effectiveness of the ruling out inferior solution strategy was tested by using the Taillard's 110 problem instances. Table 2 shows the effect after adding the ruling out inferior solution operation on the original algorithm which is denoted as $DWWO_{RIS}$. When the number of job is relatively small, the effect of using RIS strategy is not obvious. there is little difference between using and not us-

Algorithm 4
Pseudo code of breaking operator.

Input: current water wave: sequence π
Generate a sequence π'' with n jobs randomly.
For $i = 1{:}n$
  $\pi' =$ remove $\pi''(i)$ from π.
  $\pi' =$ best sequence obtained by inserting $\pi''(i)$ in all possible positions of π'
  If $C_{\max}(\pi') < C_{\max}(\pi)$
    $\pi = \pi'$.
  End If
EndFor
Output: sequence π

Table 2
Comparison of the DWWO and $DWWO_{RIS}$.

$n \times m$	Average makespan		Average improvement ratio (%)
	DWWO	$DWWO_{RIS}$	
20×5	1480.3	1480.3	0
20×10	1983	1983	0
20×20	2971.9	2971.9	0
50×5	3291.6	3282.8	0.27
50×10	4289.4	4278.6	0.25
50×20	5910.3	5899.9	0.18
100×5	6352.8	6274.7	1.24
100×10	8085	8035.2	0.62
100×20	10,803.3	10,730.8	0.68
200×10	15,546.3	15,376.3	1.11
200×20	20,428.8	20,276.7	0.75
AVG	NA	NA	0.46

ing the operation. But when the scale of the problem is gradually expanded, the difference is gradually revealed. The performance of the algorithm is improved by 0.46% on average after using the ruling out inferior solution operation. It is worth noting that the algorithm does not add extra computational cost after adding the ruling out inferior solution strategy. Therefore, the ruling out inferior solution operation is an effective optimization mechanism.

4.6. The framework of DWWO algorithm

Based on the above several important components of our proposed algorithm, the whole framework of WWO algorithm is described below. The DWWO algorithm sets the size of the parameter wavelength λ_i, $i = 1,2,\ldots,NP$ dynamically, which can directly affect the scope of propagation in the next generation, and enhance the ability of local search. At the same time, the size of the parameter h_i, $i = 1,2,\ldots,NP$ is set dynamically and refraction operation on wave π_i is performed by taking h_{max} as boundary. It can avoid this algorithm into local optimum and enhance the global search ability. The maximum computational time is the termination condition adopted in this paper. The pseudo code of the complete DWWO algorithm is given in Algorithm 5. A graphical description is shown in Fig. 7.

Algorithm 5
Pseudo code of WWO algorithm.

Initial parameters: the population size *NP*, the maximum wavelength λ_{max}, the maximum wave height h_{max}, the coefficient of ruling out inferior solution α.
Initial population $pops = [\pi_1, \pi_2, \ldots, \pi_{NP}]$.
Evaluate all the waves in *pops* and obtain the best sequence π^*.
While the termination condition is not met **do**
  While $i \le NP$ **do**
    Propagate π_i to a new π_i'.
    If $C_{\max}(\pi_i') < C_{\max}(\pi_i)$ **then**
      If $C_{\max}(\pi_i') < C_{\max}(\pi^*)$ **then**
        Break π_i' and replace it with the result.
        $\pi^* = \pi_i'$.
      End If
      $\pi_i = \pi_i'$.
      Update h_i to h_{max}.
    Else
      $h_i = h_i - 1$.
      If $h_i == 0$ **then**
        Refract π_i' and replace it with the result.
        Update h_i to h_{max}.
      Else
        If *rand* $< \alpha$ **then**
          Rule out inferior solution operation.
        End If
      End If
    End If
    Update the wavelength λ_i based on Eq. (18).
  End While
EndWhile
Return π^*

5. Convergence analysis of DWWO

According to the above definition of NWFSP, the search space of the problem is represented as $\pi_s, |\pi_s| = n!$. Let $F = \{f(\pi) | \pi \in \pi_s\}$, and clearly $|F| \le |\pi_s|$. Let $F = \{F^1, F^2, \cdots F^{|F|}\}$, $F^1 < F^2 < \cdots < F^{|F|}$. Suppose $f(\pi) = \text{makespan}(\pi), \pi \in \pi_s$, then $F^1 \le f(\pi) \le F^{|F|}$. According to the difference of fitness values, π_s is divided into $|F|$ non-empty subsets: $\pi_s^i = \{\pi | \pi \in \pi_s \quad and \quad f(\pi) = F^i\}$, $i \in \{1, 2, \cdots, |F|\}$. Then $\sum_{i=1}^{|F|} |\pi_s^i| = |\pi_s|, \forall i \in \{1, 2, \cdots, |F|\}; \pi_s = \{\pi_s^i | i = 1, 2, \cdots |F|\}$; $\pi_s^i \neq \Phi$.

Let $\pi^{i,j}$ represent the jth state in subset π_s^i, $i = \{1, 2, \cdots, |F|\}$, $j = \{1, 2, \cdots, |\pi_s^i|\}$. Let $\pi^{i,j} \to \pi^{k,l}$ denote the transition from the jth state in subset π_s^i to the lth state in subset π_s^k, and its transition probability is expressed as $p_{ij,kl}$. The transition probability from state $\pi^{i,j}$ to any state in subset π_s^k is represented as $p_{ij,k}$. And the transition probability from any state in subset π_s^i to any state in subset π_s^k is expressed as $p_{i,k}$. According to the above definition, the detailed formula is expressed as follows: $p_{ij,k} = \sum_{l=1}^{|\pi_s^k|} p_{ij,kl}$; $\sum_{k=1}^{|F|} p_{ij,k} = 1$; $p_{i,k} \ge p_{ij,k}$.

Theorem 1 (Iosifescu, 1980). Let $P = \begin{bmatrix} C & 0 \\ R & T \end{bmatrix}$ be a reducible stochastic matrix, Where C is a primitive random m-matrix and R, $T \neq 0$. Then $P^\infty = \lim_{k\to\infty} p^k = \lim_{k\to\infty} \begin{bmatrix} C^k & 0 \\ \sum_{i=0}^{k-1} T^i R C^{k-i} & T^k \end{bmatrix} = \begin{bmatrix} C^\infty & 0 \\ R^\infty & 0 \end{bmatrix}$ is a stable stochastic matrix and $P^\infty = [p_{ij}]_{n\times n}$, $\begin{cases} p_{ij} > 0 & 1 \le i \le n \quad 1 \le j \le m \\ p_{ij} = 0 & 1 \le i \le n \quad m < j \le n \end{cases}$.

Theorem 2. In DWWO algorithm, the transition probability between any two states satisfies the following conditions: $\forall k > i$, $p_{i,k} = 0$ and $\forall k \le i$, $p_{i,k} > 0$.

Proof. Let π^t denote the state set of the tth generation. Suppose x^t is a state with minimum makespan in π^t, then $fitness(x^t) = F_t$. F_{best} represents the optimal solution after t iterations. If $F_t < F_{best}$,

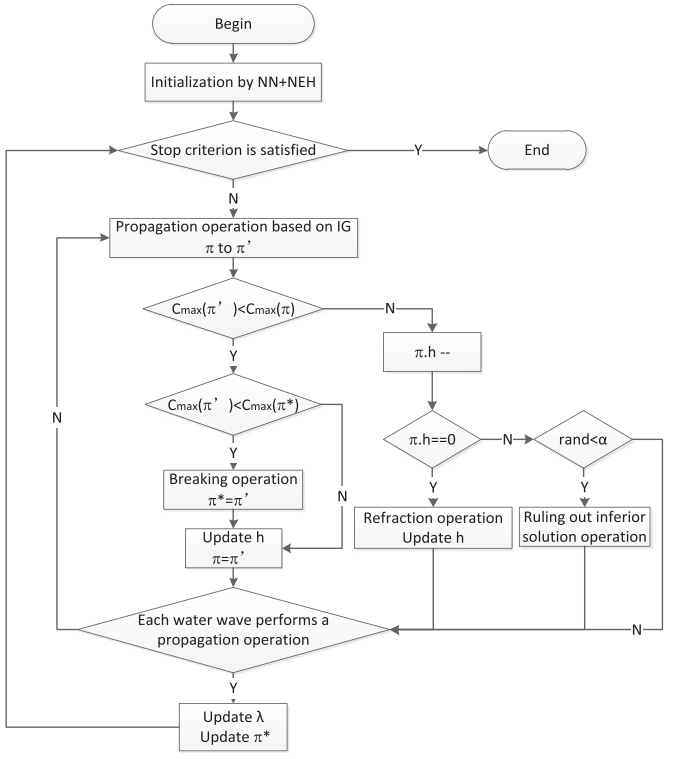

Fig. 7. The framework of DWWO algorithm.

the value of F_t will be assigned to F_{best}, otherwise it will remain unchanged. Then, the changing F_{best} is monotonous and $\forall k > i$, $p_{i,k} = 0$. In DWWO algorithm, during each iteration a propagation operation must be performed, and the refraction operation and the breaking operation are executed conditionally. In the propagation operation, the search strategy of the IG algorithm is applied. If the new solution obtained after propagation is inferior to the original solution, the new solution is retained according to the probability $P_r > 0$, otherwise the new solution is preserved according to probability 1. After propagating t $(t \to \infty)$ generations, $(P_r)^t > 0$. Then, the probability of searching for any state k in the space from the initial state i after t generations is $p_{i,\,k}{}^{\infty}$ and $p_{i,k}{}^{\infty} > (p_r)^t > 0$. Thus $\forall k \le i, \quad p_{i,\,k} > 0$.

Theorem 3. The DWWO for NWFSP can converge to the global optimal solution with probability 1.

Proof. According to Theorems 1 and 2, the transition process of $|F|$ states in π_s can be mapped to the state transition process in the Markov chain. The transition matrix is as follows:

$$P = \begin{bmatrix} p_{1,1} & 0 & \cdots & 0 \\ p_{2,1} & p_{2,2} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ p_{|F|,1} & p_{|F|,2} & \cdots & p_{|F|,|F|} \end{bmatrix} = \begin{bmatrix} C & 0 \\ R & T \end{bmatrix}.$$

From Theorem 2, $\forall k > i$, $p_{i,k} = 0$, $p_{1,2} = p_{1,3} = \cdots = p_{1,|F|} = 0$. From the previous definition, $\sum_{k=1}^{|F|} p_{1,k} = p_{1,1} + p_{1,2} + \cdots + p_{1,|F|} = 1$, thus $C = (p_{1,1}) = (1) \neq 0$. According to Theorem 2, $\forall k \le i$, $p_{i,k} > 0$, it can obtain that $R = (p_{2,1}, p_{3,1}, \cdots, p_{|F|,1})^T \neq 0, T = \begin{bmatrix} p_{2,2} & \cdots & 0 \\ \vdots & & \vdots \\ p_{|F|,2} & \cdots & p_{|F|,|F|} \end{bmatrix} \neq 0.$

Thus C is a primitive random 1-matrix and R, $T \neq 0$. $P = \begin{bmatrix} C & 0 \\ R & T \end{bmatrix}$ is a reducible stochastic n-matrix, so $P^{\infty} = \lim_{k\to\infty} p^k = \lim_{k\to\infty} \begin{bmatrix} C^k & 0 \\ \sum_{i=0}^{k-1} T^i R C^{k-i} & T^k \end{bmatrix} = \begin{bmatrix} C^{\infty} & 0 \\ R^{\infty} & 0 \end{bmatrix}$ and P^{∞} meets the following requirements: $P^{\infty} = [p_{ij}]_{n\times n}$, $\begin{cases} p_{ij} > 0 & 1 \le i \le n \quad 1 \le j \le m \\ p_{ij} = 0 & 1 \le i \le n \quad m < j \le n \end{cases}$. The following conclusions can be drawn: $R^{\infty} = (1, 1, \cdots, 1)^T$, $C^{\infty} = (1)$, $P^{\infty} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix}$ is a stable stochastic matrix.

Consequently, DWWO for NWFSP can converge to the global optimal solution with probability 1.

6. Numerical experiments

The DWWO algorithm for NWFSP was coded using MATLAB. The simulation experiments were carried out on a personal computer (PC) with Intel (R) Core (TM) i7-6700 CPU 3.4 GHz and 8.00GB memory with a windows Server 2012 Operating System.

In order to test the performance of our proposed DWWO algorithm, several comparative experiments were conducted by using three commonly used benchmark problems. (i) 8 instances provided by Carlier (1978): Car01 to Car08 that has eight sets of different sizes problems with small job and machine sizes but large processing time. (ii) 29 instances provided by Reeves (1995): Rec01 to Rec41 that consists of 21 problem instances with seven sub-sets of different sizes, ranging from 20 jobs and 5 machines to 75 jobs and 20 machines. (iii) 120 instances provided by Taillard (1993): Ta001-Ta120 consists of 120 problem instances with 12 sub-sets of different sizes, ranging from 20 jobs and 5 machines to 500 jobs and 20 machines. In order to demonstrate the effectiveness of our proposed algorithm and compare the performance with other approaches visually, the average relative percentage deviation (ARPD) was applied to measure the quality of the experimental results. The calculation formula of ARPD is as follows:

$$ARPD = \frac{1}{R}\sum_{r=1}^{R}\frac{C_r - C_R^*}{C_R^*} \times 100. \tag{19}$$

where C_r is the solution of the rth experiment which is generated by a specific algorithm, and C_R^* is the optimum solution or the best solution found so far. In addition, the standard deviation (SD) is also recorded to indicate the robustness of the algorithm, where the SD is calculated as follows:

$$SD = \sqrt{\frac{\sum_{r=1}^{R}(C_r - \overline{C})^2}{R}}. \tag{20}$$

where C_r is the solution of the rth experiment which is generated by a specific algorithm, and $\overline{C}$ is the mean value of R solutions.

6.1. Parameters analysis

The performance of algorithm is largely determined by the parameters, so it is crucial to find a best parameter combination by conducting parameter calibration experiment. The proposed algorithm has five parameters: NP (the population size), λ_{max} (the maximum wavelength), $\lambda_{min} = \lambda_{max}/2$ (the minimum wavelength), h_{max} (the maximum wave height), α (the coefficient of ruling out inferior solution). Since small-scale population is one of the characteristics of DWWO algorithm and its population size is relative to the job size in a specific instance, we set $NP = \lambda_{max}$. If λ_{min} is set to 1, the propagation operation is equivalent to the insertion operation of a single job at the beginning of the algorithm. If

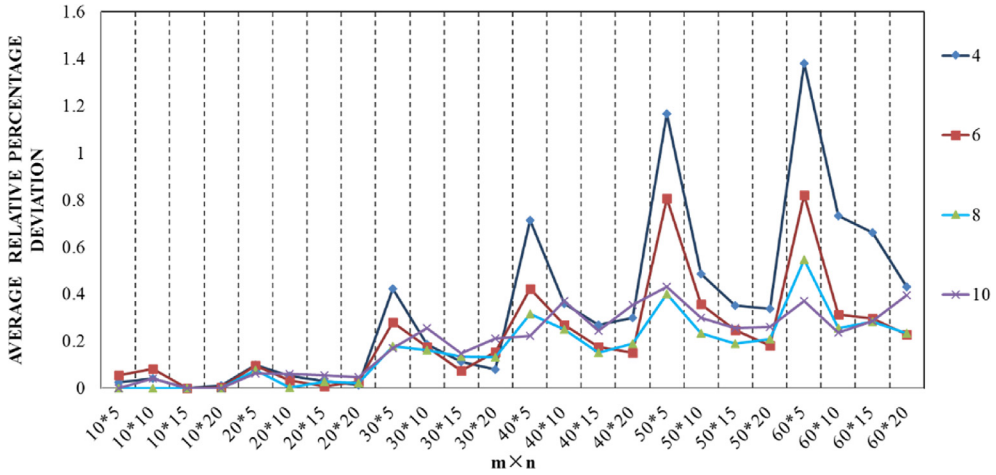

Fig. 8. Relation graph between the size of job and machine and wave height.

$\lambda_{min} = \lambda_{max}$, the propagation operation will become a common IG algorithm. Therefore, we adopted a compromise to set $\lambda_{min} = \lambda_{max}/2$. Now three parameters ($\lambda_{max}$, h_{max}, α) need to be set to find the best combination by implementing the Design of Experiments method (Montgomery, 2001). A full factorial design is carried out by using the three parameters as factors at the following levels:

$h_{max} \in \{4, 6, 8, 10, 12\}; \alpha \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}; \begin{cases} \lambda_{max} \in \{\frac{n}{2}, \frac{n}{3}, \frac{n}{4}, \frac{n}{5}\} & if \ \frac{n}{*} \le 50; \\ \lambda_{max} = 50 & otherwise \end{cases}$ (If the solution is fractional, the decimal portion is rounded.).

There are $5 \times 6 \times 4 = 120$ different combinations for the proposed algorithm. In each combination, for each instance, ten replications are tested and the ARPD values are recorded. In order to set a suitable parameter combination, instances of 9 different combinations given in Taillard are applied to determine the parameters, which include 20×5, 20×10, 20×20, 50×5, 50×10, 50×20, 100×5, 100×10, 100×20. The maximum running time is set as $(n^2/2 \times 10)$ ms for these Taillard's instances.

6.1.1. Analysis of fitting for parameter calibration

After running all the instances, we find that the size of the job and machine has a great impact on the selection of wave height. 240 VRF instances (Vallada, Ruiz, & Framinan, 2015) were used to fit the parameter h_{max}. As can be seen from Fig. 8, when the size of the job and machine is relatively small, its impact on the wave height is relatively small. When the size of the job and machine becomes larger and larger, the relationship between them appears: the size of the job is proportional to the wave height, and the size of the machine is inversely proportional to the wave height. Therefore, as a parameter with strong sensitivity, wave height cannot be set as a constant, and its selection needs to adapt according to the size of the job and machine. So according to the experimental data, the wavelength, the size of the job and the size of the machine were fitted to three-dimensional graphics as shown in Fig. 9, the formula is as follows:

$$h_{max} = 0.1133 \times n - 0.4466 \times m - 0.0022 \times n \times m + 0.0001 \times n^2 + 0.0132 \times m^2 + 4.5407$$

6.1.2. Analysis of variance for parameter calibration

After determining the parameter h_{max}, the multi-factor analysis of variance (ANOVA) is utilized to set parameter λ_{max} and α. To apply ANOVA, three main hypotheses are checked i.e., normality, homoscedasticity and independence of residuals. Statistical analysis shows that the three hypotheses can be accepted. The ANOVA results can be seen in Table 3.

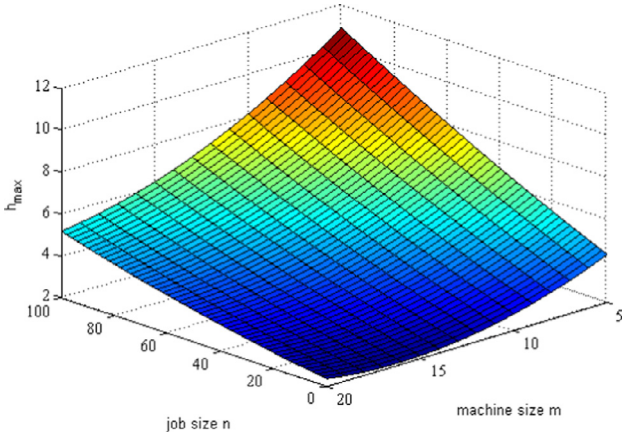

Fig. 9. Fitting graph between the size of job and machine and wave height.

Table 3
ANOVA results for the experiment on tuning parameters of DWWO.

Source	Sum of squares	Df	Mean square	*F*-Ratio	*p*-Value
Main effects					
A:λ	0.055	3	0.18	31.310	0.000
B:α	0.162	5	0.32	55.036	0.000
Interaction					
AB	0.108	15	0.007	12.221	0.000
Residual	0.014	24	0.001		
Total(Corrected)	0.346	47			

According to Table 3, all the two parameters and their interaction have small p-values, which cannot be treated as a clear indicator of the significance to analyze parameter performance. F-ratio can replace the function of p-value to show the significance of parameters. A large F-ratio indicates that this factor has a considerable effect on the algorithm. It can be seen that parameter α has the most significant effect on the performance of DWWO algorithm. With a small α, the convergence of algorithm may slow down while a large α may lead to premature convergence. This also means that the introduction of the ruling out inferior solution strategy can improve the performance of the algorithm. The maximum wavelength λ_{max} is the second significant parameter.

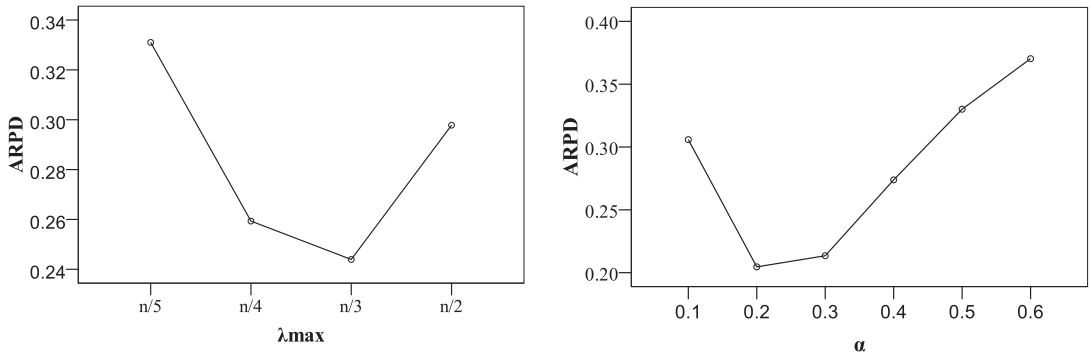

Fig. 10. Factor level trend of DWWO.

Table 4
Multiple comparison of parameter λ_{max}.

λ_{max}	Average ARPD	95% Confidence interval	99% Confidence interval
$n/5$	0.331	a	A
$n/2$	0.298	b	A
$n/4$	0.262	c	B
$n/3$	0.239	c	B

Table 5
Multiple comparison of parameter α.

α	Average ARPD	95% Confidence interval	99% Confidence interval
0.6	0.366	a	A
0.5	0.330	ab	AB
0.1	0.306	bc	BC
0.4	0.274	c	C
0.3	0.213	d	D
0.2	0.205	d	D

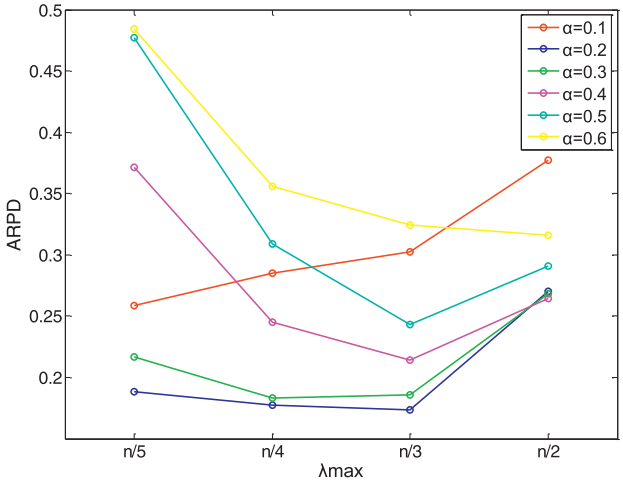

Fig. 11. AB interaction for DWWO.

λ_{max} determines the range of local search. The larger the λ_{max}, the larger the search range, and the slower the convergence speed of the algorithm. On the contrary, the smaller the λ_{max}, the smaller the search range, and the algorithm is easy to fall into local optimum.

A multiple comparison based on Tukey test is employed to examine the differences between levels of parameters. From Table 4, the levels of the parameter λ_{max} are divided into three subsets with a 95% confidence interval: {$n/5$}, {$n/2$}, {$n/3$, $n/4$}. In the case of a 99% confidence interval, the levels of the parameter λ_{max} are divided into two subsets: {$n/2$, $n/5$}, {$n/3$, $n/4$}. Between the subsets, the levels of the parameter are significantly different. There is no significant difference between the levels of the parameter within the subset. As shown in Table 5, the levels of the parameter α are divided into four subsets with a 95% confidence interval: {0.5,0.6}, {0.1,0.5}, {0.1,0.4}, {0.2,0.3}. In the case of a 99% confidence interval, the distribution of the subsets is the same as 95% confidence interval. According to Fig. 10, the parameters $\alpha = 0.2$ and $\lambda_{max} = n/3$ make the ARPD minimum. From Table 3, it is clear that there is also a significant correlation between parameter A and parameter B. Therefore, it is necessary to take into account the interaction of the two parameters of AB. The AB interaction is illustrated in Fig. 11 . From Fig. 11, we conclude that the best results are obtained with $\alpha = 0.2$ and $\lambda_{max} = n/3$, which is the same as the results of the single factor analysis above.

6.2. Comparison DWWO with existing algorithms

In the following experiment, the effectiveness and efficiency of the proposed DWWO algorithm for solving NWFSP are tested. The DWWO algorithm was compared with other four algorithms including IIGA (Pan, Wang, & Zhao, 2008), $DPSO_{VND}$ (Pan, Faith Tasgetiren, et al., 2008), TMIIG (Ding et al., 2015) and MWWO (Yun et al., 2016). In the MWWO algorithm, only small-scale instances were tested by the authors. In order to compare with the existing WWO-based algorithms, the MWWO algorithm is re-implemented and compared on the Carlier's and Reeve's instances. For the other three algorithms, three standard test sets of different scales (Carlier's 8 problem instances with small-scale, Reeve's 21 problem instances with medium-scale, Taillard's 120 problem instances with large-scale) are applied to test their performance. In order to ensure the consistency of the operating environment, the fairness of the algorithm, the contrast algorithms were re-implemented and ran on the same platform with the same maximum running time. In Carlier's instances, the maximum running time of each compared algorithm is set as $(n^2/2)$ ms. In Reeve's and Taillard's instances, the maximum running time of each compared algorithm is set as $(n^2/2 \times 10)$ ms.

6.2.1. Comparison for Carlier's instances

Table 6 shows the performance of these algorithms on the Carlier's instances. As seen in Table 6, the average ARPD value of the proposed DWWO algorithm is 0, which is superior to the corre-

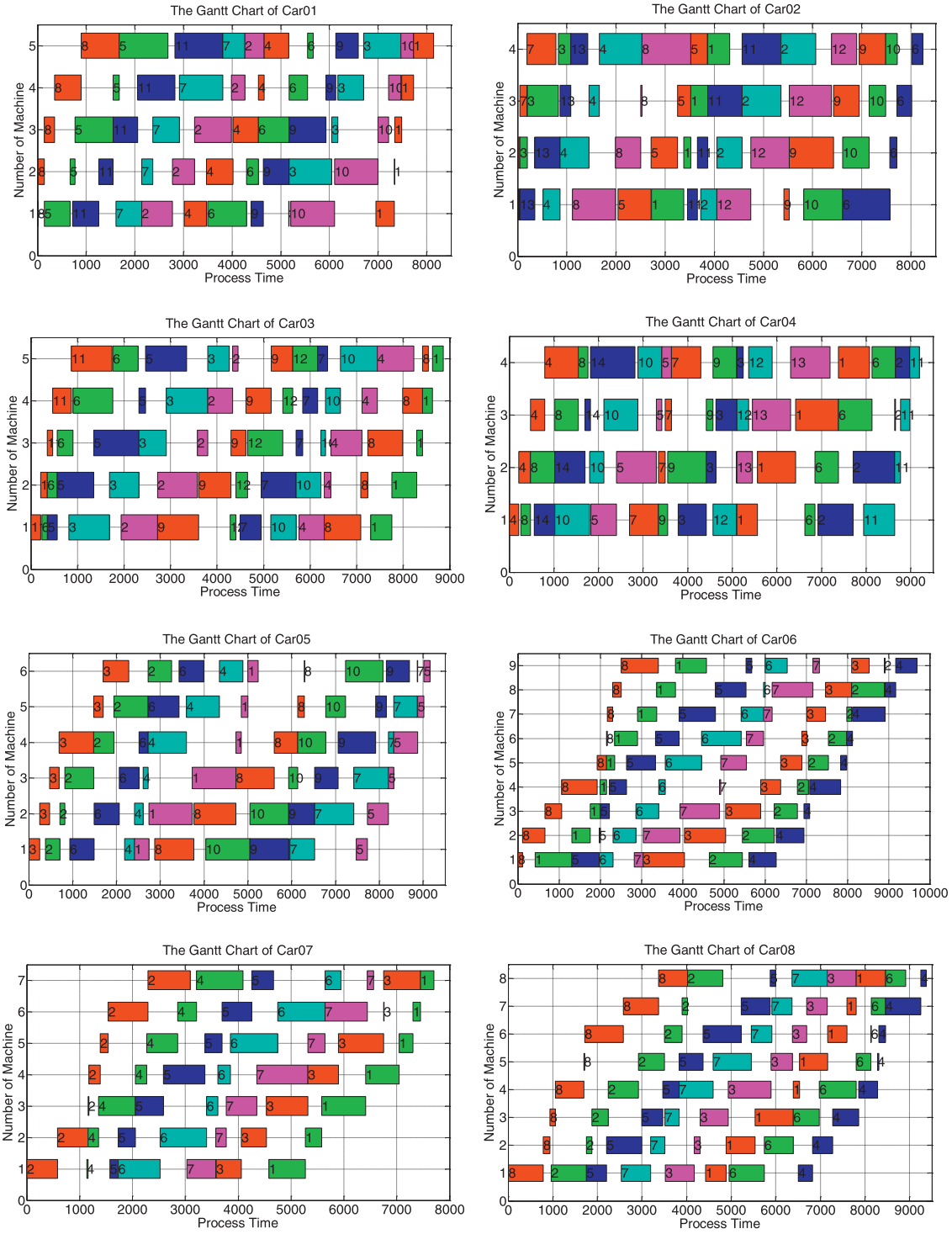

Fig. 12. The Gantt Chart from Car01 to Car08.

sponding values of 0.005 and 0.003 by the $DPSO_{VND}$ and TMIIG respectively and equals to the ARPD value of IIGA. This shows the superiority of the DWWO algorithm in solving the small scale scheduling problem. The average SD value of the DWWO algorithm is 0, which is lower than 0.367 and 0.344 obtained by the $DPSO_{VND}$ and TMIIG. It can be seen that the DWWO algorithm is robust. The best ARPD and SD, which are calcualted by DWWO, IIGA, MWWO, $DPSO_{VND}$ and TMIIG, are listed in bold in Table 6-8. In WWO algorithm, the main body is the three main operations (propagation operation, refraction operation and breaking operation). Especially, the propagation operation is the key of the three operations. Propagation operation is performed once in each iteration, and refraction operation and breaking operation are performed only under certain conditions. In the MWWO algorithm, propagation operation is based on reverse operation. The reverse operation(Wu et al., 2015) is a search strategy with a high randomness. It is the randomness as a main reason that affects the performance of the algorithm. Fig. 12 shows the Gantt chart from Car01 to Car08. From

Table 6
Comparison of results based on Carlier's benchmark set.

Instance	$n \times m$	IIGA		MWWO		$DPSO_{VND}$		TMIIG		DWWO	
		ARPD	SD	ARPD	SD	ARPD	SD	ARPD	SD	ARPD	SD
Car1	11×5	0	0	0.37	13.9	0	0	0	0	**0**	**0**
Car2	13×4	0	0	0.7	27.79	0	0	0	0	**0**	**0**
Car3	12×5	0	0	0.1	7.97	0	0	0	0	**0**	**0**
Car4	14×4	0	0	5.26	42.28	0.039	2.939	0.026	2.749	**0**	**0**
Car5	10×6	0	0	0.74	34.03	0	0	0	0	**0**	**0**
Car6	8×9	0	0	0	0	0	0	0	0	**0**	**0**
Car7	7×7	0	0	0	0	0	0	0	0	**0**	**0**
Car8	8×8	0	0	0.01	4.2	0	0	0	0	**0**	**0**
AVG		0	0	0.897	16.271	0.005	0.367	0.003	0.344	**0**	**0**

Table 7
Comparison of results based on Reeve's benchmark set.

Instance	$n \times m$	IIGA		MWWO		$DPSO_{VND}$		TMIIG		DWWO	
		ARPD	SD	ARPD	SD	ARPD	SD	ARPD	SD	ARPD	SD
Rec01	20×5	0	0	1.68	7.34	0	0	0	0	**0**	**0**
Rec03	20×5	0	0	3.65	6.75	0	0	0	0	**0**	**0**
Rec05	20×5	0	0	1.68	6.58	0.01	0.3	0	0	**0**	**0**
Rec07	20×10	0	0	1.15	11.94	0	0	0	0	**0**	**0**
Rec09	20×10	0	0	1.98	18.09	0	0	0	0	**0**	**0**
Rec11	20×10	0	0	2.28	10.82	0	0	0	0	**0**	**0**
Rec13	20×15	0	0	2.16	28.42	0	0	0	0	**0**	**0**
Rec15	20×15	0	0	0.74	15.46	0	0	0	0	**0**	**0**
Rec17	20×15	0	0	0.96	13.05	0	0	0	0	**0**	**0**
Rec19	30×10	0.07	2.45	2.61	11.7	0.26	8.13	0	0	**0**	**0**
Rec21	30×10	0.21	3.48	2.27	9.06	0.17	4.98	0.05	2.15	**0.02**	**1.5**
Rec23	30×10	0.03	1.92	3.25	12.93	0.17	9.47	0.08	6.3	**0**	**0**
Rec25	30×15	0.03	1.37	2.21	17.58	0.08	3.86	0	0	**0**	**0**
Rec27	30×15	0.16	6.31	2.63	22.26	0.16	5.5	0.32	0	**0**	**0**
Rec29	30×15	0	0	2.15	29.68	0.13	7.07	0.17	7.03	**0**	**0**
Rec31	50×10	0.62	7.79	4.11	25.54	0.51	8.31	0.09	3.49	**0.09**	**3.01**
Rec33	50×10	1.14	13.9	5.04	14.75	0.98	16.71	0.37	9.75	**0.31**	**5.47**
Rec35	50×10	0.66	10.16	3.65	20.18	0.69	15.61	0.18	5.85	**0.04**	**3.15**
Rec37	75×20	1.23	15.55	4.78	30.9	1.04	20.48	0.4	12.43	0.41	**9.97**
Rec39	75×20	1.27	17.66	4.39	13.32	0.7	13.74	**0.24**	**8.42**	0.27	12.5
Rec41	75×20	1.09	15.63	3.9	28.04	0.79	14.45	0.39	**8.55**	**0.33**	10.35
AVG		0.31	4.58	2.727	16.876	0.27	6.12	0.11	3.05	**0.07**	**2.19**

these figures, we can learn that the DWWO algorithm could obtain the best solutions.

6.2.2. Comparison for Reeve's instances

For the Reeve's instances, DWWO algorithm has advantages. As shown in Table 7, the average ARPD value of the DWWO algorithm is 0.07, which is less than 0.11,0.27,0.31 of TMIIG, $DPSO_{VND}$, IIGA, and its average SD value is 2.19, which is also less than 3.05,6.12,4.58 obtained by TMIIG, $DPSO_{VND}$ and IIGA. The above results illustrate that the performance and stability of DWWO algorithm are better than these state-of-the-art algorithms for the NWFSP in minimizing the makespan. Fig. 13 and Fig. 14 show the convergence of the four comparative algorithms in solving Rec35 and Rec41, respectively. The abscissa represents the running time, and the ordinate represents the makespan. It can be seen that the DWWO algorithm has faster convergence velocity and higher accuracy than other algorithms and has the ability to escape from local optima quickly. Fig. 17 and Fig. 18 show the boxplots of the four comparative algorithms in solving Rec35 and Rec41, respectively. The boxplot can directly show the discrete degree of the test data, which reflects the stability of the algorithm.

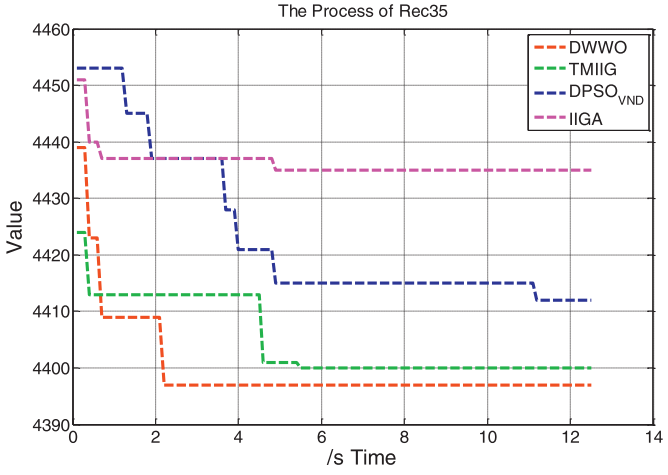

Fig. 13. Convergence curve for the instance of Rec35.

6.2.3. Comparison for Taillard's instances

To further demonstrate the performance of DWWO for solving the large-scale problems, Taillard's instances are executed to test the effectiveness of DWWO. As can be seen from Tables 8 and A.1 in Appendix, our proposed DWWO algorithm is superior to other algorithms about 70% of the instances. Especially in solving

Table 8
Comparison of results based on Taillard's benchmark set.

$n \times m$	IIGA		$DPSO_{VND}$		TMIIG		DWWO	
	SD	ARPD	SD	ARPD	SD	ARPD	SD	ARPD
20×5	**0**	**0**	0.22	0.01	**0**	**0**	**0**	**0**
20×10	0.32	0.01	0.08	0.01	0.08	**0**	**0**	**0**
20×20	0.32	0.02	1.51	0.05	0.32	0.02	**0**	**0**
50×5	6.36	1.34	5.83	0.52	**2.23**	**0.1**	4.45	0.39
50×10	5.25	0.71	7.71	0.53	4.01	0.15	**2.72**	**0.09**
50×20	8.86	0.58	14.09	0.55	6.74	0.18	**3.53**	**0.06**
100×5	13.58	2.78	13.37	0.9	**5.15**	**0.12**	10.09	0.68
100×10	13.77	1.7	18.73	0.79	9.24	**0.21**	**6.73**	0.27
100×20	17.07	1.46	22.89	0.82	15.96	0.3	**9.62**	**0.13**
200×10	22.56	2.49	32.76	1.11	**19.23**	**0.18**	27.8	0.47
200×20	26.24	2.1	39.29	1.15	25.27	0.32	**23.38**	**0.16**
500×20	62.69	2.94	56.94	1.71	**44.39**	0.65	50.72	**0.15**

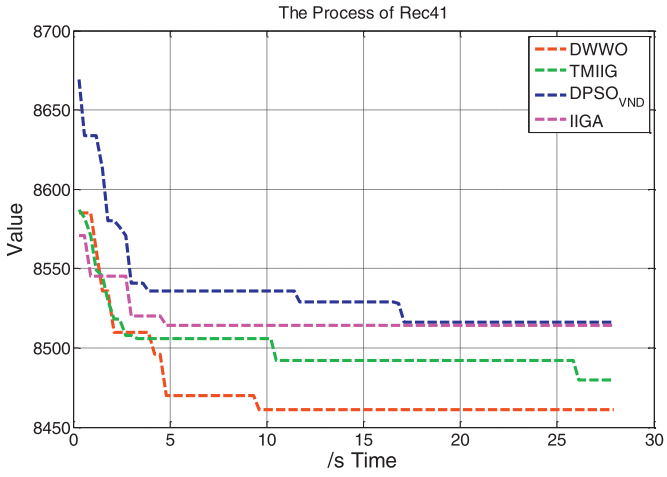

Fig. 14. Convergence curve for the instance of Rec41.

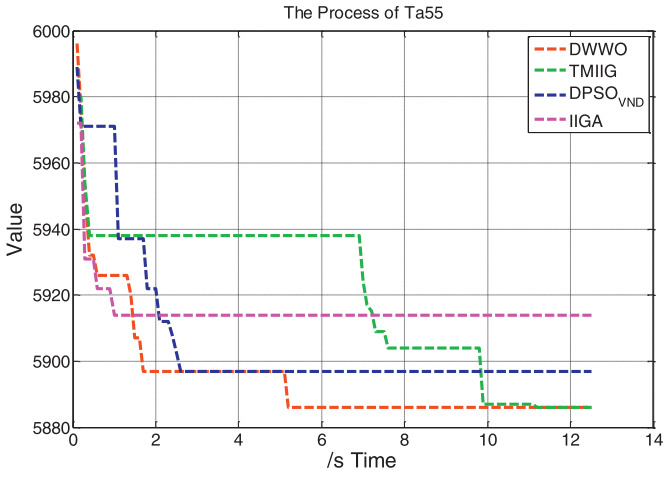

Fig. 15. Convergence curve for the instance of Ta55.

large-scale problems, such as 200×20, 500×20, the performance value of the DWWO algorithm is much better than other algorithms. There are still 30% instances of the solution to be inferior to the TMIIG algorithm, which is attributed to the impact of parameter sensitivity. In order to reflect the adaptability of parameters in different scale instances, the selected parameters can be adapted to most instances, but not all instances. In Ta90 problem, the wave height is 7. Fig. 21 shows that 7 is not the optimal wave height for Ta90 instance. Therefore, the DWWO algorithm does not fully play the performance of the algorithm in a few instances. Figs. 15 and 16 are the convergence graphs of Ta55 and Ta90, respectively. It is obvious that the convergence rate of DWWO algorithm is faster than other algorithms and not easy to be precocious. Figs. 19 and 20 suggest that our proposed DWWO still has superior stability when solving relative large-scale problems. These benefits from the search strategy of the DWWO algorithm framework and the combination with IG algorithm. Because the operating environment of all comparison algorithms is the same, the proposed DWWO algorithm is better than other comparison algorithms.

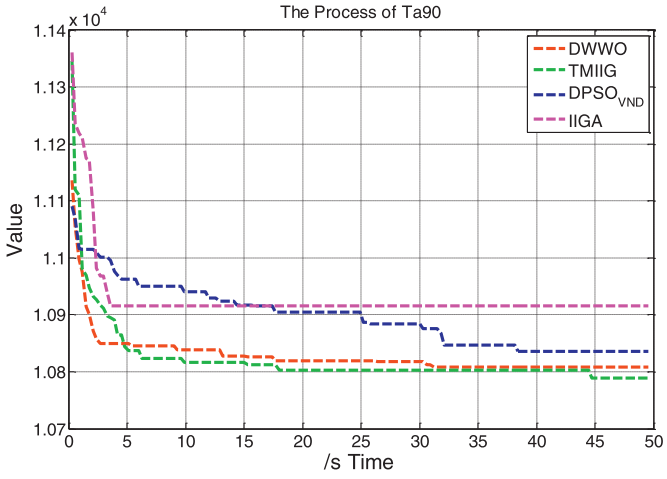

Fig. 16. Convergence curve for the instance of Ta90.

7. Conclusion and future work

In this paper, a novel DWWO algorithm is applied to solve the NWFSP problem. The main idea of the proposed algorithm is to integrate the IG algorithm into the framework of the WWO algorithm in an adaptive way. In the framework of WWO algorithm, the ever-changing wavelength and wave height increase the global search ability of the algorithm, and avoid the algorithm to fall into the local optima. The IG algorithm enhances the local search capability of the algorithm. Thereafter, the ruling out inferior solution operation focuses on searching around the relatively superior so-

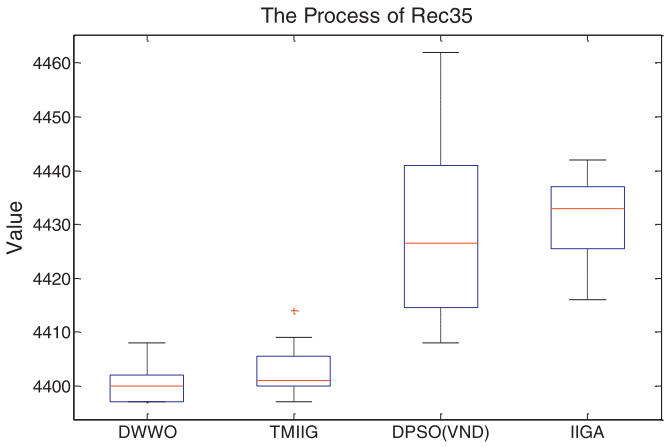

Fig. 17. Boxplot for the instance of Rec35.

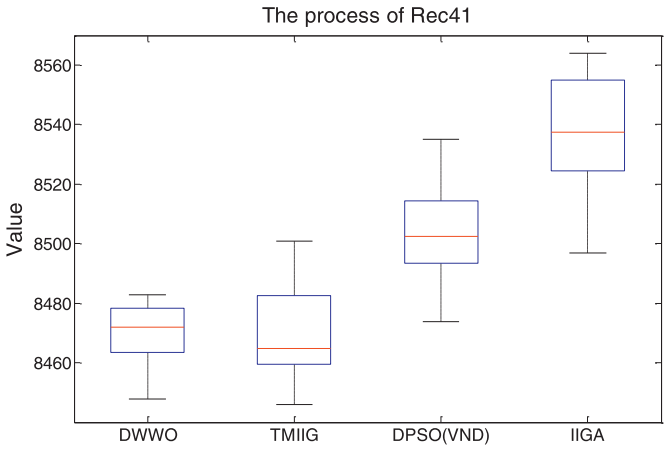

Fig. 18. Boxplot for the instance of Rec41.

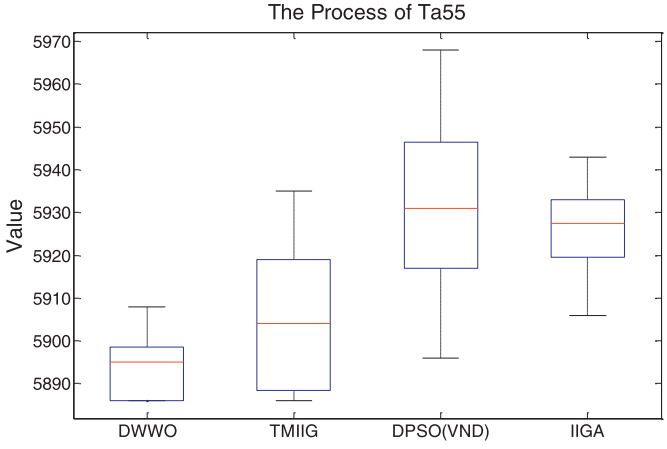

Fig. 19. Boxplot for the instance of Ta55.

lution, which enhances the convergence rate of the algorithm. The integration of these three basic strategies provides a guarantee for the efficiency and effectiveness of the algorithm.

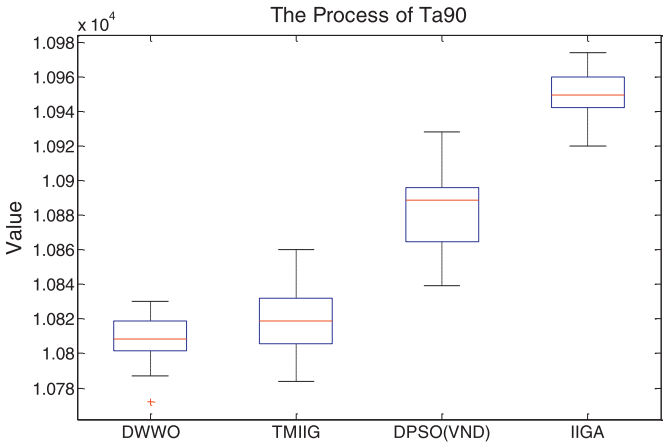

Fig. 20. Boxplot for the instance of Ta90.

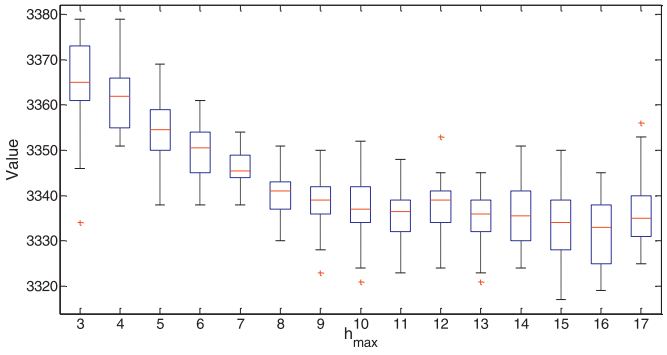

Fig. 21. Boxplot for the instance of Ta90.

Several issues deserve more in-deep study in the future. Firstly, in refraction and breaking operations, only two basic algorithmic strategies are utilized. If higher efficient local search strategies can be applied to the two parts, the performance of the algorithm will be more superior. Secondly, it would be meaningful to combine other efficient search strategies and optimization mechanisms with the framework of WWO algorithm to solve NWFSP. Moreover, the DWWO algorithm will be extended to solve other complex scheduling problems, such as the flexible flow shop scheduling, the job shop scheduling and the hybrid flow shop scheduling.

Acknowledgments

This work was financially supported by the National Natural Science Foundation of China under grant numbers 61663023. It was also supported by the Key Research Programs of Science and Technology Commission Foundation of Gansu Province (17YF1WA160), General and Special Program of the Postdoctoral Science Foundation of China, the Science Foundation for Distinguished Youth Scholars of Lanzhou University of Technology, Lanzhou Science Bureau project under grant numbers 2012M521802, 2013T60889, J201405, and 2013-4-64, respectively.

Appendix

Table A.1
Comparison of results based on Taillard's benchmark set.

Instance	IIGA		$DPSO_{VND}$		TMIIG		DWWO	
	Best	Mean	Best	Mean	Best	Mean	Best	Mean
Ta001	1486	1486	1486	1486	1486	1486	1486	1486
Ta002	1528	1528	1528	1528.6	1528	1528	1528	1528
Ta003	1460	1460	1460	1460	1460	1460	1460	1460
Ta004	1588	1588	1588	1588	1588	1588	1588	1588
Ta005	1449	1449	1449	1449	1449	1449	1449	1449
Ta006	1481	1481	1481	1481	1481	1481	1481	1481
Ta007	1483	1483	1483	1483	1483	1483	1483	1483
Ta008	1482	1482	1482	1483.2	1482	1482	1482	1482
Ta009	1469	1469	1469	1469	1469	1469	1469	1469
Ta010	1377	1377	1377	1377	1377	1377	1377	1377
Ta011	2044	2044	2045	2045	2044	2044	2044	2044
Ta012	2166	2166	2166	2166	2166	2166	2166	2166
Ta013	1940	1940	1940	1940.4	1940	1940.4	1940	1940
Ta014	1811	1811	1811	1811	1811	1811	1811	1811
Ta015	1933	1933	1933	1933	1933	1933	1933	1933
Ta016	1892	1892	1892	1892	1892	1892	1892	1892
Ta017	1963	1963	1963	1963	1963	1963	1963	1963
Ta018	2057	2058.6	2057	2057	2057	2057	2057	2057
Ta019	1973	1973	1973	1973	1973	1973	1973	1973
Ta020	2051	2051	2051	2051	2051	2051	2051	2051
Ta021	2973	2973	2973	2973.8	2973	2973	2973	2973
Ta022	2852	2852	2852	2852	2852	2852	2852	2852
Ta023	3013	3019.4	3013	3013	3013	3019.4	3013	3014.3
Ta024	3001	3001	3001	3003.4	3001	3001	3001	3001
Ta025	3003	3003	3003	3003	3003	3003	3003	3003
Ta026	2998	2998	2998	2998	2998	2998	2998	2998
Ta027	3052	3052	3052	3052	3052	3052	3052	3052
Ta028	2839	2839	2839	2849.4	2839	2839	2839	2839
Ta029	3009	3009	3009	3009	3009	3009	3009	3009
Ta030	2979	2979	2979	2979	2979	2979	2979	2979
Ta031	3209	3222	3169	3178.4	3161	3162.4	3170	3171.8
Ta032	3469	3474.4	3444	3450.4	3440	3441	3441	3444.5
Ta033	3254	3262.8	3226	3234.4	3213	3216	3218	3231.8
Ta034	3366	3374.8	3348	3351.8	3343	3346.6	3349	3350.8
Ta035	3398	3406	3370	3374.8	3361	3364.6	3376	3376.7
Ta036	3371	3382.2	3354	3362.8	3346	3347.6	3352	3356.2
Ta037	3257	3267.6	3244	3250.8	3234	3235.8	3243	3245.3
Ta038	3266	3275.2	3247	3257.8	3241	3242.4	3239	3240.3
Ta039	3115	3123.2	3087	3092	3075	3078.8	3078	3086.8
Ta040	3373	3377.2	3336	3344.8	3322	3327.2	3330	3336.5
Ta041	4303	4311.2	4284	4298.6	4274	4276.8	4274	4281.5
Ta042	4197	4201	4193	4212.2	4179	4185	4180	4184.5
Ta043	4110	4124	4119	4128	4099	4107.6	4099	4105.5
Ta044	4432	4439.2	4411	4422.8	4399	4405	4407	4405.7
Ta045	4336	4347	4334	4342.8	4324	4330.4	4324	4324.7
Ta046	4312	4330	4311	4316.4	4290	4297.2	4294	4295.2
Ta047	4433	4441.4	4435	4447.6	4420	4429.6	4420	4420
Ta048	4353	4357.6	4331	4341	4321	4327	4323	4323.3
Ta049	4190	4194.8	4162	4169.2	4158	4164.2	4155	4161.7
Ta050	4299	4301	4287	4290	4286	4286.2	4286	4286.2
Ta051	6144	6154.6	6165	6178.8	6129	6139.6	6129	6138.5
Ta052	5748	5762.2	5730	5751.8	5725	5741	5725	5733.5
Ta053	5879	5907	5881	5898.6	5873	5882.4	5862	5865.5
Ta054	5797	5802.6	5802	5813.4	5789	5791.4	5789	5790.7
Ta055	5924	5930.6	5908	5923	5886	5899.4	5886	5893.5
Ta056	5904	5912.6	5886	5901.4	5874	5883.4	5871	5874.3
Ta057	6004	6012	5968	5991.4	5968	5974	5969	5974
Ta058	5947	5970.2	5937	5977.2	5940	5945.4	5926	5930.5
Ta059	5881	5900.6	5889	5908.6	5876	5883.2	5876	5876
Ta060	5970	5982	5959	5971.6	5959	5959	5958	5958.8
Ta061	6563	6586.4	6458	6464.6	6397	6413.4	6433	6438.3
Ta062	6409	6428.2	6268	6292.4	6246	6252.2	6268	6285.5
Ta063	6254	6292.8	6172	6185	6133	6135.8	6162	6164.2
Ta064	6173	6184.8	6071	6085.8	6028	6031.4	6055	6050.7
Ta065	6319	6358.4	6244	6261.2	6206	6217.8	6221	6223.3

(*continued on next page*)

Table A.1 (*continued*)

Instance	IIGA		$DPSO_{VND}$		TMIIG		DWWO	
	Best	Mean	Best	Mean	Best	Mean	Best	Mean
Ta066	6255	6269.2	6133	6149.4	6088	6096	6121	6122.7
Ta067	6432	6442.2	6292	6316	6254	6263.4	6311	6308.8
Ta068	6329	6338.6	6191	6209.6	6150	6156.6	6197	6210.8
Ta069	6552	6557.4	6412	6431.8	6391	6395	6418	6422
Ta070	6523	6561	6434	6451.6	6396	6401.6	6404	6427.2
Ta071	8209	8224.6	8105	8125	8080	8094.2	8093	8100.8
Ta072	8041	8061.4	7948	7968.4	7888	7909	7891	7938.3
Ta073	8136	8152	8084	8101.8	8042	8054.4	8047	8051
Ta074	8470	8484.2	8406	8420.6	8350	8362.4	8364	8378.2
Ta075	8065	8087.6	7992	8028.2	7967	7981.2	7966	7980
Ta076	7915	7930.4	7826	7850.8	7808	7821.6	7791	7809
Ta077	7943	7983.8	7914	7934.4	7880	7887.2	7881	7889
Ta078	8034	8051.4	7944	7971	7912	7924.8	7924	7931.7
Ta079	8294	8312.6	8201	8231.8	8164	8172	8152	8183.8
Ta080	8255	8263.6	8151	8190	8130	8148.8	8126	8141
Ta081	10,857	10,887	10,736	10,772.4	10,722	10,782.4	10,727	10,744
Ta082	10,734	10,755.6	10,661	10,714.6	10,611	10,623	10,604	10,608.7
Ta083	10,740	10,774.6	10,664	10,680.2	10,629	10,650	10,624	10,637
Ta084	10,750	10,765.8	10,669	10,713.6	10,615	10,647.2	10,615	10,629.8
Ta085	10,654	10,690.6	10,617	10,659.8	10,563	10,579.8	10,551	10,568.3
Ta086	10,789	10,819.4	10,719	10,740.6	10,684	10,696.2	10,680	10,690
Ta087	10,967	10,997.2	10,930	10,949	10,832	10,849.2	10,824	10,843.2
Ta088	10,997	11,011.6	10,896	10,919	10,846	10,862	10,839	10,850.2
Ta089	10,859	10,875	10,803	10,811	10,763	10,780.4	10,745	10,758
Ta090	10,933	10,944.6	10,858	10,881	10,797	10,817.2	10,787	10,808.5
Ta091	15,701	15,751.4	15,498	15,520.6	15,377	15,397.4	15,418	15,449.3
Ta092	15,538	15,572.2	15,276	15,333	15,167	15,203	15,252	15,281.3
Ta093	15,699	15,762.4	15,542	15,596	15,416	15,433.2	15,412	15,454.5
Ta094	15,612	15,625	15,394	15,413	15,250	15,279.8	15,304	15,304.7
Ta095	15,598	15,626.2	15,386	15,444.6	15,268	15,280.8	15,277	15,295.8
Ta096	15,535	15,557	15,311	15,334.8	15,163	15,189.6	15,222	15,235.5
Ta097	15,812	15,842	15,560	15,602.8	15,441	15,476.6	15,459	15,508.3
Ta098	15,646	15,685.6	15,395	15,468.4	15,295	15,319.2	15,307	15,340
Ta099	15,494	15,525.8	15,290	15,343.4	15,155	15,185.2	15,186	15,240.2
Ta100	15,724	15,754.4	15,489	15,534.4	15,382	15,410.8	15,378	15,412.2
Ta101	20,067	20,127.8	19,926	19,965.4	19,723	19,770.8	19,724	19,736.7
Ta102	20,506	20,536.6	20,230	20,285.8	20,127	20,148.8	20,091	20,118.8
Ta103	20,322	20,355.6	20,125	20,180	19,973	19,995.2	19,989	19,999
Ta104	20,324	20,349.6	20,060	20,137	19,997	20,032	19,940	19,967.2
Ta105	20,307	20,331.4	20,056	20,118.8	19,900	19,936	19,875	19,923.8
Ta106	20,367	20,407	20,141	20,198	20,003	20,050.6	19,980	20,011.3
Ta107	20,447	20,539.2	20,247	20,320.6	20,145	20,167.4	20,119	20,147.5
Ta108	20,434	20,449.8	20,250	20,293.6	20,053	20,095.6	20,053	20,084
Ta109	20,343	20,358.2	20,131	20,161	19,998	20,022.2	19,932	19,975.5
Ta110	20,268	20,284	20,140	20,190	19,932	19,968.6	19,916	19,946.3
Ta111	48,135	48,253.4	47,634	47,655.4	47,046	47,147.2	46,871	46,965.5
Ta112	48,658	48,699	48,037	48,104.8	47,630	47,692.2	47,294	47,429.8
Ta113	48,032	48,116.4	47,428	47,524.6	46,977	47,049.6	46,846	46,878.7
Ta114	48,390	48,513.2	47,810	47,954.4	47,328	47,375.8	47,185	47,232.8
Ta115	48,222	48,345.8	47,762	47,834.8	47,238	47,280	47,037	47,099.2
Ta116	48,628	48,701.6	48,066	48,129.2	47,553	47,591.2	47,166	47,331.5
Ta117	48,117	48,186	47,497	47,532.2	46,944	47,087.2	46,794	46,866.3
Ta118	48,411	48,468.2	47,761	47,907.6	47,346	47,421.4	47,127	47,248.7
Ta119	48,212	48,297.2	47,571	47,683.4	47,205	47,248.6	47,025	47,083.5
Ta120	48,396	48,483.4	47,907	47,951	47,374	47,402.4	47,105	47,183.2

References

Agnetis, A. (2000). Scheduling no-wait robotic cells with two and three machines. *European Journal of Operational Research, 123*(2), 303–314.

Akrout, H., Jarboui, B., Rebaï, A., & Siarry, P. (2013). New greedy randomized adaptive search procedure based on differential evolution algorithm for solving no-wait flowshop scheduling problem. In *International conference on advanced logistics and transport* (pp. 327–334). Sousse, Tunisia.

Aldowaisan, T., & Allahverdi, A. (2012). Minimizing total tardiness in no-wait flowshops. *Foundations of Computing & Decision Sciences, 37*(3), 149–162.

Bozorgirad, M. A., & Logendran, R. (2013). Bi-criteria group scheduling in hybrid flowshops. *International Journal of Production Economics, 145*(2), 599–612.

Carlier, J. (1978). Ordonnancements à contraintes disjonctives. *RAIRO – Operations Research, 12*(4), 333–350.

Davendra, D., Zelinka, I., Senkerik, R., & Jasek, R. (2011). Discrete self-organising migrating algorithm for flow shop scheduling with no wait makespan. *Mathematical & Computer Modelling, 57*(1–2), 100–110.

Ding, J. Y., Song, S. J., Gupta, J. N. D., Zhang, R., Chiong, R., & Wu, C. (2015). An improved iterated greedy algorithm with a Tabu-based reconstruction strategy for the no-wait flowshop scheduling problem. *Applied Soft Computing, 30*, 604–613.

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

Ferrer, A., Guimarans, D., Ramalhinho, H., & Juan, A. A. (2016). A BRILS metaheuristic for non-smooth flow-shop problems with failure-risk costs. *Expert Systems with Applications, 44*(C), 177–186.

Fink, A., & Voß, S. (2003). Solving the continuous flow-shop scheduling problem by metaheuristics. *European Journal of Operational Research, 151*(2), 400–414.

Gao, K. Z., Pan, Q. K., & Li, J. Q. (2011). Discrete harmony search algorithm for the no-wait flow shop scheduling problem with total flow time criterion. *International Journal of Advanced Manufacturing Technology, 56*(5-8), 683–692.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A Survey. *Annals of Discrete Mathematics, 5*(1), 287–326.

Hall, N. G., & Sriskandarajah, C. (1996). A survey of machine scheduling problems with blocking and no-wait in Process. *Operations Research, 44*(3), 510–525.

Huang, Y. M., & Lin, J. C. (2011). A new bee colony optimization algorithm with idle–time-based filtering scheme for open shop-scheduling problems. *Expert Systems with Applications, 38*(5), 5438–5447.

Iosifescu, M. (1980). *Finite Markov processes and their applications*. New York: Wiley.

Jarboui, B., Eddaly, M., & Siarry, P. (2011). A hybrid genetic algorithm for solving no-wait flowshop scheduling problems. *The International Journal of Advanced Manufacturing Technology, 54*(9), 1129–1143.

Johnson, S. M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly, 1*(1), 61–68.

Li, X., Wang, Q., & Wu, C. (2008). Heuristic for no-wait flow shops with makespan minimization. *International Journal of Production Research, 46*(9), 2519–2530.

Mehravaran, Y., & Logendran, R. (2013). Non-permutation flowshop scheduling with dual resources. *Expert Systems with Applications, 40*(13), 5061–5076.

Montgomery, D. C. (2001). *Design and analysis of experiment*. Phoenix, AZ: John Wiley and Sons.

Nagano, M. S., Silva, A. A. D., & Lorena, L. A. N. (2014). An evolutionary clustering search for the no-wait flow shop problem with sequence dependent setup times. *Expert Systems with Applications, 41*(8), 3628–3633.

Pan, Q. K., Fatih Tasgetiren, M., & Liang, Y. C. (2008a). A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. *Computers & Operations Research, 35*(9), 2807–2839.

Pan, Q. K., Wang, L., & Zhao, B. H. (2008b). An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion. *The International Journal of Advanced Manufacturing Technology, 38*(7), 778–786.

Potts, C. N., & Kovalyov, M. Y. (2000). Scheduling with batching: A review. *European Journal of Operational Research, 120*(2), 228–249.

Röck, H. (1984). The three machine no-wait flow shop is NP-complete. *Journal of the Acm, 31*(2), 336–345.

Raaymakersa, W. H. M. (2000). Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing. *European Journal of Operational Research, 126*(1), 131–151.

Rajendran, C. (1994). A no-wait flowshop scheduling heuristic to minimize makespan. *Journal of the Operational Research Society, 45*(4), 472–478.

Reeves, C. R. (1995). A genetic algorithm for flowshop sequencing. *Computers & Operations Research, 22*(1), 5–13.

Riahi, V., Khorramizadeh, M., Newton, M. A. H., & Sattar, A. (2017). Scatter search for mixed blocking flowshop scheduling. *Expert Systems with Applications, 79*, 20–32.

Rossi, A., Soldani, S., & Lanzetta, M. (2015). Hybrid stage shop scheduling. *Expert Systems with Applications, 42*(8), 4105–4119.

Ruizab, R. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research, 177*(3), 2033–2049.

Samarghandi, H., & ElMekkawy, T. Y. (2012). A meta-heuristic approach for solving the no-wait flow-shop problem. *International Journal of Production Research, 50*(24), 7313–7326.

Shabtay, D. (2012). The just-in-time scheduling problem in a flow-shop scheduling system. *European Journal of Operational Research, 216*(3), 521–532.

Shahvari, O., & Logendran, R. (2016). Hybrid flow shop batching and scheduling with a bi-criteria objective. *International Journal of Production Economics, 179*, 239–258.

Shahvari, O., Salmasi, N., Logendran, R., & Abbasi, B. (2012). An efficient tabu search algorithm for flexible flow shop sequence-dependent group scheduling problems. *International Journal of Production Research, 50*(15), 4237–4254.

Shao, W., & Pi, D. (2016). A self-guided differential evolution with neighborhood search for permutation flow shop scheduling. *Expert Systems with Applications, 51*, 161–176.

Shao, W., Pi, D., & Shao, Z. (2017). Memetic algorithm with node and edge histogram for no-idle flow shop scheduling problem to minimize the makespan criterion. *Applied Soft Computing, 54*, 164–182.

Siva, M., Balamurugan, R., & Lakshminarasimman, L. (2016). Water wave optimization algorithm for solving economic dispatch problems with generator constraints. *International Journal of Intelligent Engineering and Systems, 9*(4), 31–40.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research, 64*(2), 278–285.

Vallada, E., Ruiz, R., & Framinan, J. M. (2015). New hard benchmark for flowshop scheduling problems minimising makespan. *European Journal of Operational Research, 240*(3), 666–677.

Wang, C. (2014). Fast iterated local search algorithm with high order neighborhood for no-wait flowshop scheduling problem. In *Control and Decision Conference* (pp. 4460–4464). Changsha, China.

Wang, C., Li, X., & Wang, Q. (2010). Accelerated tabu search for no-wait flowshop scheduling problem with maximum lateness criterion. *European Journal of Operational Research, 206*(1), 64–72.

Wang, Z., Xing, W., & Bai, F. (2005). No-wait flexible flowshop scheduling with no-idle machines. *Operations Research Letters, 33*(6), 609–614.

Wu, X. B., Liao, J., & Wang, Z. C. (2015). Water wave optimization for the traveling salesman problem. In *International Conference on Intelligent Computing: 9225* (pp. 137–146). Fuzhou, China: Springer International Publishing.

Yun, X., Feng, X., Xin, L., Wang, S., & Liu, B. (2016). A novel water wave optimization based memetic algorithm for flow-shop scheduling. In *IEEE Congress on Evolutionary Computation* (pp. 1971–1976). Vancouver, BC, Canada.

Zhao, F., Tang, J., Wang, J., & Jonrinaldi, J. (2014). An improved particle swarm optimisation with a linearly decreasing disturbance term for flow shop scheduling with limited buffers. *International Journal of Computer Integrated Manufacturing, 27*(5), 488–499.

Zhao, F., Zhang, J., Wang, J., & Zhang, C. (2015a). A shuffled complex evolution algorithm with opposition-based learning for a permutation flow shop scheduling problem. *International Journal of Computer Integrated Manufacturing, 28*(11), 1220–1235.

Zhao, F., Zhang, J., Zhang, C., & Wang, J. (2015b). An improved shuffled complex evolution algorithm with sequence mapping mechanism for job shop scheduling problems. *Expert Systems with Applications, 42*(8), 3953–3966.

Zheng, Y.-J. (2015). Water wave optimization: A new nature-inspired metaheuristic. *Computers & Operations Research, 55*, 1–11.

Zheng, Y. J., Zhang, B., & Xue, J. Y. (2016). Selection of key software components for formal development using water wave optimization. *Journal of Software, 27*(4), 933–942.