Expert Systems With Applications 146 (2020) 113166

Contents lists available at ScienceDirect

Expert Systems With Applications

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

A hybrid discrete water wave optimization algorithm for the no-idle flowshop scheduling problem with total tardiness criterion

Fuqing Zhao[a,*], Lixin Zhang[a], Yi Zhang[b], Weimin Ma[c], Chuck Zhang[d], Houbin Song[a]

[a] School of Computer and Communication Technology, Lanzhou University of Technology, Lanzhou 730050, China
[b] School of Mechanical 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 21 October 2019
Revised 23 December 2019
Accepted 27 December 2019
Available online 28 December 2019

Keywords:
Water wave optimization
No-idle flowshop scheduling problem
Total tardiness
Variable neighborhood search

ABSTRACT

The no-idle flowshop has attracted enormous attention owing to its widespread application in the manufacturing industry domain. In this paper, a hybrid discrete water wave optimization algorithm, named HWWO, is presented to solve the NIFSP with total tardiness. In order to improve the quality of a population, an initialize method based on a new priority rule combined with the modified NEH method is proposed to generate a population. In the propagation phase, a self-adaption selection neighborhood search structure is introduced to amplify the search range of waves and balance the exploration and exploitation ability of the HWWO. Afterwards, a variable neighborhood search is adopted to strengthen the local search and maintain the diversity of the population in the breaking phase. In the refraction operation, a perturbation sequence is generated and combined with the local optimal solution found by the breaking operation, in order to generate a new solution, and prevent the algorithm from becoming trapped in the local optimum. Furthermore, the control parameters and time complexity are analyzed. The experimental results and comparisons with the other state-of-the-art algorithms evaluated on Taillard's and Ruiz's benchmark sets reveal that the effectiveness and efficiency of the HWWO outperformed the compared algorithms for solving the NIFSP.

1. Introduction

Intelligent manufacturing and energy consumption are the focus of current and future industrial development in modern manufacturing (Tasgetiren, Eliiyi, Öztop, Kizilay & Pan, 2018). Resource-efficient engineering, low-carbon economic solutions and green manufacturing for conserving resources, maximizing material recovery, reusing and recycling, and minimizing waste to respond to and actively prepare for major technological challenges of sustainable manufacturing (Ding, Song & Cheng, 2015). The no-idle permutation flowshop scheduling problem (NIFSP) (Billaut, Croce, Salassa & T'Kindt, 2019; Goncharov, 2009), as a classic combinational optimization problem, has been investigated by various researchers. In real life, the NIFSP is applied in foundry production (Saadani, Guinet & Moalla, 2003) and fiber glass processing (Kalczynski & Kamburowski, 2005) for certain economic, technical or environment reasons. Therefore, it is significant and important for real life.

The NIFSP is different from the other PFSP. For the NIFSP, each machine has to process jobs without any idle time from the start of processing the first job to the completion of the last job. Up to now, less research on the NIFSP has been performed than the other PFSP. The NIFSP is denoted as $F_m|prmu, no-idle|\sum T_j$ (Pinedo & Hadavi, 2016), where *prmu* means that the identical job sequence is maintained on each machine, T_j is the tardiness of a job j and $T_j = \max\{C_j - d_j, 0\}$, C_j is the completion time of job j on the last machine of all operations and d_j is the due date. The aim of total tardiness minimization is to find schedules, which satisfy the external due dates committed to customers (Tasgetiren, Pan, Suganthan & Chua, 2011). The two-machine no-idle constraint with the total completion time criterion for FSP first was proposed by Adiri and Pohoryles (1982). Afterwards, a number of methods were introduced by various scholars, such as exact methods (Baptiste & Hguny, 1997) and heuristic algorithms (Chakrabortty, Abbasi & Ryan, 2020), to solve different types of the NIFSP.

In past decades, evolutionary algorithms (EAs) for solving complex problems have a pivotal status. The representative EAs, such

* Corresponding author.
E-mail addresses: fzhao2000@hotmail.com (F. Zhao), 1577851713@qq.com (L. Zhang), 1049440546@qq.com (Y. Zhang), mawm@tongji.edu.cn (W. Ma), chuck.zhang@isye.gatech.edu (C. Zhang), 523712418@qq.com (H. Song).

https://doi.org/10.1016/j.eswa.2019.113166
0957-4174/© 2019 Elsevier Ltd. All rights reserved.

as the differential evolution algorithm (DE) (Ras, Wilke, Groenwold & Kok, 2019; Storn & Price, 1997), particle swarm optimization (PSO) (h$) have been widely applied in various domains. Meanwhile, hybrid evolutionary algorithms and certain novel algorithms, for instance, TDBBO (Zhao, Qin, Zhang, Ma & Song, 2018), SGSADE (Zhao et al., 2018) and Jaya (Rao, 2016), are proposed by numerous scholars. Various EAs are used to solve flowshop scheduling problems. For example, a discrete gravitational search algorithm (DSGA) (Zhao et al., 2019) for the blocking flowshop problem (BFSP) (Ronconi & Henriques, 2009) and a hybrid biogeography-based optimization (HBV) (Zhao et al., 2019) with a variable neighborhood search mechanism for the no-wait flowshop scheduling problem (NWFSP) (Allahverdi, Aydilek & Aydilek, 2018). Furthermore, the heuristics for solving the NIFSP are classified into two main categories, constructive heuristics and metaheuristics. A highly efficient simple constructive heuristic with total flowtime criterion was proposed by Nagano, Rossi and Martarelli (2019) and embedded in a high performance iterated greedy algorithm. A speed-up method, which is introduced for the evaluation of the insertion neighborhood to reduce computational complexity, is described in a novel hybrid discrete particle swarm optimization (HDPSO) algorithm proposed by Pan and Wang (2008). A variable iterated greedy algorithm (IG) with differential evolution (*vIG_DE*) was designed to solve the no-idle permutation flowshop scheduling problem and was proposed by Tasgetiren, Pan, Suganthan and Buyukdagli (2013).

Considering the afore-stated issue is of practical importance, the no-idle problem has been paid attention with various considerations. An iterated reference greedy (IRG) algorithm (Ying, Lin, Cheng & He, 2017) was proposed to solve the distributed no-idle permutation flowshop scheduling problem (DNIFSP) with the makespan criterion. A two-stage memetic algorithm (TSMA) (Jing-Fang, Ling & Jing-Jing, 2018) was proposed to minimize the makespan of the distributed no-idle permutation flowshop scheduling problem. A novel cloud theory-based iterated greedy (CTBIG) algorithm (Cheng, Ying, Chen & Lu, 2018) was proposed to solve the distributed mix no-idle permutation flowshop scheduling problem using makespan as an optimality criterion.

As a population-based meta-heuristic algorithm, the water wave optimization algorithm (WWO) (Zheng, 2015), which draws on the shallow water wave model, has obtained attention during a period of recent years. The WWO algorithm is iteratively evolved by three main phases including propagation, breaking and refraction operations to find the global optimum. The main idea of WWO is that low finesse waves with large wavelengths are applied for exploration in large regions and high fitness waves with small wavelengths are devoted to intensify a local search in small regions. The WWO algorithm has a desirable balance obtained by the mechanism between exploration and exploitation. In addition, WWO shows satisfactory performance in numerical tests and practical applications. At present, WWO and its variants have been applied to solve various optimization problems and practical issues. A novel discrete water wave optimization for solving the blocking flowshop scheduling problem (BFSP) with sequence-dependent setup times (SDST) is proposed by Shao, Pi and Shao (2017). A discrete water wave optimization algorithm is introduced by Zhao, Liu, Yi, Ma and Zhang (2017) to solve the no-wait flowshop scheduling problem with respect to the makespan criterion. A novel multi-objective discrete water wave optimization (MODWWO) algorithm is presented to solve a multi-objective BFSP (MOBFSP) (Shao, Pi & Shao, 2019) that minimizes both makespan and total flow time. Additionally, the WWO algorithm is also applied in software engineering (Zheng, Zhang & Xue, 2016) and optimization of bidding strategy (Hematabadi & Foroud, 2018). A water wave optimization algorithm with the single wave mechanism (SWWO) (Zhao et al., 2019) is proposed to solve the no-wait flowshop scheduling problem with an objective to minimize the makespan. However, the applications of WWO are still scarce due to the short time since the WWO was proposed. As seen from Fig. 1, a hybrid discrete water wave optimization algorithm for

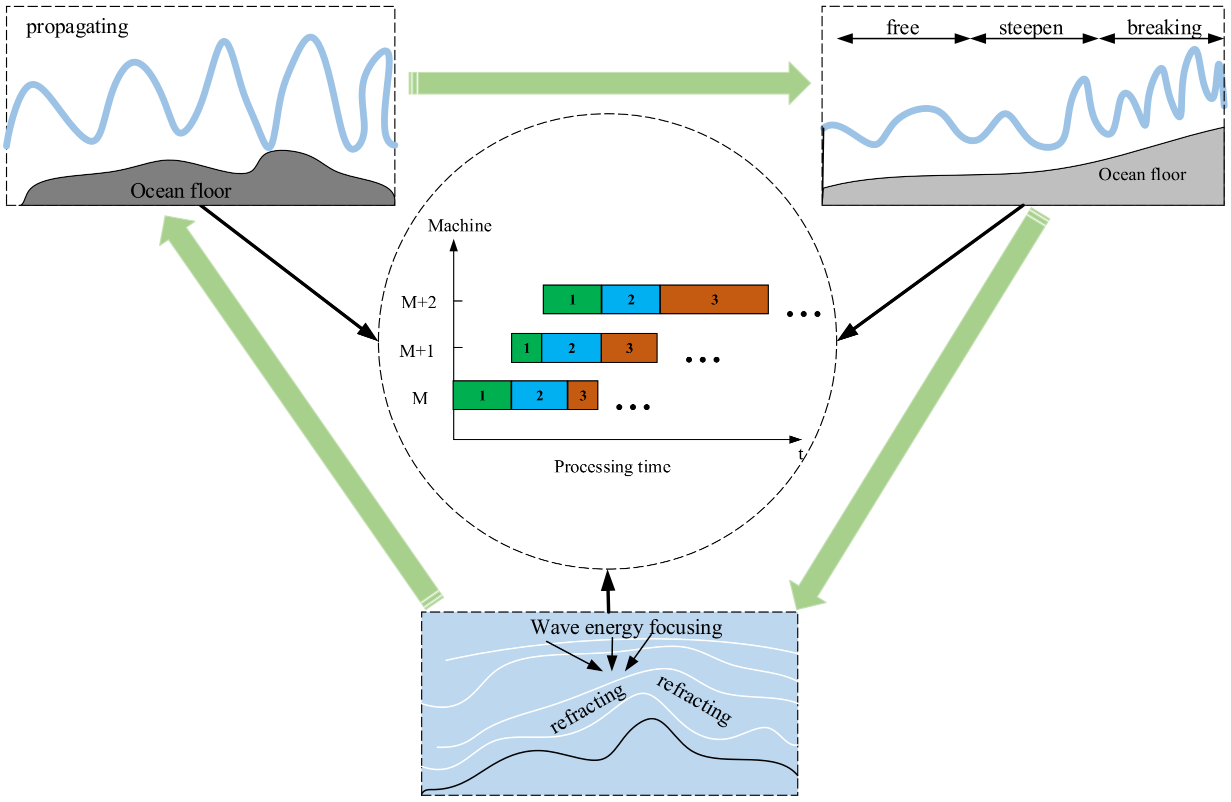


Fig. 1. Water Wave Optimization algorithm for the NIFSP.

solving the NIFSP with the aim of minimizing total tardiness is considered in this paper and there is no reported study on the WWO algorithm to solve the NIFSP. The proposed HWWO is composed by a new priority rule initialize method, a self-adaption propagation operator, a local reinforcement based breaking operator and the perturbation of refraction operator.

The contributions and innovation in this paper are summarized as follows.

- An initialize method based on a new priority rule and the modified NEH method, named $PR_{COV\_SKE}RNEH$, is proposed to enhance the quality of the initial population. A new priority rule is introduced to generate a desirable population. Afterwards, the inversus NEH is the inversion of the original NEH. Namely, the first two jobs in the sequence ordered by sum of the processing times are chosen in the NEH while the last two jobs in the sequence ordered by the priority rule are chosen to replace the first two jobs from the ordered sequence in the inversus NEH. In order to test the performance and stability of the initialize method for the proposed algorithm, the proposed initialize method is compared with the other three methods on the benchmark instances.
- The three operators of the WWO algorithm are redefined based on the original framework to solve $F_m|prmu, no-idle|\sum T_j$. In the propagation phase, a self-adaption selection neighborhood search strategy, which has eleven different neighborhood structures, is introduced to enlarge the search scope and implement the balance between exploration and exploitation. A local reinforcement method is presented in a breaking operation to exploit the area around the current best optimal and maintain the diversity of the population. A perturbation strategy as the refraction operation is applied to avoid the proposed algorithm being trapped in a local optimum.
- In order to verify the performance of the proposed algorithm, several parameters and the time complexity of the HWWO algorithm are analyzed. Furthermore, six state-of-the-art algorithms are compared with the HWWO algorithm to test the effectiveness of the proposed algorithm. All the compared algorithms are evaluated based on Ruiz's benchmark set and Taillard's benchmark set.

The remainder of the paper is organized as follows.

The problem statement of NIFSP is given in Section 2. In Section 3, the proposed HWWO algorithm is described in detail for $F_m|prmu, no-idle|\sum T_j$. In Section 4, several experiments are used to verify the performance of the proposed algorithm. Conclusions and future work are summarized in Section 5.

2. Problem statement

The symbols adopted in this paper are summarized in Table 1.

2.1. Description of the NIFSP

The no-idle flowshop scheduling problem (NIFSP) indicated in Fig. 2, which is a NP-hard problem (Ruiz & Maroto, 2005), is described as follows.

- There are n jobs processed on m machines according to an identical pipeline.
- The processing order of jobs on all machines is the identical.
- There is no idle time between adjacent jobs processed on the identical machine.
- A job is only processed on one machine at a certain time.
- A machine only processed one job at a certain time.
- The processing times of the jobs are known.
- The problem to be solved is determining how to arrange the permutation, which is guaranteed that the identical processing order of jobs is processed on each machine, to minimize total tardiness.

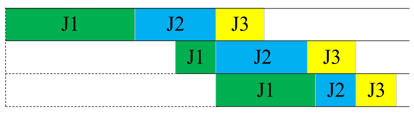


Fig. 2. A Gantt chart for a no-idle flowshop scheduling problem.

2.2. Formulations of the NIFSP

The NIFSP is formulated as a mixed integer programming model (Narain & Bagga, 2003) as follows.

$$Min\ TTD(\pi) \tag{1}$$

$$\sum_{k=1}^{n} X_{j,k} = 1, j \in \{1, 2, \ldots, n\} \tag{2}$$

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

$$C_{1,1} = \sum_{j=1}^{n} X_{j,1} \cdot p_{j,1} \tag{4}$$

$$C_{k+1,i} = C_{k,i} + \sum_{j=1}^{n} X_{j,k+1} * p_{j,i},\ \ k \in \{1, 2, \ldots, n-1\}, i \in \{1, 2, \ldots, m\} \tag{5}$$

$$C_{k,i+1} \geq C_{k,i} + \sum_{j=1}^{n} X_{j,k} * p_{j,i+1},\ \ k \in \{1, 2, \ldots, n\}, i \in \{1, 2, \ldots, m-1\} \tag{6}$$

$$C_{k,i} \geq 0,\ \ k \in \{1, 2, \ldots, n\}, i \in \{1, 2, \ldots, m-1\} \tag{7}$$

$$X_{j,k} \in \{0, 1\},\ \ j, k \in \{1, 2, \ldots, n\} \tag{8}$$

Constraint (2) and Constraint (3) show that all jobs have to appear only once in sequence π. Constraint (4) illustrates that the completion time of the first job processed on the first machine ensures that the first machine starts at zero. Constraint (5) defines that the relationship between the completion time of two adjacent jobs processed on the identical machine ensures that the machine has no idle time. Constraint (6) explains that the relationship between the completion time of two adjacent sequences of the same job ensures that one job is not machined by multiple machines at the identical time. Constraint (7) limits the completion time of all operations to greater than zero. Constraint (8) confines a limit on the value of a decision variable.

In this paper, total tardiness (TTD) as an evaluation criterion is calculated as follows (Tasgetiren, Buyukdagli, Pan & Suganthan, 2013).

$$F\left(\pi_1^E, k, k+1\right) = p(\pi(1), k+1), k = 1, 2, \ldots, m-1 \tag{9}$$

$$\begin{aligned} F\left(\pi_j^E, k, k+1\right) = \max &\left\{F\left(\pi_{j-1}^E, k, k+1\right) - p(\pi(j), k), 0\right\} \\ &+ p(\pi(j), k+1) \\ &j = 2, 3, \ldots, n \ and\ k = 1, 2, \ldots, m-1 \end{aligned} \tag{10}$$

$$C(\pi(n), m) = C_{max}\left(\pi_n^E\right) = \sum_{k=1}^{m-1} F\left(\pi_n^E, k, k+1\right) + \sum_{j=1}^{n} p(\pi(j), 1) \tag{11}$$

Table 1
Parameters and variables of the mathematical model.

Parameters	
n	The number of jobs.
m	The number of machines.
S	The sequences corresponding to the population.
NP	The size of the population.
p_{ij}	Processing time of job i on machine j
a	A random number.
Duetime	The due dates.
h	The wave height.
h_i	The wave height of the ith sequence.
λ	The wave length named *lambda*.
λ_{max}	The maximum wave length.
λ_i	The wave length of the ith sequence.
τ	The tightness factor.
Computing variables	
C_{ij}	Completion time for job i on machine j.
TTD	Total tardiness.

Table 2
Processing times of NIFSP example with three machines and four jobs.

	Jobs(i)			
Machines(j)	1	2	3	4
1	3	2	2	1
2	3	1	2	2
3	2	2	2	4

$$C(\pi(j), m) = C(\pi(j+1), m) - p(\pi(j+1), m),\quad j = n-1, n-2, \ldots, 1 \tag{12}$$

$$TTD = \sum_{j=1}^{n} (\max(C(\pi(j), m) - d(\pi(j)), 0)) \tag{13}$$

where $F(\pi_j^E, k, k+1)$ represents the completion time difference between machine k and machine $k+1$ while the sequence π_j^E is processed. $C(\pi(j), m)$ denotes the completion time of job $\pi(j)$ on the last machine m.

An example problem with three machines and four jobs is utilized to expediently understand the no-idle constraints. The processing times $p_{i,j}$ are exhibited in Table 2.

As seen from Fig. 3(a) and (b), the processing times of $J_1J_2J_3$ and J_4 on each machine is $J_1 = \{3, 3, 2\}$, $J_2 = \{2, 1, 2\}$, $J_3 = \{2, 2, 2\}$, $J_4 = \{1, 2, 4\}$. The sum of the processing times for $J_1J_2J_3$ and J_4 are {8, 6, 6, 7}, respectively. The *TTD*s of $\pi_1 = \{J_1J_2J_3J_4\}$ and $\pi_2 = \{J_2J_1J_3J_4\}$ are 20 and 14, respectively. Fig. 3. illustrates that the first job is fairly important for the sequence ordered. Therefore, constructive heuristics and meta-heuristics are applied to optimize sequences for relatively reasonable processing times and benefits.

3. The proposed algorithm for the NIFSP

Based on the original WWO algorithm (Zheng, 2015), a hybrid discrete water wave optimization algorithm is proposed for solving the NIFSP. The proposed algorithm, named HWWO, consists of four parts, a new priority rule to initialize the population, self-adaption selection propagation, a variable neighborhood search embedded into breaking and a perturbation refraction. Each component of the HWWO algorithm is detailed in the following subsections. Afterwards, the entire framework of the HWWO algorithm is described in the last subsection.

3.1. New priority rule to initialize the population

The heuristic method quickly completes the construction of the solution and the quality of the population is usually undesirable. In the HWWO algorithm, individuals are generated and each individual corresponds to a sequence $\pi = [\pi_1, \pi_2, \ldots \pi_n]$, which is a solution of the NIFSP. Inspired by the original NEH heuristic (Nawaz, Enscore & Ham, 1983; Ruiz, Maroto & Alcaraz, 2009) and the PR_{SKE} (Liu, Yan & Price, 2017), $PR_{COV\_SKE}RNEH$ (MNEH1) is proposed as the initialization method. In the original NEH, all jobs are ordered by a suitable method at first and inserted one by one. Different from the original NEH, the last two jobs as the first two jobs are inserted in the sequence to improve the diversity of the population in the $PR_{COV\_SKE}RNEH$(MNEH1). There are four initialization methods, including *NEH*, $PR_{SKE}NEH$, $PR_{COV\_SKE}RNEH$(MNEH1) and $PR_{SKE}RNEH$(MNEH2), presented in this paper. $PR_{SKE}NEH$ is priority rule combined with *NEH* and the priority rule is the sequence ordered according to *Eq*(16). $PR_{SKE}RNEH$(MNEH2) is an initialization method based on the $PR_{SKE}NEH$ and the sequence ordered according to $PR_{COV\_SKE}RNEH$(MNEH1). The different between the $PR_{COV\_SKE}RNEH$(MNEH1) and $PR_{SKE}RNEH$(MNEH2) is the calculation methods of processing times for jobs. Furthermore, the speed-up method for the insert neighborhood proposed by Tasgetiren et al. (2013) is applied in all methods. All jobs are ordered according to the following formulas for the $PR_{COV\_SKE}RNEH$.

$$AVG_j = \frac{1}{m} * \sum_{i=1}^{m} p_{i,j} \tag{14}$$

$$STD_j = \sqrt{\frac{1}{m-1}\left(\sum_{i=1}^{m} \left(p_{j,i} - AVG_j\right)^2\right)} \tag{15}$$

$$SKE_j = \frac{\frac{1}{m}\sum_{i=1}^{m} \left(p_{j,i} - AVG_j\right)^3}{\left(\sqrt{\frac{1}{m}\left(\sum_{i=1}^{m} \left(p_{j,i} - AVG_j\right)^2\right)}\right)^3} \tag{16}$$

$$COV_j = \frac{STD_j}{AVG_j} \tag{17}$$

$$Sum = AVG_j + STD_j + abs\left(SKE_j\right) \tag{18}$$

$$Sum = AVG_j + STD_j + COV_j * abs\left(SKE_j\right) \tag{19}$$

where AVG_i is the average processing times of job j, STD_j represents the standard deviation of the processing times for job j, COV_j stands for the coefficient of variation of job j and $abs(SKE_j)$ is delegated the absolute of the skewness of job j.

The coefficient of variation is an indicator to measure the dispersion of a data set and the skewness, which is positive or negative, is a measure of asymmetry of a probability distribution. The role of $COV_j{}^*abs(SKE_j)$ as the third moment is embedded in the priority rule to distinguish jobs. The average and standard deviation stand for the first and second moments of the processing time distribution, respectively. These moments characterize the processing time distribution accurately and are used for job differentiation effectively. $COV_j{}^*abs(SKE_j)$ is applied to immensely enhance the distinction of jobs to obtain a desirable initial sequence. For example, if there are two jobs with processing time [1,3,3,5,6,8,9] and [1,2,4,5,7,7,9]. The average, standard deviation and coefficient of variations of two jobs are identical (5, 2.67, 1.87), and the *skewnesses* of the two jobs are 0.1164 and −0.1163. Therefore, the $COV_j{}^*abs(SKE_j)$ is proposed to clearly distinguish the two jobs and generate a desirable solution. The pseudocode of the proposed initialization method is presented in Procedure 1.

Procedure 1 Initialization.

```
Input $P$, $NP$, $Duetime$, $S$
1 Calculate $AVG_i$, $STD_i$, $COV_i$ and $SKE_i$, respectively.
2 Calculate the sum of $AVG_i$, $STD_i$, $COV_i$ and $SKE_i$ according from Eq. (14) to Eq. (17)
3 All jobs are ordered according to the value of Eq. (19) to obtain $\pi_0$.
4 for $i = 1$: $NP$ do
5     $[S_i, fitness_i] = PR_{COV\_SKE}RNEH(\pi_0)$
6 end for
7 Output $S$, $fitnesses$
```

3.2. Propagation based on self-adaption selection

According to the basic framework of the WWO, new waves generated are determined by the propagation operator and obtain an offspring wave in the immediate neighborhood of the current wave. Therefore, the propagation operator has a significant impact on the basic WWO algorithm, and to a great extent, has immense influence on the exploration of the algorithm. In this section, a propagation operator based on self-adaption selection of the neighborhood structure is designed to achieve a balance between exploration and exploitation, and to access a desirable offspring wave. The pseudocode of the proposed propagation operation is shown in Procedure 2.

Procedure 2 Self-adaption Selection Propagation operation.

```
Input $jobIndex_i$, $fitness_i$, $P$, $Duetime$, $r(i)$
1 [$jobIndex0$, $fitness0$] = $neighborhoodsearchstrategy$(Procedure 3)
2 Output $jobIndex0$, $fitness0$
```

For the self-adaption selection of the neighborhood structure, as seen from Procedure 3, proposed in this paper, insert and swap operators as well as the destruction and construction operators (Ruizab, 2007) are primarily applied to yield proximity neighboring waves and enrich the neighborhood structure to enhance the exploration of the proposed algorithm. The details are shown in

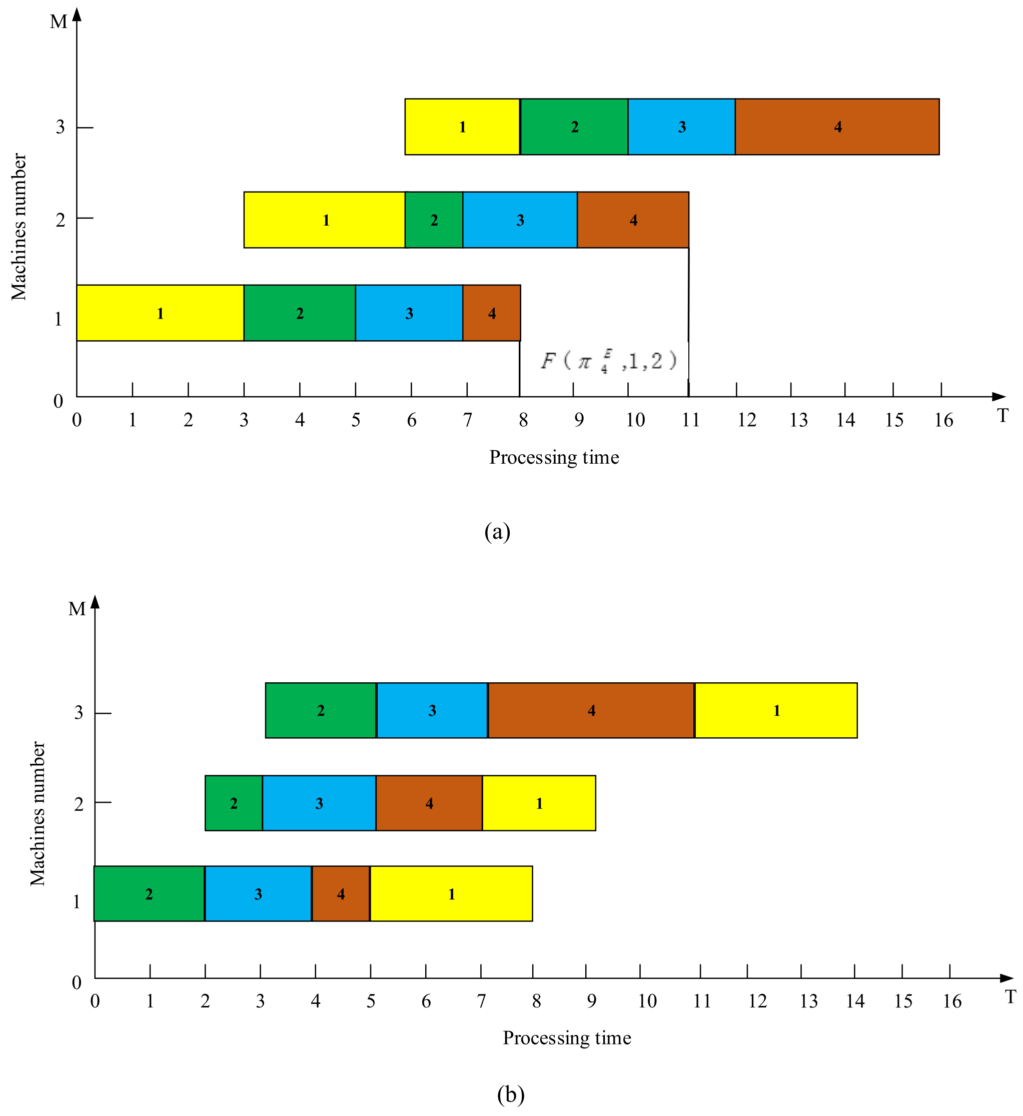


Fig. 3. An example for the no-idle flowshop scheduling problem.

Procedure 3 *neighborhoodsearchstratergy*.

```
Input P, Duetime, $jobIndex_1$, $r(i)$
1 for $i = 1: r$ do
2 $\pi = jobIndex_1$
3   switch ($R = unidrnd(11)\%12$)
4     case 1:$\pi_0 = shiftmutation(\pi)$
5     case 2:$\pi_0 = swapmutation(\pi)$
6     case 3:$\pi_1 = shiftmutation(\pi)$, $\pi_0 = shiftmutation(\pi_1)$
7     case 4:$\pi_1 = swapmutation(\pi)$, $\pi_0 = swapmutation(\pi_1)$
8     case 5:$\pi_1 = shiftmutation(\pi)$, $\pi_0 = swapmutation(\pi_1)$
9     case 6:$\pi_1 = swapmutation(\pi)$, $\pi_0 = shiftmutation(\pi_1)$
10    case 7:$\pi_1 = shiftmutation(\pi)$,$\pi_2 = shiftmutation(\pi_1)$,$\pi_0 = shiftmutation(\pi_2)$
11    case 8:$\pi_1 = swapmutation(\pi)$,$\pi_2 = swapmutation(\pi_1)$,$\pi_0 = swapmutation(\pi_2)$
12    case 9: $\pi_1 = shiftmutation(\pi)$,$\pi_2 = shiftmutation(\pi_1)$,$\pi_0 = swapmutation(\pi_2)$
13    case 10:$\pi_1 = swapmutation(\pi)$,$\pi_2 = swapmutation(\pi_1)$,$\pi_0 = shiftmutation(\pi_2)$
14    case 11:$\pi_0 = Destruction - Construction(\pi)$
15  end switch
16 end for
17 Calculate fitness of $\pi_0$
18 Output $\pi_0$, fitness
```

Fig. 4. For each strategy for the generation of neighboring waves it is possible to have a different performance during the iteration process. Thus, each wave, which is controlled by wavelength in the population, settles on one of the eleven strategies to generate a neighboring wave. Similar to the basic WWO, the fitness of a new solution generated by the propagation operator is compared with the corresponding fitness of the wave and the historical best solution. A new wave is always accepted if the new fitness is better than that of the other two individuals. The motivation to assign one of eleven strategies for each wave in the population is that high fitness waves propagate offspring waves in a little area to implement exploitation and low fitness waves generate offspring waves in a wide region to achieve exploration. The important significance of the proposed strategy is that each wave in the population is chosen for the appropriate strategy according to the fitness to generate offspring to satisfy the requirements of the basic propagation operation under the control of wavelength. Based on the design above, a desirable balance between exploration and exploitation is provided by the proposed propagation for the HWWO algorithm.

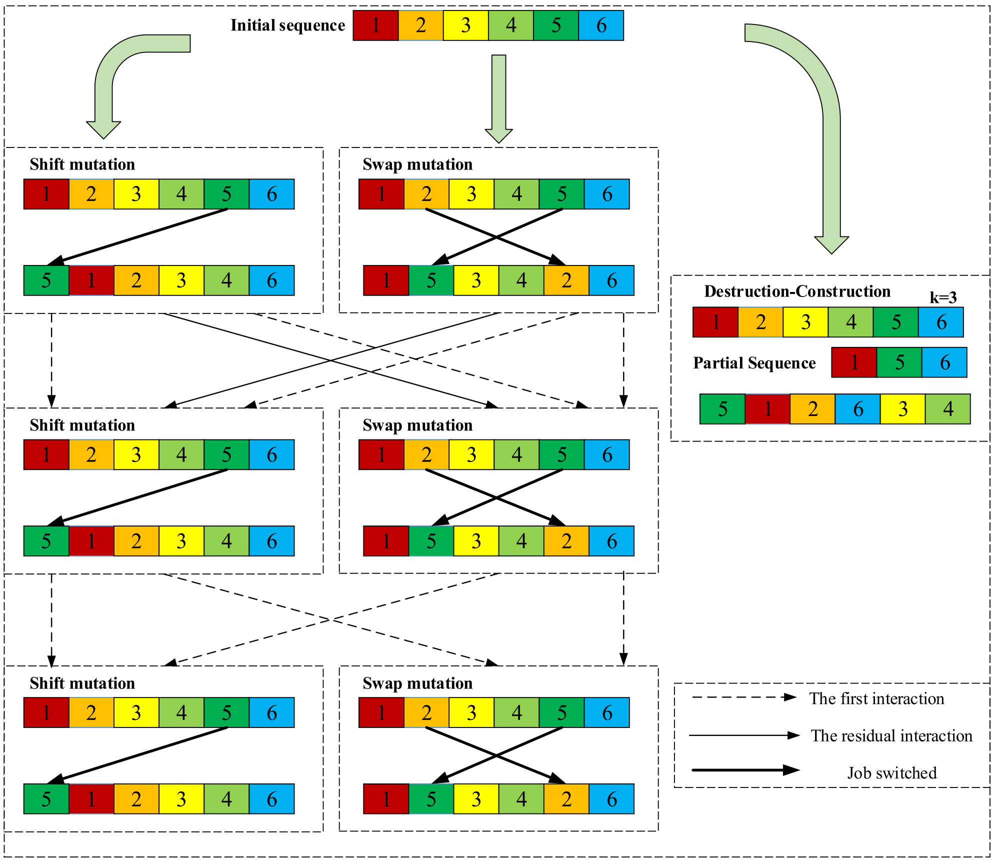


Fig. 4. The self-adaption selection of neighborhood structure for the Propagation operation.

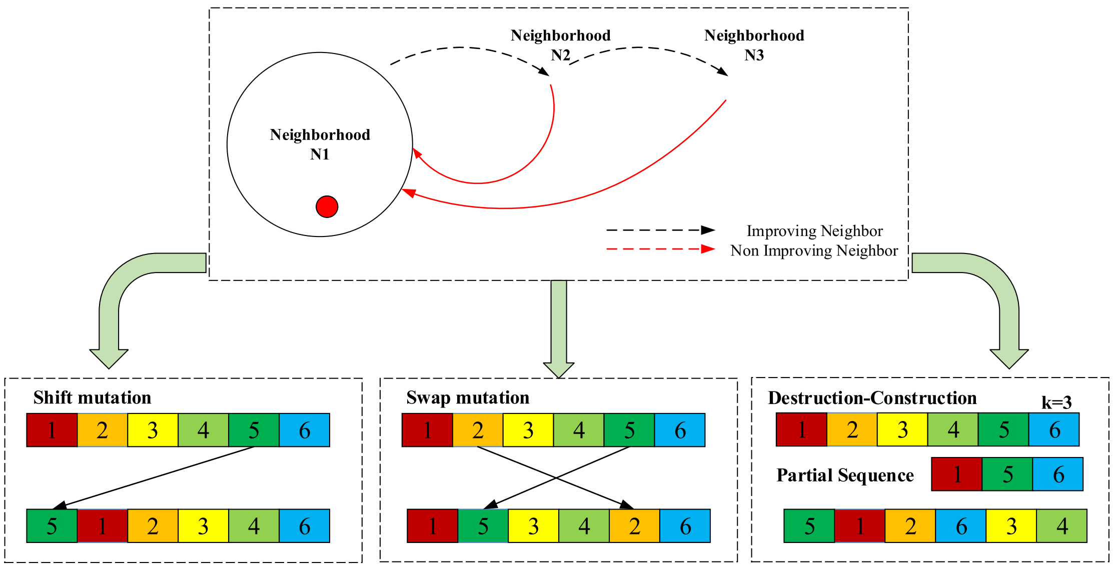


Fig. 5. The local intensification search for the Breaking operation.

3.3. Local reinforcement based breaking

When the peak velocity exceeds the wave velocity, the wave splits into a series of solitary waves (Zheng, 2015). In the original WWO, the breaking operation is executed as a new best solution is found by a wave and the role of the breaking operation is to exploit the region around the current wave. In this paper, the local intensification search is provided by the breaking operator only when a new best solution is found in the original WWO algorithm. Variable neighborhood search (VNS) (Hansen & Mladenović, 2005), which is a common approach to enhance the solution quality by systematic changes of neighborhoods, is introduced to achieve an optimal solution and maintain the diversity of population. Afterwards, a local search is applied to further enhance the quality of solutions with a probability of $a < 0.01$. The details of the search are shown in Procedure 4.

Procedure 4 Local reinforcement for Breaking operation.

```
1 Input P, Duetime, jobIndex$_1$
2 [jobIndex2, fitness2] = VNS(Procedure 5)
3 If $a < 0.01$
4     [sequence, fitness] = IterativeLocalsearchspeedup(jobIndex0, fitness0)
5   If fitness < fitness2
6     jobIndex2 = sequence;
7   end If
8 end If
9 Output $\pi$
```

There are three problem specific decisions obtained by a valid VNS, including the types of neighborhoods, the order of neighborhoods and the way neighborhoods are altered in the search process (Kirlik & Oguz, 2012). For the property of breaking, the three neighborhood structures of VNS, which is exhibited in Procedure 5, are introduced to establish the types of neighborhoods and enhance the convergence of the proposed algorithm. The sketch map of the breaking operation is demonstrated in Fig. 5. VNS is executed on the current best solution, if the fitness of sequence π_s generated by the VNS is smaller than the fitness of the current best π, π is replaced by π_s. k_{max} is set to 3, the current best π is switched among three structures to enhance the diversity of the population and the exploitation of HWWO. Furthermore, the diversity of the population resulting from the proposed initialize method is addressed.

Procedure 5 *VNS.*

```
1 Input the current solution $\pi$ required to search
2 $k_{max} = 3$, $k = 1$, $bestL2 = fitness_\pi$
3 while $k \le k_{max}$ do
4   switch $k$ do
5     case 1:$\pi_s = shiftmutation$
6       case 2:$\pi_s = swapmutation$
7       case 3:$\pi_s = Destruction - Construction$
8     end switch
9     if $fitness_{\pi_s} < bestL2$
10        $bestL2 = fitness_{\pi_s}$,$k = 1$,$\pi = \pi_s$
11      else $k = k + 1$ endif
12 end while
13 Output $\pi$
```

3.4. The perturbation of refraction

In the refraction operation, a low energy individual is removed from the population and the stagnation of the search is effectively avoided. The convergence of the algorithm to some extent is accelerated by the method, which is the original poor individual replaced by the new individual generated and obtained from the best individual. Based on this idea, a refraction operator perturbation is designed as seen from Procedure 6. The perturbation operation, which has two swap moves, is proposed by Intellektik, Darmstadt and Stutzle (1998). The first way is that a random job is selected and swapped with its following one. The second way is that two jobs are selected at random to swap. To be specific, when the height h of a wave π_i is equal to zero, a plain perturbation operator is executed on the current best solution and the historical best solution to generate a neighboring solution *pers*. Afterwards, the *Destruction–Construction* operator (Ruizab, 2007) is performed on *pers* to generate a desirable solution $\pi_{current}$, if the fitness of

Procedure 6 Refraction operation.

```
1 Input P, Duetime, hbs, $\pi_i$
2 A perturbed sequence pers is generated by hbs and $\pi_i$.
3 $\pi_{current} = Destruction - Construction(pers)$.
4 Calculate fitness of $\pi_{current}$.
5 Select the optimal fitness value by comparing the fitness value.
6 Output $\pi_{best}$
```

Table 3
Comparison of the ARPD for *NEH*, $PR_{SKE}NEH$(PRNEH), $PR_{SEK}RNEH$(MNEH2) and $PR_{COV\_SKE}RNEH$(MNEH1).

	$\tau=1$				$\tau=2$				$\tau=3$			
*n*m*	*MNEH1*	*MNEH2*	*PRNEH*	*NEH*	*MNEH1*	*MNEH2*	*PRNEH*	*NEH*	*MNEH1*	*MNEH2*	*PRNEH*	*NEH*
20*5	**3.122**	4.868	5.063	4.865	**4.508**	6.391	6.266	6.077	**6.731**	8.7616	12.022	10.392
20*10	**4.051**	4.409	5.058	5.086	**4.365**	10.665	12.577	12.386	**7.331**	19.283	24.007	17.403
20*20	**2.525**	3.362	3.432	4.634	**6.363**	10.313	8.529	9.371	**8.110**	16.426	27.970	28.837
50*5	**1.692**	2.399	2.212	2.097	**2.582**	3.535	4.045	2.621	**1.967**	1.975	2.256	2.052
50*10	**2.074**	2.725	3.047	2.643	4.345	4.403	4.024	**3.968**	3.930	4.069	3.732	**3.609**
50*20	**1.862**	2.403	2.094	3.483	**3.637**	4.142	4.829	4.846	**3.253**	3.597	5.024	3.282
100*5	**1.758**	1.769	1.789	2.171	**1.857**	2.706	2.761	1.951	1.043	1.092	**1.035**	1.101
100*10	**1.327**	2.126	2.185	2.149	**1.585**	2.587	1.659	2.346	**1.107**	1.410	1.412	1.130
100*20	**1.334**	1.705	1.769	1.643	2.502	2.578	**2.232**	2.277	**1.623**	1.652	1.915	1.744
200*10	**1.055**	1.177	1.375	1.150	**1.032**	1.631	1.481	1.048	**0.648**	0.789	0.688	0.698
200*20	**0.937**	1.226	1.270	1.386	**1.424**	1.709	1.515	1.563	**1.029**	1.041	**1.029**	1.045
500*20	**0.672**	0.858	1.063	0.973	**0.911**	2.380	2.191	3.416	**0.569**	0.582	1.350	0.768
Average	**1.993**	2.419	2.530	2.690	**3.617**	4.420	4.342	4.322	**4.695**	5.057	6.870	6.005

$\pi_{current}$ is better than π_i, π_i is replaced by the $\pi_{current}$. In fact, the authentic idea of the refraction operator is that a promising solution uses the knowledge acquired in the evolutionary search. The search region around $\pi_{current}$ is the most likely area to obtain the optimal. Therefore, perturbation and *Destruction−Construction* operators are employed to replace solutions that have fallen into the local optimal.

3.5. The framework of HWWO

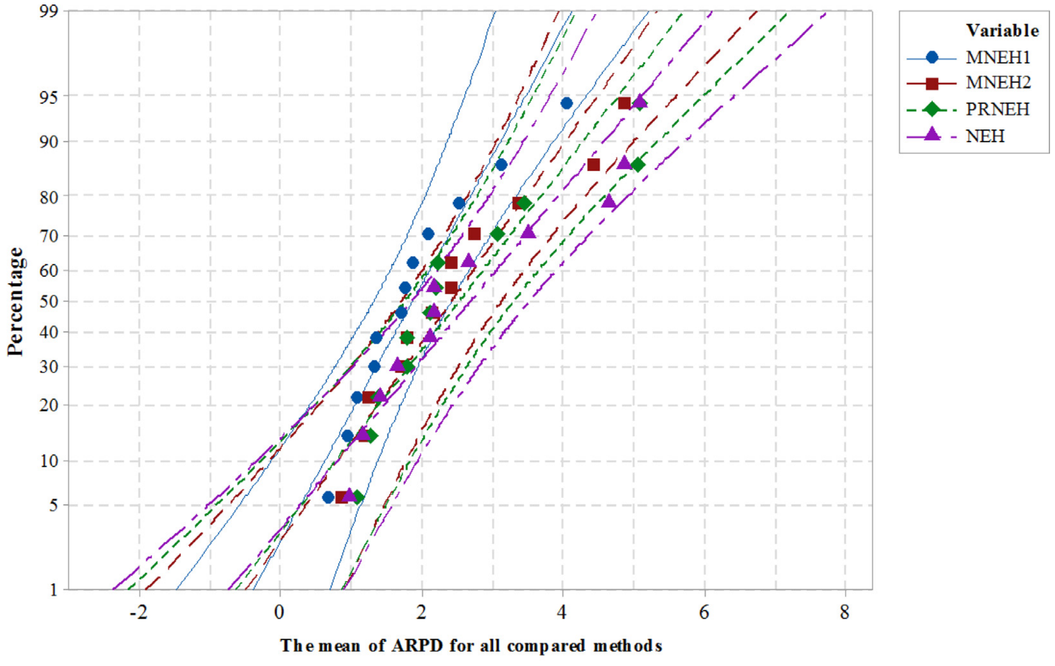
After the above discussions about each component, the general framework of the proposed algorithm is described in Algorithm 1. According to the pseudocode of the algorithm, four main sections are presented in the HWWO. In the initialization stage, a new priority rule of processing time is combined with $PR_{COV\_SKE}RNEH$ to generate an initial population with high-quality and diversity. A self-adaption selection strategy is applied in the propagation phase. The role of the propagation operator, controlled by the wavelength, is to balance the exploration and exploitation of HWWO. Afterwards, a variable neighborhood search as an intensified mechanism is embedded into the breaking phase to detect the waves around the current solution. Furthermore, a perturbation is performed for waves to avoid falling into the local optima while all wavelengths are zero. Combined with the above methods, the proposed algorithm is expected to achieve a desirable solution and performance.

4. Analysis and comparisons

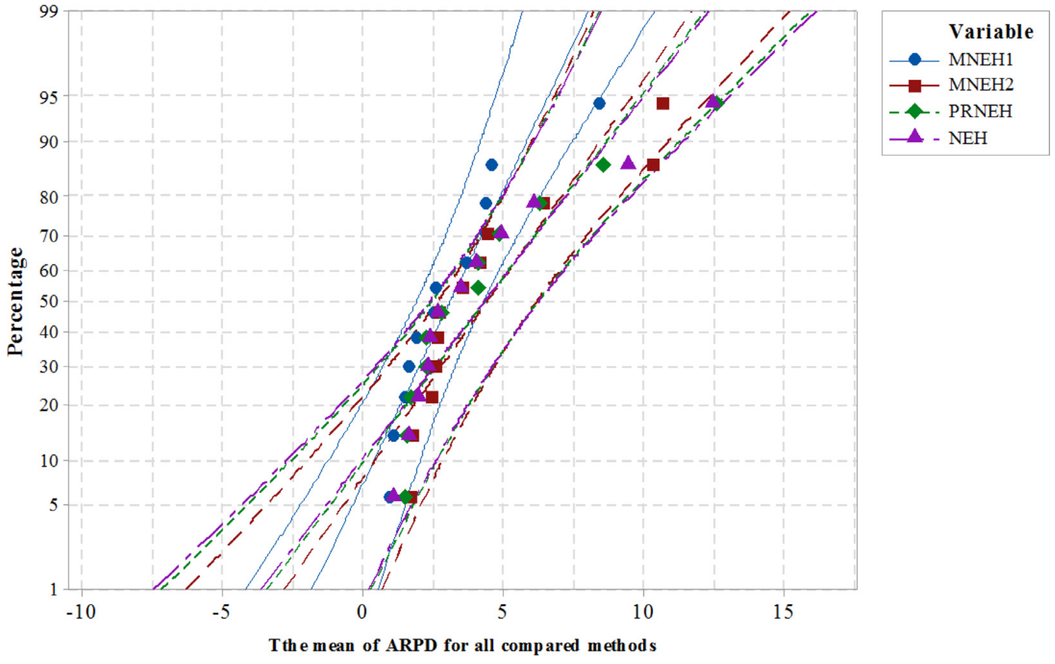
The computational complexity of the proposed algorithm, the analysis of the three operators, the analysis of all parameters and

Algorithm 1 The pseudocode of the HWWO.

```
Input: $NP$, $\lambda_{max}$,$\lambda_{min}$,$h$, $a$
Output: the optimal $TTD$
1 [$S$, $fitnesses$] = Procedure 1. Initialization
2 Record the historical best solution $hbf$ and the historical best sequence $hbs$.
3 while termination condition do
4   for $i = 1$ to $NP$ do
5     [$jobIndex1$,$fitnesses_1$]=Procedure 2. Propagation
6     If $fitnesses_1 < fitnesses_i$
7       $fitnesses_i = fitnesses_1$,$S_i = jobIndex1$
8     If $fitnesses_1 < hbf$
9     $hbf = fitnesses_1$;$hbs = \pi_1$;
10      [$jobIndex2$,$fitnesses_2$]=Procedure 4. Breaking
11        If $fitnesses_2 < hbf$
12          $hbf = fitnesses_2$; $hbs = jobIndex2$;
13          $fitnesses_i = fitnesses_2$; $S_i = jobIndex2$
14        else
15          $fitnesses_i = fitnesses_1$,$S_i = jobIndex1$
16        end If
17      else
18        $fitnesses_i = fitnesses_1$,$S_i = jobIndex1$
19      end If
20      $h(i) = h(i) - 1$
21      If $h(i) = 0$
22        [$jobIndex3$,$fitnesses_3$]=Procedure 6. Refraction
23        If $fitnesses_3 < hbf$
24          $hbf = fitnesses_3$;$hbs = jobIndex3$;
25        end If
26      $fitnesses_i = fitnesses_3$,$S_i = jobIndex3$
27      else
28        $h = h - 1$
29        Eliminate the inferior solution operation ($a < 0.02$).
30      end If
31    Update wavelength
32    end for
33 end while
```



(a) $\tau = 1$



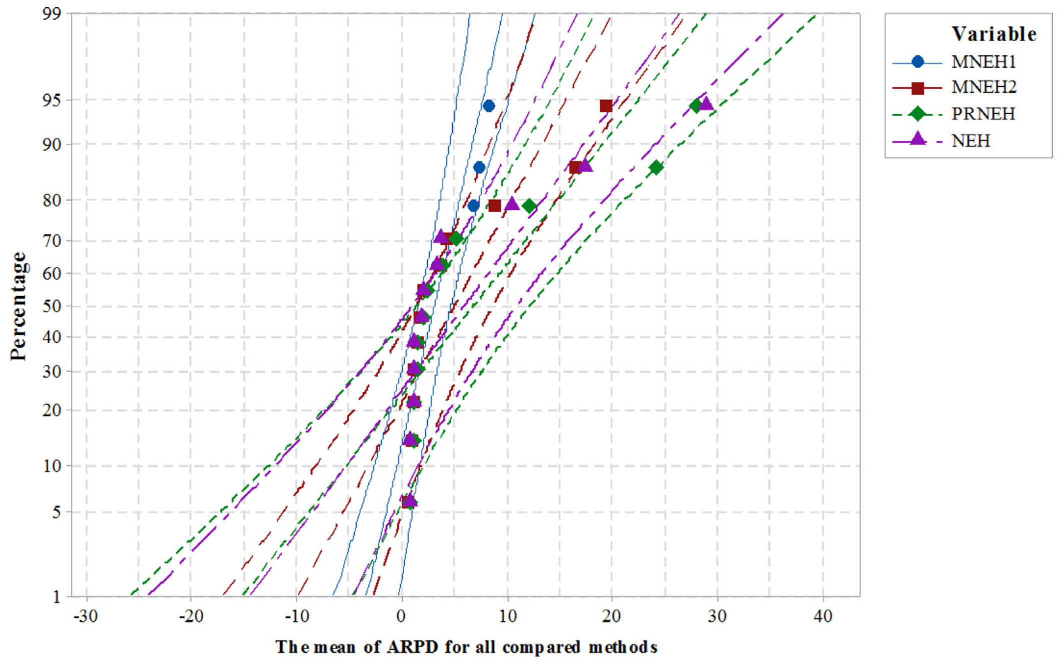
(b) $\tau = 2$

Fig. 6. Probability plot for NEH, PRNEH, MNEH2 and MNEH1with different τ.

the experimental results obtained by all compared algorithms for solving the NIFSP with total tardiness criterion are described in this section. Moreover, the HWWO algorithm and the other compared algorithms for solving the NIFSP with total tardiness criterion are coded using Java and MATLAB. The simulation experiments are carried out on a personal computer (PC) with Intel (R) Core (TM) i7-6700 CPU 3.4 GHz and 8.00GB memory with the Windows Server 2012 Operating System. The specific descriptions are as follows.

4.1. Computational complexity of the HWWO

The computational complexity of the HWWO is analyzed in this section. There are n jobs and m machines. The size of the population is set to NP. In the initial phase, NP individuals are generated by the proposed initialize method and the time complexity of initialization is $O(NP^{*}mn^{2})$. As shown in Procedures 2. and 3, the time complexity of propagation is $O(NP^{*}n + r^{*}mn)$. The time complexity of propagation is mainly determined by the self-



(c) $\tau = 3$

Fig. 6. Continued

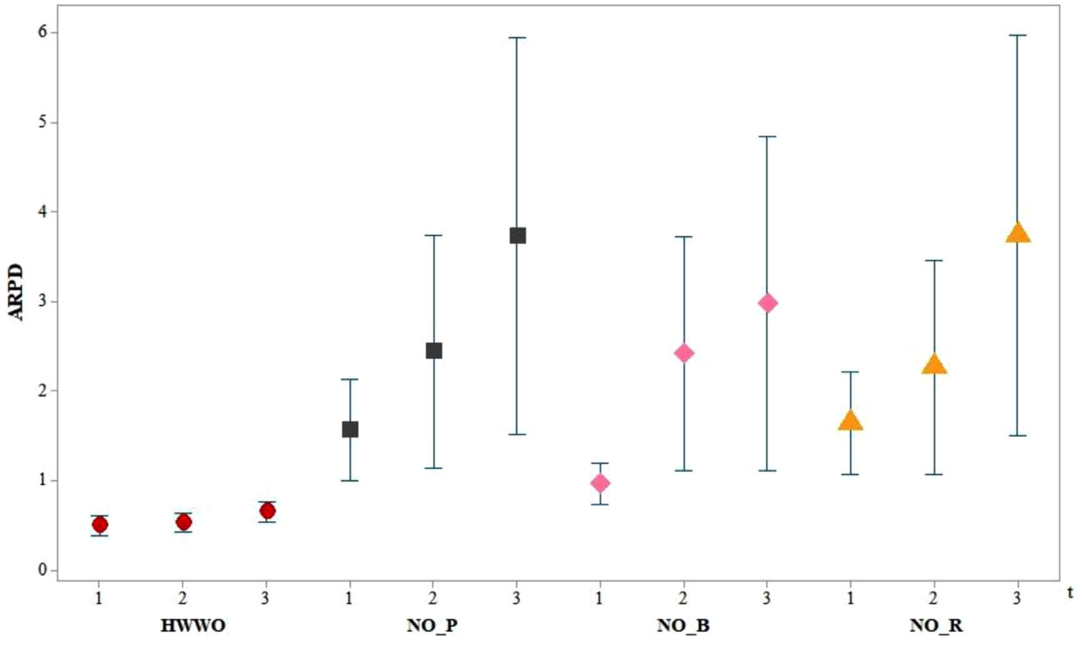


Fig. 7. Interval plot of interaction among HWWO, NO_P, NO_B, NO_R with different τ.

adaption selection neighborhood search strategy and the selection strategy is decided by wavelength r. In the breaking operation, VNS is the main operator and the time complexity of the breaking operation is $O(NP^*k^*mn^2)$. The last phase is the refraction operation, the main process is determined by the perturbation operation. The time complexity for the worst case of the refraction operation is $O(NP^*mn^2)$. Therefore, the time complexity of the HWWO with k iterations is calculated based on the above analysis as follows.

$$\begin{aligned} O(NP, m, n, k, r) &= O(k) + O\left(NP * mn^2\right) + O(NP * n + r * mn) \\ &\quad + O\left(NP * k * mn^2\right) + O\left(NP * mn^2\right) \\ &= O\left(k * NP * mn^2 + mn\right) = O\left(k * NP * mn^2\right) \end{aligned}$$

4.2. Initialization methods analysis

In order to evaluate the performance of the proposed initialize method, the other three initialization methods, *NEH*, $PR_{SKE}NEH$(PRNEH) and $PR_{SKE}RNEH$(MNEH2) as control methods are introduced to compare and all compared methods are evaluated using Taillard's benchmark set with the same termination criterion $t_{max} = n^*m^*10$ milliseconds and the average relative percentage deviation (ARPD), which is employed to measure the quality of solutions, is adopted to measure the performance of the experimental results. As seen from Table 3, the ARPD values of the four initialization methods with different due dates are calculated to eval-

Table 4
Comparison for NO_P, NO_B, NO_R and HWWO with different τ.

$n*m$	$\tau = 1$				$\tau = 2$				$\tau = 3$			
	HWWO	NO_P	NO_B	NO_R	HWWO	NO_P	NO_B	NO_R	HWWO	NO_P	NO_B	NO_R
20*5	**0.353**	2.710	1.534	2.189	**0.281**	5.474	4.396	4.682	**0.686**	8.983	3.252	7.922
20*10	**0.428**	3.119	1.458	3.361	**0.677**	6.317	6.404	6.162	**0.549**	9.770	8.640	9.767
20*20	**0.249**	2.417	0.807	2.214	**0.649**	5.087	5.873	4.549	**0.561**	8.484	9.036	9.699
50*5	**0.613**	1.577	0.784	1.729	**0.618**	1.657	1.917	1.747	**0.801**	1.968	1.403	2.370
50*10	**0.590**	2.002	1.544	2.492	**0.586**	2.702	2.543	2.357	**0.649**	3.676	3.134	3.732
50*20	**0.688**	2.148	0.957	2.478	**0.629**	2.381	2.483	2.461	**0.774**	5.111	3.957	4.892
100*5	**0.568**	0.858	0.993	0.928	**0.665**	0.975	1.295	0.801	**0.839**	1.004	0.956	1.128
100*10	**0.554**	0.972	0.679	0.967	**0.355**	1.084	0.884	1.206	**0.703**	1.467	1.135	1.658
100*20	**0.693**	1.246	0.998	1.433	**0.602**	1.456	1.229	1.169	**0.882**	2.259	1.778	1.489
200*10	**0.505**	0.540	0.632	0.655	**0.519**	0.610	0.751	0.571	**0.733**	0.756	0.757	0.740
200*20	**0.648**	0.814	0.792	0.717	**0.666**	0.938	0.811	1.010	**0.338**	0.826	1.172	0.887
500*20	**0.129**	0.432	0.399	0.548	**0.177**	0.598	0.406	0.445	**0.292**	0.564	0.515	0.526
Average	**0.501**	1.569	0.965	1.643	**0.535**	2.440	2.416	2.263	**0.650**	3.739	2.978	3.734

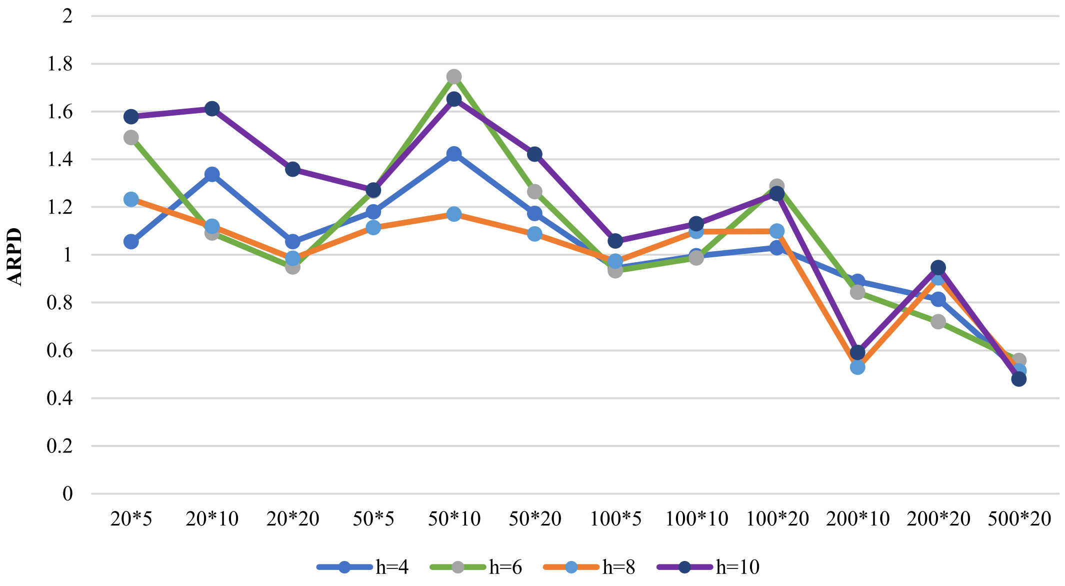


Fig. 8. Fitting graph between the size of job and machine and wave height.

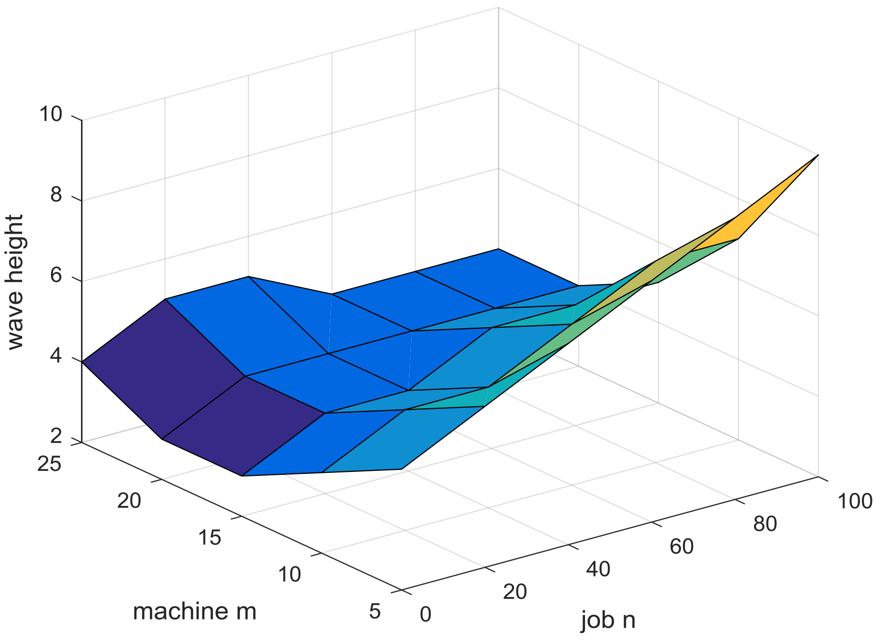


Fig. 9. Fitting graph between the size of number of jobs and machines and wave height.

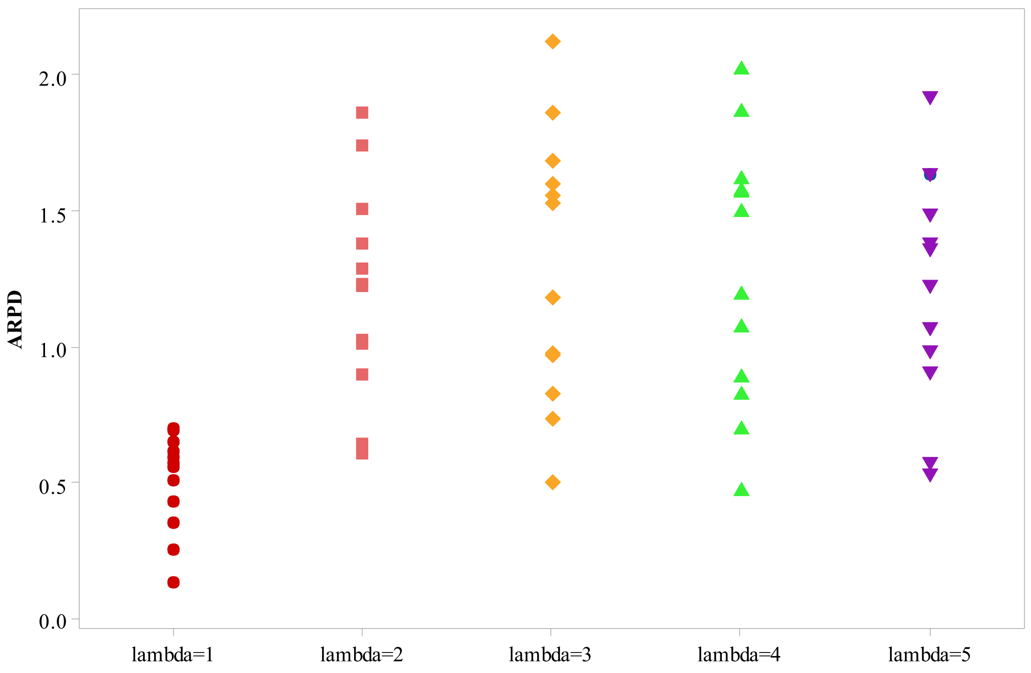


Fig. 10. Individual value plot of different wave lengths.

Table 5
A brief description for six compared algorithms.

Algorithm	Description
HDTLM	The size of the population $NP = 40$, the size of the elite population $\gamma = 0.2$, the global learning rate $\alpha = 0.7$, the local learning rate $\beta = 0.5$, the local searching rate $ls = 0.1$.
BEDA	The size of the population $NP = n$, the size of superior population is $0.2{*}NP$, the proportion of sub-population is set to $0.05{*}NP$, the ranges of the local learning rate and the global learning rate are $\{0.005{*}n, 5\}$.
vDE_IG	The size of the population is set to 30, the probability of crossover $Cr = 0.9$, the mutation scale factor $Fr = 0.9$, the destruction size $d = 4$.
DABC	The size of the population $NP = 100$, the population size of employed bees, onlooker bees, and scout bees are set as NP, $NP{*}2$, $NP{*}0.1$, respectively.
GVNS	The size of population is set to 100 and the destruction size $d = 4$.
vpsDE	the population size is set to 30, the crossover rate $Cr = 0.1$, the mutation scale factor $Fr = 0.1$.

Table 6
The results of all compared algorithms on Taillard's benchmark set with $\tau = 1$.

$n{*}m$	*HDTLM*	*vDE_IG*	*BEDA*	*DABC*	*GVNS*	*vpsDE*	*HWWO*
20*5	0.809	0.553	1.047	0.628	0.810	0.675	**0.353**
20*10	0.533	0.463	0.976	0.457	0.494	0.608	**0.428**
20*20	0.201	0.265	0.614	0.232	**0.180**	0.595	0.249
50*5	0.809	0.663	1.190	0.740	0.824	0.714	**0.613**
50*10	1.129	1.004	1.285	1.123	1.258	1.413	**0.590**
50*20	0.880	0.713	1.471	0.722	0.882	1.148	**0.688**
100*5	0.927	0.893	0.754	0.789	0.773	1.099	**0.568**
100*10	0.696	**0.409**	0.845	0.576	0.714	0.953	0.554
100*20	0.826	0.733	0.828	0.914	0.867	1.055	**0.693**
200*10	0.587	0.616	0.955	0.748	0.712	0.753	**0.505**
200*20	0.679	0.710	0.801	0.859	0.677	0.658	**0.648**
500*20	0.646	0.497	0.523	0.857	0.342	0.805	**0.129**
Average	0.727	0.627	0.941	0.720	0.711	0.873	**0.501**

Table 7
The results of all compared algorithms on Taillard's benchmark set with $\tau = 2$.

$n{*}m$	*HDTLM*	*vDE_IG*	*BEDA*	*DABC*	*GVNS*	*vpsDE*	*HWWO*
20*5	0.951	0.681	1.581	0.853	0.991	1.347	**0.281**
20*10	1.033	**0.390**	2.206	0.723	0.976	1.606	0.677
20*20	0.817	**0.414**	1.689	0.569	0.764	0.894	0.649
50*5	1.101	0.754	1.153	1.132	0.948	1.162	**0.618**
50*10	1.518	1.181	1.673	1.810	1.599	1.427	**0.586**
50*20	1.569	1.332	1.716	1.335	1.272	1.633	**0.629**
100*5	1.105	0.676	1.011	0.896	0.899	1.133	**0.665**
100*10	0.799	0.668	0.825	0.858	0.829	1.173	**0.355**
100*20	1.123	0.997	1.073	1.098	1.136	1.391	**0.602**
200*10	0.912	0.599	0.620	0.548	0.605	0.670	**0.519**
200*20	0.906	1.030	1.082	1.014	0.770	1.019	**0.666**
500*20	0.535	0.629	0.614	0.747	0.387	0.852	**0.177**
Average	1.031	0.779	1.270	0.965	0.931	1.192	**0.535**

uate the performance of all compared initialization methods. Obviously, the stability and quality of the population are improved by the proposed initialize method from Table 3. Fig. 6. shows the probability plot for $PR_{COV\_SKE}RNEH$ (MNEH1), $PR_{SKE}RNEH$(MNEH2), $PR_{SKE}NEH$(PRNEH), NEH with different due dates for a 95% confidence interval. From Fig. 6, compared with the other initialize methods, the distribution of the population for $PR_{COV\_SKE}RNEH$ is correspondingly stable and compact. However, the diversity of the population is likely to decrease. Note that, due dates are divided into three levels, tight due dates ($\tau = 1$), medium due dates ($\tau = 2$) and loose due dates ($\tau = 3$), respectively. Obviously, the proposed initialize method $PR_{COV\_SKE}RNEH$ (MNEH1) outperformed the other three initialize methods. The computational formula of ARPD is shown as follows.

$$ARPD = \frac{1}{R} * \sum_{i=1}^{R} \frac{TTD_i - TTD_{OPT}}{TTD_{OPT}} * 100\% \tag{20}$$

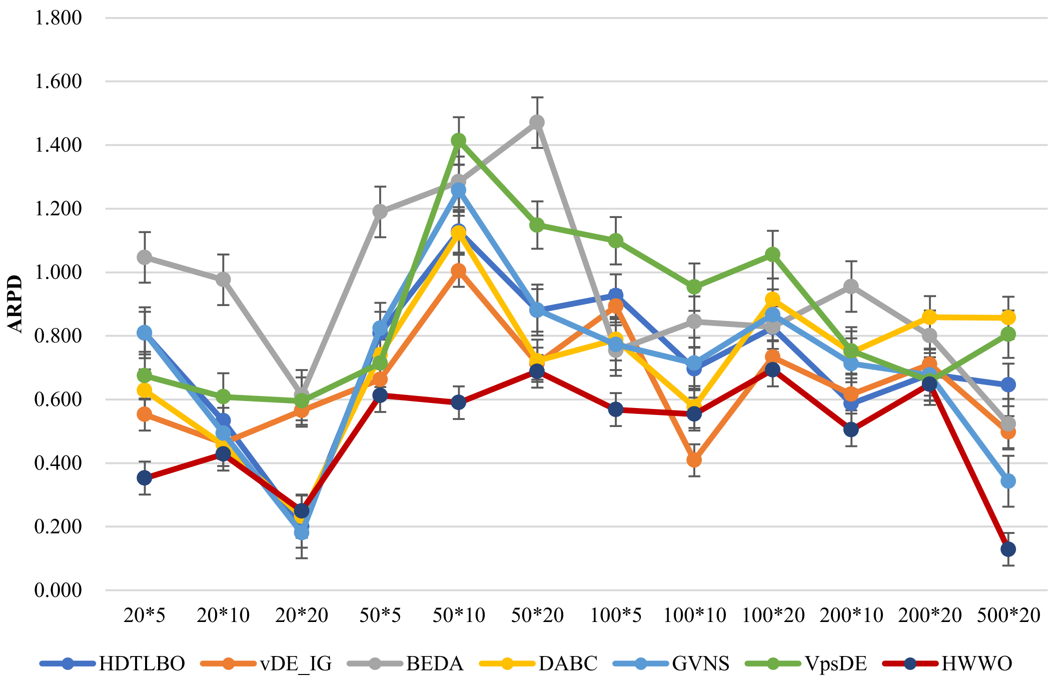


Fig. 11. Mean plot for all compared algorithms with $\tau = 1$.

Table 8
The results of all compared algorithms on Taillard's benchmark set with $\tau = 3$.

*n*m*	*HDTLM*	*vDE_IG*	*BEDA*	*DABC*	*GVNS*	*vpsDE*	*HWWO*
20*5	1.234	0.708	3.814	0.987	1.741	0.997	**0.686**
20*10	0.627	0.727	6.731	**0.351**	0.427	1.451	0.549
20*20	1.696	0.887	8.103	1.151	1.585	3.866	**0.561**
50*5	1.534	1.185	1.414	1.457	1.001	1.568	**0.801**
50*10	2.039	1.983	2.669	1.907	1.921	3.067	**0.649**
50*20	2.443	2.021	2.540	2.978	2.017	2.552	**0.774**
100*5	1.188	0.841	1.431	1.050	1.103	1.610	**0.839**
100*10	**0.664**	0.762	0.906	1.009	0.786	1.628	0.703
100*20	1.837	1.409	1.520	1.257	1.778	2.300	**0.882**
200*10	0.975	**0.645**	1.057	0.881	0.692	0.989	0.733
200*20	1.001	0.955	1.067	0.865	0.941	1.415	**0.338**
500*20	0.804	0.846	0.906	0.776	0.661	0.982	**0.292**
Average	1.337	1.081	2.680	1.222	1.221	1.869	**0.650**

where the TTD_i is the solution obtained by an algorithm tested for the *ith* run of a certain instance, the TTD_{OPT} means the best solution found so far, R is the number of run times for the algorithm on a single case.

4.3. Operational analysis

In this section, the contribution of the self-adaption propagation operation, local reinforcement based on the breaking operation and the perturbation of the refraction operation are analyzed. There are three algorithms, including HWWO without the propagation operation (NO_P), HWWO without the breaking operation (NO_B), HWWO without the refraction operation (NO_R). All compared algorithms are tested on Taillard's benchmark set with an identical termination criterion $t_{max} = n^*m^*10$ milliseconds. The computational results of NO_P, NO_B, NO_R and HWWO are shown in Table 4 and the best results are highlighted in boldface. As seen from Table 4, the ARPD of HWWO are better than the ARPD of NO_P, NO_B, NO_R. Fig. 7 represents the interval plot of interaction among HWWO, NO_P, NO_B, NO_R for different τ with a 95% confidence interval. Combined with Fig. 7, the interval plot of HWWO is under the other three algorithms, which illustrates that the HWWO is significantly superior to NO_P, NO_B, NO_R. Table 4 and Fig. 7. also demonstrate that NO_P and NO_R are the two worst algorithms. On the contrary, NO_B is the key component and plays a vital role. The results obtained by the NO_R are better than that of the results of the NO_P, which implies that the contribution of the perturbation of refraction is superior to the self-adaption selection propagation. All in all, a desirable balance obtained by the HWWO is obtained among the propagation operation based self-adaption selection, local intensification search for the breaking operation and the perturbation of the refraction operation.

4.4. Control parameters

A mass of experiments is implemented to find the optimal combination of the parameters, which is the fact that the performance of an algorithm is determined by parameters (Shao et al., 2019). Obviously, the performance of the algorithm is immensely improved by the methods of parameter adjustment and the interaction between the parameters achieves the predetermined effect. In the HWWO, there are three important parameters considered, the population size NP, the maximum wavelength λ_{max} and wave height h. In this paper, the different methods of the control parameter are adopted to detect the influence of the parameters on the algorithm and to provide a set of desirable parameter combinations. By the way, the population size is decided by the number of jobs. If $\frac{n}{3} > 50,\ NP = 50$, otherwise $NP = round(\frac{n}{3})$. According to the experimental results, it is found that the number of jobs and machines has an effect on the choice of wave height during the process of experiments. Inspired by the control of wave height according to Zhao et al. (2017), Taillard's benchmark set is adopted to detect the proper value of wave height and the wave height is strongly affected by the size of number of jobs and machines. Ob-

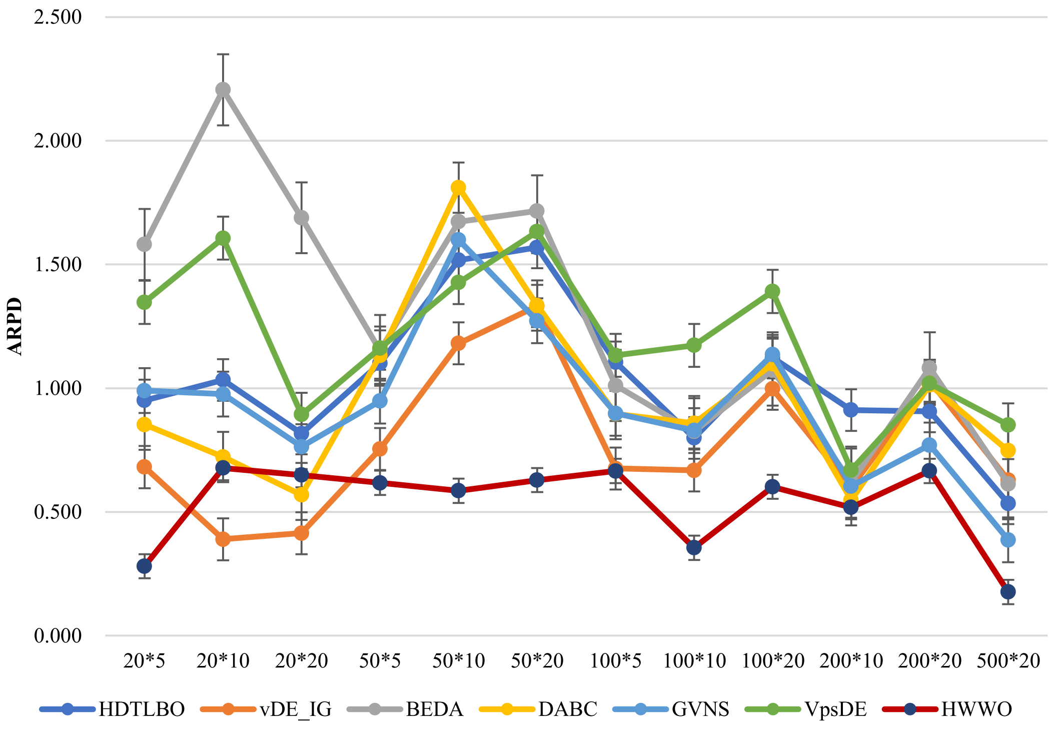


Fig. 12. Mean plot for all compared algorithms with $\tau = 2$.

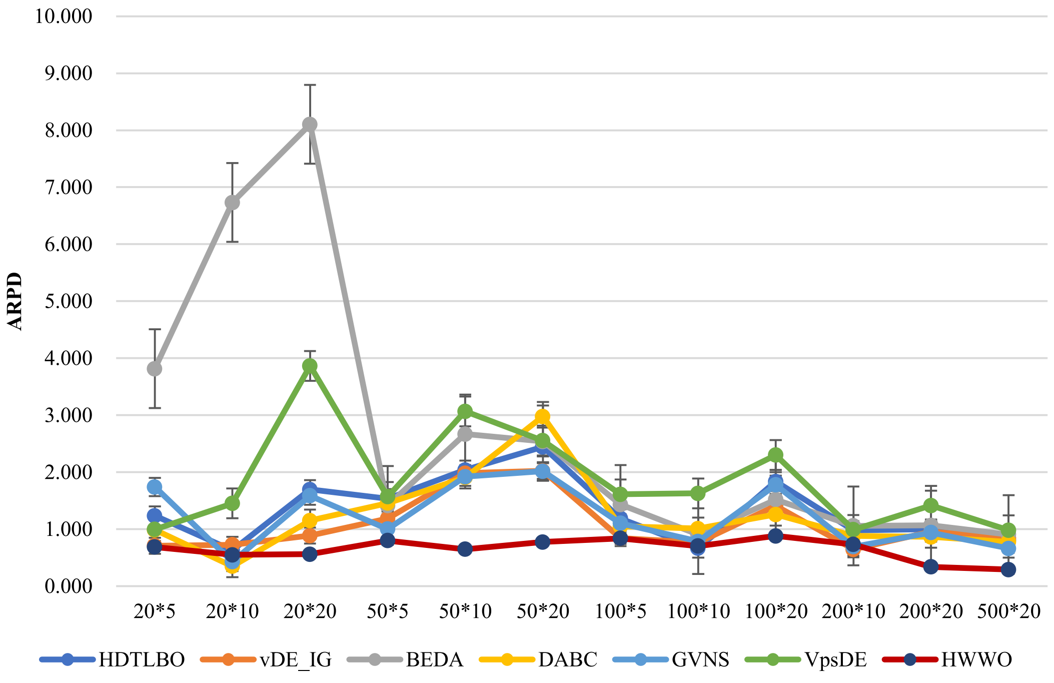


Fig. 13. Mean plot for all compared algorithms with $\tau = 3$.

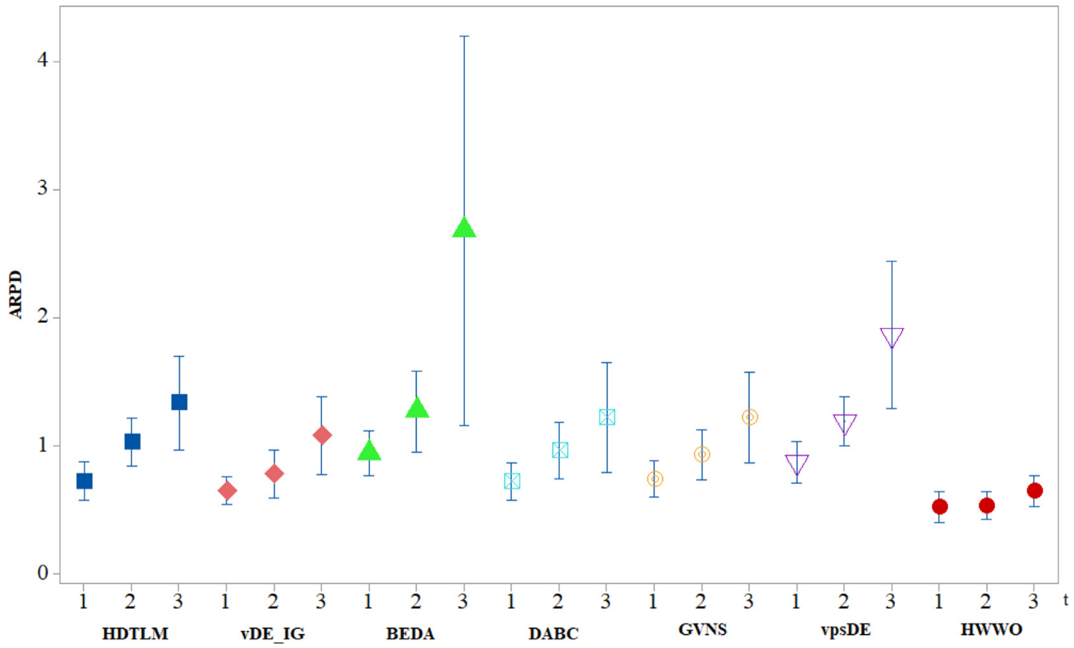


Fig. 14. Interval plot of interaction for all compared algorithms with different τ on Taillard's benchmark set.

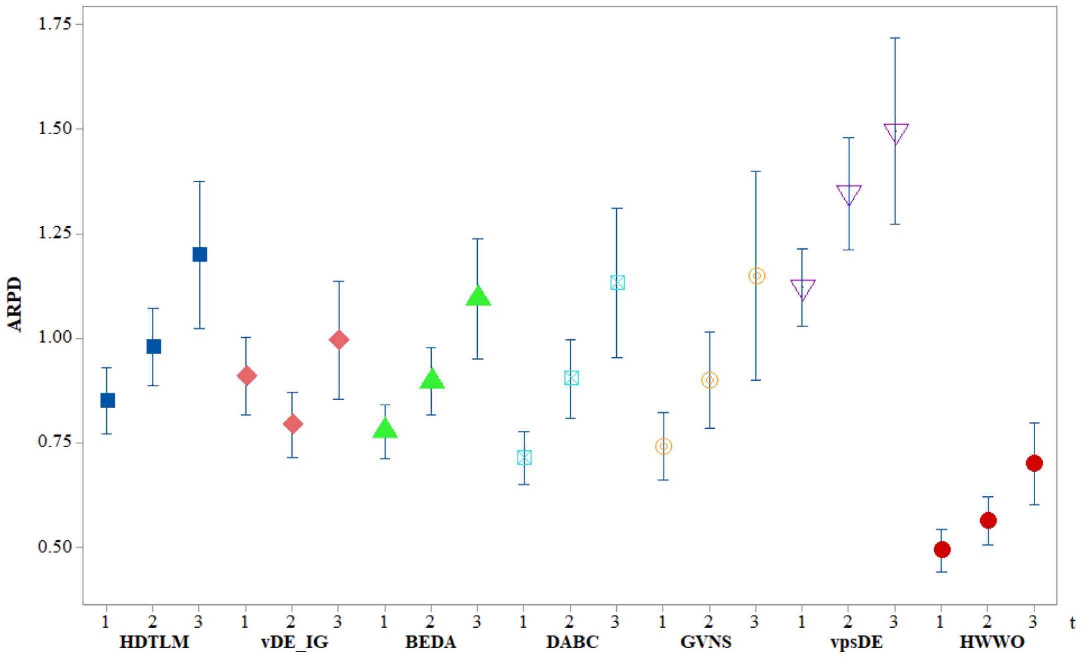


Fig. 15. Interval plot of interaction for all compared algorithms with different τ on Ruiz's benchmark set.

viously, the performance of the proposed algorithm is altered with different wave heights in the Fig. 8. Therefore, the wave height is not set to a constant owing to its strong sensitivity.

As seen from Fig. 9, wave height is controlled by the number of jobs and machines. Therefore, the computational formula for wave height is shown as follows.

$$h = (0.0680 * n - 0.5607 * m - 0.0025 * n * m - 0.0001 * n^2 + 0.0173 * m^2 + 7.6592) \tag{21}$$

For the choice of the parameter λ_{max}, the feasible levels of λ_{max} are listed as follows. $\lambda_{max} \in \{1, 2, 3, 4, 5\}$. Similar to the wave height, λ_{max} also has a significant effect on the performance of the HWWO and the minimum neighboring search scope is determined by the wave length λ_{max}. A large value of λ_{max} leads to the newly generated individuals being removed from the current promising region and a decrease of the quality of the offspring. Therefore, λ_{max} controls the scope of [1,5] and the analysis of the λ_{max} is presented as follows. In this section, an individual value plot is applied to evaluate and compare the distribution of the different maximum wave lengths. As seen from Fig. 10, a λ_{max} with the range of [1,5] are shown. $\lambda_{max} = 1$, indicates that the performance of the proposed algorithm is the best, while the performance of the proposed algorithm with $\lambda_{max} = 3$ is the worst. The reason is that a large value of λ_{max} leads to newly generated individuals

Table 9
The results of all compared algorithms on Ruiz's benchmark set with $\tau = 1$.

n	*m*	*HDTLM*	*vDE_IG*	*BEDA*	*DABC*	*GVNS*	*vpsDE*	*HWWO*
50	10	0.779	1.142	0.766	0.478	0.725	1.042	**0.469**
	20	1.275	1.594	1.043	0.709	1.079	1.241	**0.607**
	30	0.768	1.413	0.888	0.806	0.969	1.249	**0.492**
	40	1.410	0.992	0.978	1.069	1.498	1.597	**0.866**
	50	1.377	1.145	0.904	1.294	1.026	1.715	**0.804**
100	10	**0.430**	0.983	0.943	0.667	0.789	1.393	0.524
	20	0.629	1.225	0.666	0.625	1.017	0.921	**0.397**
	30	0.726	1.253	1.145	0.859	0.946	1.053	**0.510**
	40	1.383	1.468	0.991	1.245	1.091	1.554	**0.721**
	50	1.225	1.059	0.963	1.325	1.461	1.773	**0.950**
150	10	0.993	1.005	0.635	0.930	0.704	0.899	**0.470**
	20	0.791	1.191	0.506	0.922	0.717	0.954	**0.442**
	30	0.780	0.949	1.023	**0.663**	0.780	1.209	0.898
	40	1.047	1.069	1.045	0.763	0.702	1.304	**0.549**
	50	1.190	1.309	1.120	0.644	1.245	1.393	**0.527**
200	10	0.686	0.692	0.705	0.784	0.628	0.818	**0.439**
	20	1.037	0.908	0.819	1.005	0.932	1.346	**0.498**
	30	0.915	1.202	1.012	1.072	0.747	1.734	**0.602**
	40	1.218	1.700	1.051	0.894	0.871	1.392	**0.587**
	50	0.916	1.419	0.955	0.813	0.742	1.381	**0.477**
250	10	0.593	0.772	0.587	0.609	0.463	0.963	**0.363**
	20	0.809	1.272	0.699	0.707	0.814	1.040	**0.475**
	30	0.708	1.213	0.899	0.635	0.543	1.220	**0.468**
	40	1.164	1.168	1.224	0.708	1.146	1.593	**0.662**
	50	1.022	1.074	1.167	**0.709**	1.014	1.596	0.999
300	10	0.505	0.603	0.742	0.501	0.980	0.868	**0.375**
	20	0.553	0.884	0.699	0.770	0.716	0.631	**0.488**
	30	0.739	**0.414**	0.679	0.914	0.898	0.912	0.638
	40	0.935	0.768	0.937	0.470	0.573	1.228	**0.437**
	50	0.677	0.576	0.770	0.640	0.719	1.471	**0.325**
350	10	0.598	0.484	0.727	0.439	0.344	0.705	**0.257**
	20	0.528	0.571	0.687	0.635	0.420	0.788	**0.357**
	30	1.127	0.659	0.542	0.707	0.697	1.161	**0.417**
	40	1.085	0.737	0.914	0.796	0.904	1.176	**0.428**
	50	0.628	0.817	0.789	0.621	**0.558**	1.237	0.668
400	10	0.776	0.501	0.185	0.730	0.333	0.727	**0.169**
	20	0.673	0.877	0.541	0.496	0.495	0.647	**0.441**
	30	0.703	0.612	0.607	0.413	0.644	0.854	**0.373**
	40	0.506	0.769	0.516	0.520	0.606	1.183	**0.317**
	50	1.485	0.539	0.748	0.618	0.631	1.075	**0.440**
450	10	0.525	0.761	**0.360**	0.510	0.399	0.784	0.394
	20	0.688	0.722	0.798	0.590	0.422	0.743	**0.318**
	30	0.648	0.551	0.643	0.445	0.643	1.101	**0.371**
	40	1.011	0.749	0.714	0.582	0.541	0.653	**0.401**
	50	0.639	0.595	0.849	0.577	0.781	1.570	**0.344**
500	10	0.586	0.609	0.333	0.496	**0.299**	0.502	0.301
	20	0.458	0.505	0.597	0.473	0.248	0.692	**0.239**
	30	0.725	0.454	0.362	0.643	0.373	0.885	**0.328**
	40	0.754	0.727	0.667	0.390	0.423	1.054	**0.317**
	50	1.021	0.708	0.621	0.686	0.684	0.996	**0.603**
Average		0.849	0.908	0.775	0.712	0.740	1.120	**0.491**

being removed from the current promising region and decreases the quality of the offspring. However, the performance of $\lambda_{max} = 5$ and $\lambda_{max} = 2$ has no significant difference. The second worst is $\lambda_{max} = 4$ and the performance of the proposed algorithm outperforms that obtained under $\lambda_{max} = 3$. Therefore, the performance of HWWO is determined by $\lambda_{max} = 1$. To sum up, all parameters determined in the HWWO are demonstrated as the above analysis.

4.5. Comparative experimental analysis

In this section, the proposed algorithm and the other compared algorithms, including *HDTLM* (Shao, Pi & Shao, 2018), *BEDA* (Shen, Ling & Wang, 2015), *vDE_IG* (Tasgetiren et al., 2013), *DABC* (Tasgetiren, Pan, Suganthan & Oner, 2013), *GVNS* (Tasgetiren et al., 2013) and *vpsDE* (Tasgetiren et al., 2011), are tested on Taillard's benchmark set (Taillard, 1993) and Ruiz's benchmark set (Ruiz, Vallada & Fernándezmartínez, 2009) to verify the performance of the HWWO. In addition, a brief description of the compared algorithms is shown in Table 5. All compared algorithms use an identical termination criterion $t_{max} = n*m*10$ milliseconds and the average relative percentage deviation (ARPD) is adopted to measure the performance of the experimental results to illustrate the effectiveness of our proposed algorithm and compare its performance with the other approaches. Each algorithm is run 10 times and the speed-up method for the insertion neighborhood (Tasgetiren et al., 2013) is applied in all compared algorithms as far as possible.

The experimental results based on Taillard's benchmark set for the seven algorithms with different due dates are shown in Tables 6–8. The best solutions are shown in boldface. As seen from Table 6, $\tau = 1$, a desirable performance for almost all groups except for 20*20 and 100*10 is obtained by the HWWO and the average ARPD of the HWWO is 0.501 according to improve all strategies

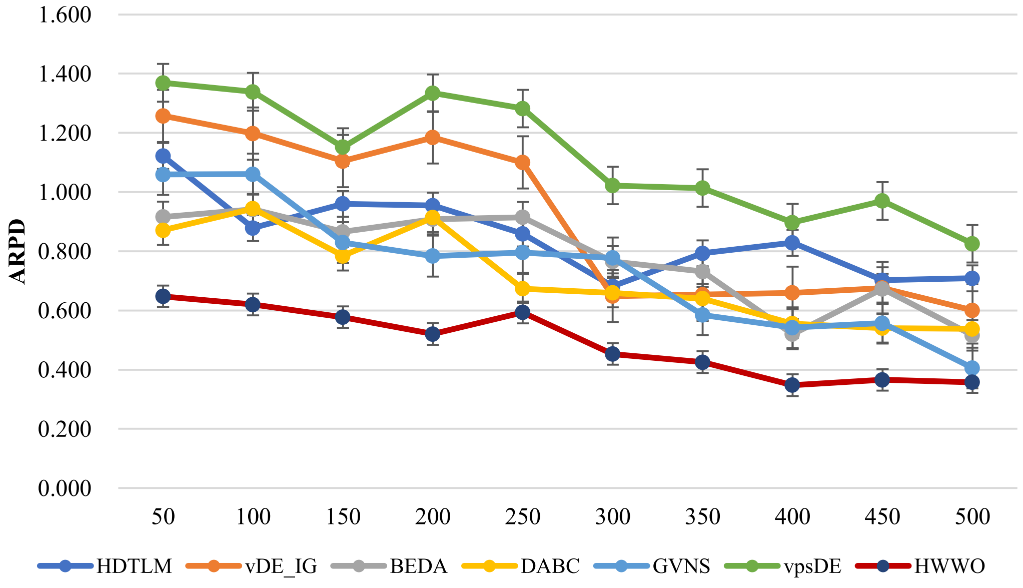


Fig. 16. Mean plot for all compared algorithms and *n*.

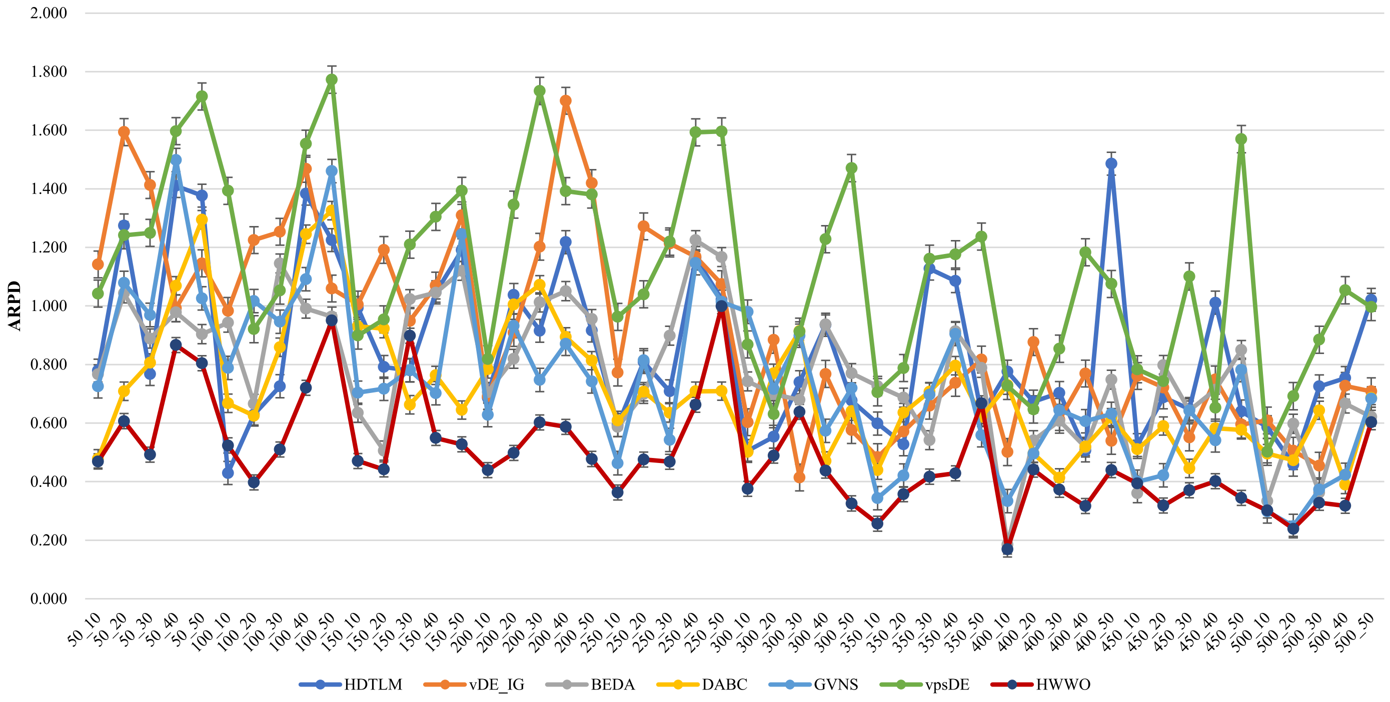


Fig. 17. Mean plot for all compared algorithms with $\tau = 1$ and *m*.

applied in the HWWO, which outperformed the corresponding values obtained by the other compared algorithms. For the medium due date $\tau = 2$ from Table 7, the results achieved by the HWWO are superior to other compared algorithms in almost all groups except for 20*10, 20*20 and 50*5. The ARPD acquired by the HWWO is superior than the other six compared algorithms, which have been certified to be promising algorithms to solve the NIFSP in the literature. Considering the loose due dates $\tau = 3$ shown in Table 8, the performance of the HWWO outperforms the other six compared algorithms in a majority of the groups but 20*10, 100*10 and 200*10. Furthermore, the mean plots for all compared algorithms with different due dates are shown from Figs. 11–13. It is obvious that the plots of the HWWO are better than the other plots for the six compared algorithms. Moreover, the interval plot of interaction among the algorithms for different due dates with 95% confidence intervals is shown in Fig. 14, it is demonstrated that the interval plot of the HWWO is better than the plots of the other compared algorithms. To sum up, the proposed algorithm presents a satisfactory result based on Taillard's benchmark set to solve the NIFSP.

Moreover, the experimental results based on Ruiz's benchmark set for the seven algorithms with different due dates are shown from Tables 9–11. The best solutions are highlighted in boldface.

Table 10
The results of all compared algorithms on Ruiz's benchmark set with $\tau = 2$.

n	*m*	*HDTLM*	*vDE_IG*	*BEDA*	*DABC*	*GVNS*	*vpsDE*	*HWWO*
50	10	1.234	0.959	1.448	1.126	1.712	1.840	**0.869**
	20	1.397	1.517	1.034	2.090	1.278	1.935	**0.817**
	30	2.116	0.744	1.266	1.514	1.545	2.536	**0.647**
	40	1.598	**0.840**	1.507	0.977	1.573	1.737	0.901
	50	1.633	1.179	1.267	1.822	1.946	2.156	**0.958**
100	10	0.888	0.972	1.167	0.831	1.139	1.447	**0.697**
	20	0.857	1.012	1.229	0.799	0.990	1.520	**0.765**
	30	1.559	1.209	1.019	1.064	1.281	2.061	**0.767**
	40	1.253	1.776	1.104	1.585	1.458	1.855	**0.895**
	50	1.292	**0.990**	1.045	1.121	1.560	2.140	1.037
150	10	0.708	1.155	0.689	0.617	0.788	0.977	**0.540**
	20	0.824	0.881	0.870	1.088	1.040	1.294	**0.511**
	30	0.751	0.655	1.124	0.886	0.804	1.040	**0.644**
	40	1.471	0.959	1.158	1.392	1.435	1.787	**0.703**
	50	1.147	0.864	**0.815**	1.064	1.250	1.825	0.920
200	10	0.773	0.510	0.867	0.866	0.605	1.012	**0.474**
	20	1.048	0.726	0.771	0.793	0.804	0.646	**0.499**
	30	0.900	0.774	1.023	0.732	0.735	1.627	**0.721**
	40	1.057	0.983	0.943	0.999	1.268	2.059	**0.787**
	50	1.289	0.888	1.037	0.633	1.257	1.973	**0.626**
250	10	0.744	0.399	0.877	0.767	0.668	0.769	**0.329**
	20	0.925	0.926	0.766	0.598	0.866	1.252	**0.547**
	30	1.034	0.857	1.258	0.639	0.732	1.309	**0.563**
	40	1.101	1.055	1.274	0.959	1.059	1.548	**0.870**
	50	0.919	1.051	1.449	0.825	1.517	1.262	**0.510**
300	10	0.849	0.650	0.609	0.556	0.476	0.725	**0.354**
	20	0.743	0.783	0.777	0.817	0.550	1.022	**0.466**
	30	0.860	0.795	0.883	0.832	0.590	0.889	**0.405**
	40	0.706	0.609	0.969	1.019	0.801	1.575	**0.540**
	50	1.211	0.607	0.946	1.232	0.917	1.901	**0.522**
350	10	0.863	0.300	0.295	0.930	**0.283**	0.697	0.519
	20	0.677	0.594	0.753	0.773	0.491	1.256	**0.423**
	30	0.847	0.729	0.549	0.576	0.505	1.106	**0.255**
	40	0.735	0.572	0.873	0.636	0.851	0.895	**0.332**
	50	1.190	0.389	1.124	0.741	0.944	1.582	**0.384**
400	10	0.584	0.556	0.430	0.931	0.352	0.612	**0.334**
	20	0.650	0.768	0.592	0.474	0.399	1.611	**0.357**
	30	0.923	0.660	0.424	0.596	0.902	1.160	**0.401**
	40	0.654	0.874	0.515	0.864	**0.469**	1.283	0.487
	50	0.797	0.627	0.848	0.618	1.060	1.558	**0.476**
450	10	0.528	0.413	0.498	0.650	0.536	0.768	**0.229**
	20	0.718	0.611	0.808	0.857	**0.447**	0.835	0.520
	30	1.254	0.555	0.582	0.699	0.585	1.082	**0.364**
	40	0.660	0.689	1.071	0.975	0.831	0.972	**0.387**
	50	1.241	0.719	0.665	0.854	0.951	1.056	**0.576**
500	10	0.623	0.697	0.731	**0.241**	0.464	0.851	0.617
	20	0.510	0.528	0.528	0.719	0.410	0.974	**0.298**
	30	0.782	0.537	0.716	0.779	0.354	0.861	**0.298**
	40	1.040	0.572	0.699	0.933	0.749	1.067	**0.301**
	50	0.729	0.840	0.882	1.018	0.670	1.301	**0.402**
Average		0.978	0.791	0.895	0.902	0.898	1.345	**0.521**

As seen from Tables 9–10, the results obtained by the HWWO are better than the other compared algorithms in a majority of groups and the ARPD achieved by the HWWO on average are 0.491, 0.521 and 0.688, respectively. Additionally, the interval plot of the interaction among algorithms for different due dates with 95% confidence intervals is shown in Fig. 15. As seen from Fig. 15, the interval plot of the HWWO is under the other plots and the results obtained by the HWWO are better than the other compared algorithms. Afterwards, the mean plots of all compared algorithms with n is shown in Fig. 16. and the mean plots of all compared algorithms with different τ and m are shown in Figs. 17–19. It's clearly seen that the mean plots of the HWWO are under the plots of the other compared algorithms and the values of ARPD for the HWWO again illustrated the performance of the HWWO. Furthermore, the detailed analysis is as follows. First, the number of jobs and machines have a significant impact on all the compared algorithms from Tables 6–11. Second, the values of ARPD obtained by the HWWO are increased when the number of machines is increased from $m = 30$ to $m = 50$ in most instances. Combined with Fig. 14, the values of ARPD are increased with increased due dates, namely, the larger the tightness factor, the higher the values of ARPD in most instances. Obviously, the desirable performance obtained by the HWWO is shown when the number of jobs is equal to 400 and 450. Finally, the values of ARPD for all the compared algorithms decreases as the number of jobs increased. In addition, *DABC* performed worst when the number of machines is increased from $m = 30$ to $m = 50$, which is different from the other compared algorithms. In conclusion, the proposed HWWO is an effective and novel algorithm to solve the NIFSP with total tardiness criterion.

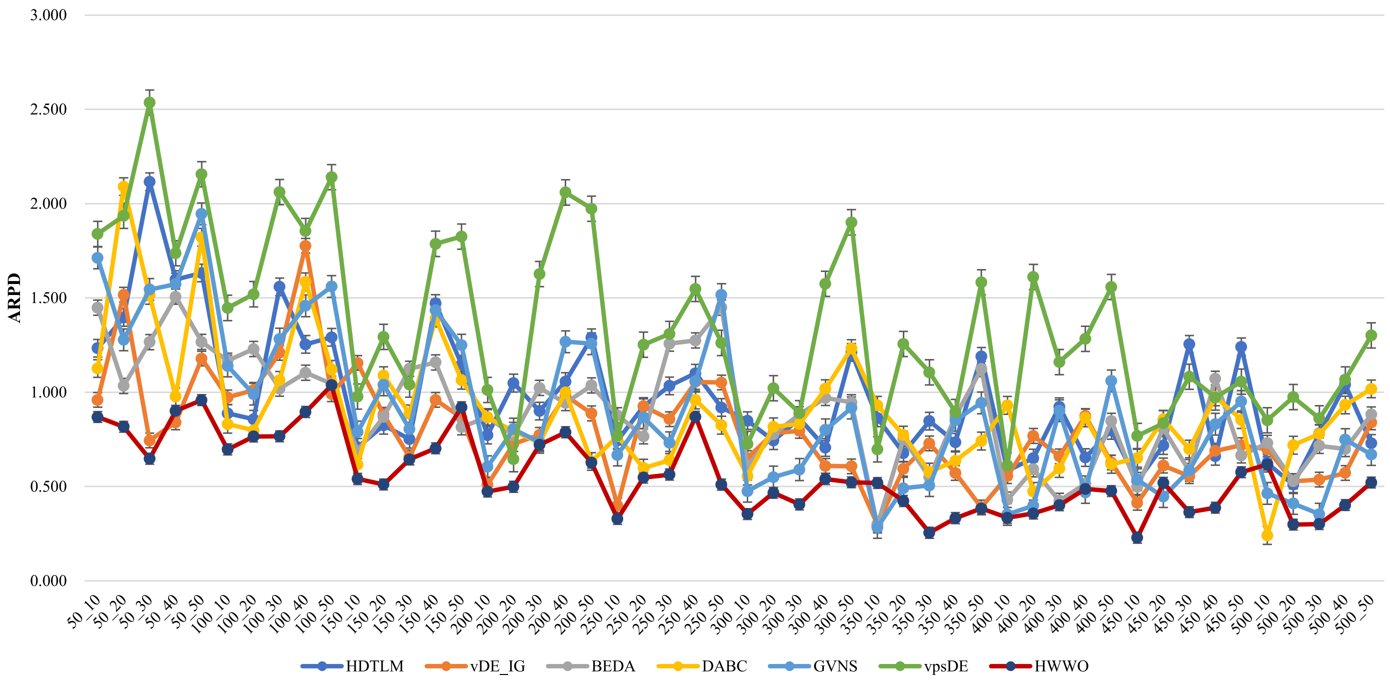


Fig. 18. Mean plot for all compared algorithms with $\tau = 2$ and m.

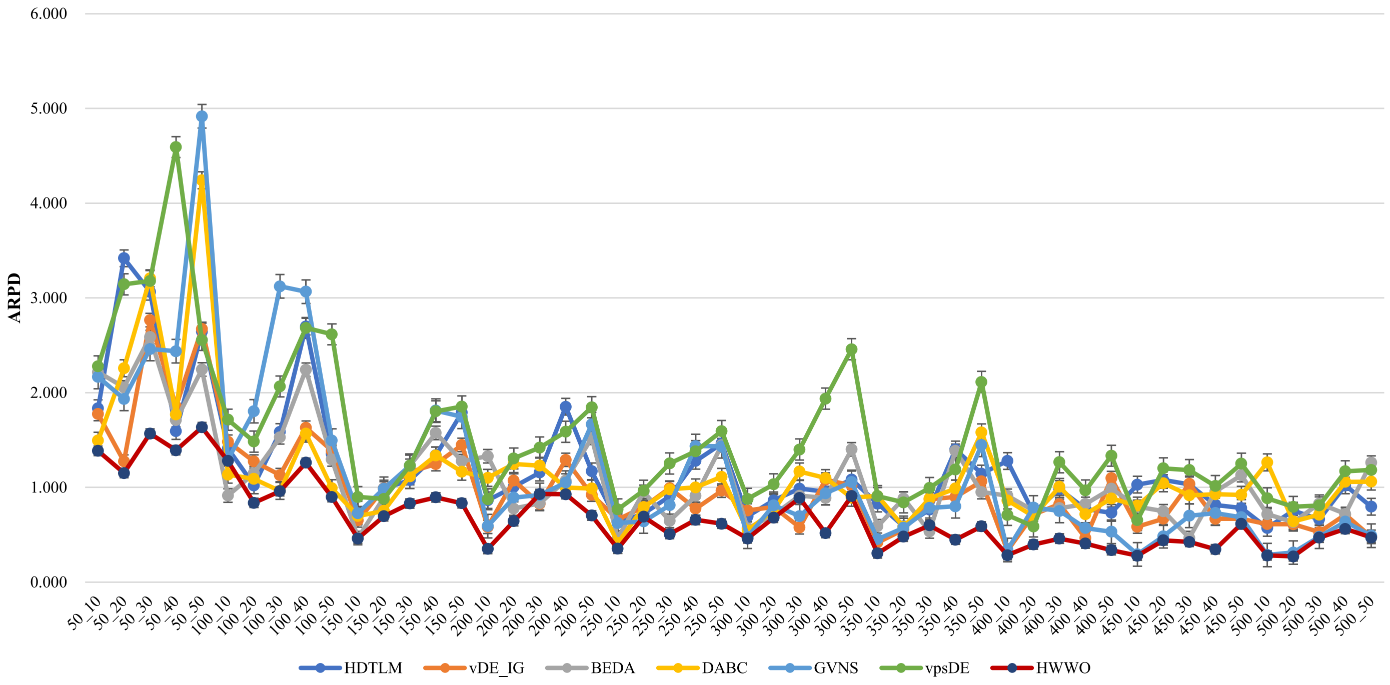


Fig. 19. Mean plot for all compared algorithms with $\tau = 3$ and m.

5. Conclusion

In this paper, a hybrid discrete water wave optimization algorithm (HWWO) is proposed to solve the no-idle flowshop scheduling problem (NIFSP) with total tardiness criterion. First, a new priority rule based on the modified NEH method for initialization is designed to generate solutions. The quality and diversity of solutions are improved by the combination of the coefficient of variation and skewness. Afterwards. A self-adaption selection controlled by wavelength for propagation, which has eleven selection strategies, is applied to generate a new wave and enhance the exploration of the HWWO. In the breaking phase, VNS is provided to improve the exploitation of the HWWO and maintain the diversity of solutions. Furthermore, a perturbation sequence combined with the historical best solution obtained by the breaking operation to generate a new solution is developed to avoid the proposed algorithm falling into a local optimal in the refraction operation. Moreover, the effectiveness of each component is appraised to demonstrate the contribution of the proposed algorithm. In addition, the control parameters of the HWWO are calibrated and the time complexity of the HWWO is analyzed. The proposed algorithm is evaluated on Taillard's benchmark set and Ruiz's benchmark set, and

Table 11
The results of all compared algorithms on Ruiz's benchmark set with $\tau = 3$.

n	*m*	*HDTLM*	*vDE_IG*	*BEDA*	*DABC*	*GVNS*	*vpsDE*	*HWWO*
50	10	1.836	1.775	2.217	1.494	2.166	2.278	**1.386**
	20	3.419	1.274	2.055	2.257	1.934	3.144	**1.151**
	30	3.064	2.766	2.588	3.207	2.461	3.178	**1.569**
	40	1.594	1.856	1.710	1.770	2.438	4.592	**1.393**
	50	2.645	2.673	2.245	4.242	4.917	2.557	**1.635**
100	10	1.411	1.483	**0.913**	1.135	1.295	1.715	1.279
	20	1.021	1.274	1.149	1.090	1.804	1.484	**0.837**
	30	1.586	1.130	1.528	0.963	3.122	2.065	**0.960**
	40	2.698	1.631	2.243	1.568	3.067	2.683	**1.261**
	50	1.352	1.386	1.295	0.992	1.495	2.616	**0.898**
150	10	0.712	0.651	0.466	0.698	0.728	0.899	**0.460**
	20	0.989	0.989	0.891	0.753	0.986	0.878	**0.696**
	30	1.076	1.162	1.228	1.117	1.229	1.230	**0.830**
	40	1.339	1.246	1.575	1.337	1.811	1.803	**0.895**
	50	1.792	1.448	1.277	1.165	1.744	1.854	**0.833**
200	10	0.865	0.583	1.330	1.101	0.592	0.875	**0.351**
	20	1.005	1.073	0.777	1.247	0.893	1.306	**0.644**
	30	1.157	**0.826**	0.833	1.229	0.920	1.421	0.930
	40	1.851	1.291	1.073	0.998	1.050	1.587	**0.928**
	50	1.170	0.923	1.523	0.988	1.667	1.846	**0.704**
250	10	0.631	0.662	0.464	0.415	0.624	0.768	**0.353**
	20	0.724	0.793	0.952	0.791	**0.641**	0.966	0.691
	30	0.904	0.992	0.648	0.980	0.813	1.253	**0.508**
	40	1.280	0.776	0.911	1.000	1.436	1.382	**0.657**
	50	1.463	0.965	1.447	1.111	1.433	1.595	**0.617**
300	10	0.695	0.757	0.568	0.547	0.478	0.879	**0.461**
	20	0.863	0.793	0.709	0.755	0.811	1.033	**0.681**
	30	0.987	**0.578**	0.912	1.168	0.696	1.400	0.888
	40	0.955	1.082	0.889	1.099	0.939	1.937	**0.518**
	50	1.084	1.021	1.402	**0.891**	1.057	2.459	0.908
350	10	0.829	0.418	0.590	0.910	0.455	0.908	**0.304**
	20	0.589	0.556	0.884	0.585	0.574	0.842	**0.481**
	30	0.766	0.874	**0.535**	0.883	0.783	0.994	0.599
	40	1.401	0.900	1.387	0.989	0.800	1.188	**0.448**
	50	1.146	1.064	0.951	1.580	1.450	2.113	**0.590**
400	10	1.279	0.331	0.913	0.863	0.340	0.709	**0.284**
	20	0.731	0.713	0.754	0.696	0.788	0.588	**0.397**
	30	0.982	0.824	0.781	1.006	0.751	1.266	**0.460**
	40	0.772	0.464	0.825	0.718	0.571	0.968	**0.407**
	50	0.731	1.098	0.987	0.883	0.532	1.335	**0.337**
450	10	1.029	0.586	0.795	0.807	0.293	0.654	**0.280**
	20	1.078	0.669	0.749	1.044	0.484	1.202	**0.441**
	30	1.032	1.039	0.466	0.917	0.700	1.183	**0.425**
	40	0.815	0.669	0.957	0.927	0.729	1.015	**0.347**
	50	0.783	0.669	1.128	0.921	0.685	1.251	**0.615**
500	10	0.572	0.613	0.717	1.264	0.286	0.886	**0.282**
	20	0.755	0.611	0.633	0.641	0.312	0.795	**0.271**
	30	0.664	0.529	0.829	0.713	0.479	0.809	**0.470**
	40	1.021	0.712	0.720	1.059	0.639	1.170	**0.562**
	50	0.796	0.478	1.262	1.062	0.491	1.185	**0.471**
Average		1.199	0.994	1.094	1.131	1.148	1.495	**0.688**

compared with six state-of-the-art algorithms. The comparative results demonstrate that the proposed algorithm is significantly superior to the other compared algorithms.

For the progress of future work, there are several issues, which are worthy of study. First, a more efficient initialization method and constructive heuristic algorithm are expanded to generate a promising population. Second, in order to avoid the algorithm being trapped into the local optimal, various efficient local optimization strategies are introduced in the refraction operation. Finally, the HWWO algorithm is applied to solve the other flowshop scheduling problems, for instance, the distributed flowshop scheduling problem, the hybrid flowshop scheduling problem or the multi-objective flowshop scheduling problem. Moreover, several numerous realistic constraints, such as energy consumption and machine maintenance, are considered.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Credit authorship contribution statement

Fuqing Zhao: Funding acquisition, Investigation, Supervision, Writing - review & editing. **Lixin Zhang:** Investigation, Software, Writing - original draft. **Yi Zhang:** Conceptualization, Formal analysis. **Weimin Ma:** Methodology, Resources. **Chuck Zhang:** Project administration, Writing - review & editing. **Houbin Song:** Visualization.

Acknowledgements

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 TechnologyCommission Foundation of Gansu Province (2017GS10817), Lanzhou Science Bureau project (2018-rc-98), Public Welfare Project of Zhejiang Natural Science Foundation (LGJ19E050001), Wenzhou Public Welfare Science and Technologyproject (G20170016), respectively.

References

Adiri, I., & Pohoryles, D. (1982). Flowshop/no-idle or no-wait scheduling to minimize the sum of completion times. *Naval Research Logistics Quarterly, 29*, 495–504.
Allahverdi, A., Aydilek, H., & Aydilek, A. (2018). No-wait flowshop scheduling problem with two criteria; total tardiness and makespan. *European Journal of Operational Research, 269*, 590–601.
Baptiste, P., & Hguny, L. K. (1997). A branch and bound algorithm for the F/no-idle/Cmax. In *Proceedings of the International Conference on Industrial Engineering and Production Management: 1* (pp. 429–438).
Billaut, J. C., Croce, F. D., Salassa, F., & T'Kindt, V. (2019). No-idle, no-wait: When shop scheduling meets dominoes, Eulerian paths and hamiltonian paths. Journal of Scheduling, *22*, 59–68.
Chakrabortty, R. K., Abbasi, A., & Ryan, M. J. (2020). Multi-mode resource–constrained project scheduling using modified variable neighborhood search heuristic. *International Transactions in Operational Research, 27*, 138–167.
Cheng, C. Y., Ying, K. C., Chen, H. H., & Lu, H. S. (2018). Minimising makespan in distributed mixed no-idle flowshops. *International Journal of Production Research*, 1–13.
Ding, J. Y., Song, S., & Cheng, W. (2015). Carbon-efficient scheduling of flow shops by multi-objective optimization. *European Journal of Operational Research, 248*, 758–771.
Goncharov, Y. (2009). The flow shop problem with no-idle constraints: A review and approximation. European Journal of Operational Research, *196*, 450–456.
Hansen, P., & Mladenović, N. (2005). Variable neighborhood search. In E. K. Burke, & G. Kendall (Eds.), Search methodologies: Introductory tutorials in optimization and decision support techniques (pp. 211–238). Boston, MA: Springer US.
Hematabadi, A. A., & Foroud, A. A. (2018). Optimizing the multi-objective bidding strategy using min–max technique and modified water wave optimization method. *Neural Computing & Applications*, 1–19.
Intellektik, F., Darmstadt, T. H., & Stutzle, T. (1998). Applying iterated local search to the permutation flow shop problem. *Handbook of Metaheuristics*.
Jing-Fang, C., Ling, W., & Jing-Jing, W. (2018). A two-stage memetic algorithm for distributed no-idle permutation flowshop scheduling problem.
Kalczynski, P. J., & Kamburowski, J. (2005). A heuristic for minimizing the makespan in no-idle permutation flow shops. *Computers & Industrial Engineering, 49*, 146–154.
Kirlik, G., & Oguz, C. (2012). A variable neighborhood search for minimizing total weighted tardiness with sequence dependent setup times on a single machine. *Computers & Operations Research, 39*, 1506–1520.
Liu, W., Yan, J., & Price, M. (2017). A new improved NEH heuristic for permutation flowshop scheduling problems. *International Journal of Production Economics, 193*, 21–30.
Nagano, M. S., Rossi, F. L., & Martarelli, N. J. (2019). High-performing heuristics to minimize flowtime in no-idle permutation flowshop. *Engineering Optimization, 51*, 185–198.
Narain, L., & Bagga, P. C. (2003). Minimizing total elapsed time subject to zero total idle time of machines in n×3 flowshop problem. *Indian Journal of Pure & Applied Mathematics*, 219–228.
Nawaz, M., Enscore, E., & Ham, I. (1983). A heuristic algorithm for the m-Machine, n-Job flow-shop sequencing problem. Omega, *11*, 91–95.

Pan, Q. K., & Wang, L. (2008). No-idle permutation flow shop scheduling based on a hybrid discrete particle swarm optimization algorithm. *International Journal of Advanced Manufacturing Technology, 39*, 796–807.
Pinedo, M., & Hadavi, K. (2016). Scheduling: Theory, algorithms and systems development. AIIE Transactions, *28*, 695–697.
Rao, R. V. (2016). Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. International Journal of Industrial Engineering Computations, *7*, 19–34.
Ras, M. N., Wilke, D. N., Groenwold, A. A., & Kok, S. (2019). On the rotational variance of the differential evolution algorithm. *Advances in Engineering Software, 136*, 19.
Ronconi, D. P., & Henriques, L. R. S. (2009). Some heuristic algorithms for total tardiness minimization in a flowshop with blocking. Omega, *37*, 272–281.
Ruiz, R., & Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research, 165*, 479–494.
Ruiz, R., Maroto, C., & Alcaraz, J. (2009a). Two new robust genetic algorithms for the flowshop scheduling problem. Omega, *34*, 461–476.
Ruiz, R., Vallada, E., & Fernándezmartínez, C. (2009b). Scheduling in flowshops with no-idle machines. In *Computational intelligence in flow shop and job shop scheduling: 230* (pp. 21–51). BerlinHeidelberg: Springer.
Ruizab, R. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research, 177*, 2033–2049.
Saadani, N. E. H., Guinet, A., & Moalla, M. (2003). Three stage no-idle flow-shops. *Computers & Industrial Engineering, 44*, 425–434.
Shao, W., Pi, D., & Shao, Z. (2018). A hybrid discrete teaching-learning based meta-heuristic for solving no-idle flow shop scheduling problem with total tardiness criterion. *Computers & Operations Research, 94*, 89–105.
Shao, Z. S., Pi, D. C., & Shao, W. S. (2019). A novel multi-objective discrete water wave optimization for solving multi-objective blocking flow-shop scheduling problem. *Knowledge-Based Systems, 165*, 110–131.
Shao, Z., Pi, D., & Shao, W. (2017). A novel discrete water wave optimization algorithm for blocking flow-shop scheduling problem with sequence-dependent setup times. *Swarm & Evolutionary Computation, 40*, 53–75.
Shen, J. N., Ling, W., & Wang, S. Y. (2015). A bi-population EDA for solving the no-idle permutation flow-shop scheduling problem with the total tardiness criterion. *Knowledge-Based Systems, 74*, 167–175.
Storn, R., & Price, K. (1997). Differential evolution; a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*(4), 341–359.
Taillard (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research, 64*, 278–285.
Tasgetiren, M. F., Buyukdagli, O., Pan, Q. K., & Suganthan, P. N. (2013a). A general variable neighborhood search algorithm for the no-idle permutation flowshop scheduling problem. *International Conference on Swarm, 8297*, 24–34.
Tasgetiren, M. F., Pan, Q. K., Suganthan, P. N., & Buyukdagli, O. (2013b). A variable iterated greedy algorithm with differential evolution for the no-idle permutation flowshop scheduling problem. *Computers & Operations Research, 40*, 1729–1743.
Tasgetiren, M. F., Pan, Q. K., Suganthan, P. N., & Chua, T. J. (2011). A differential evolution algorithm for the no-idle flowshop scheduling problem with total tardiness criterion. *International Journal of Production Research, 49*, 5033–5050.
Tasgetiren, M. F., Pan, Q. K., Suganthan, P. N., & Oner, A. (2013c). A discrete artificial bee colony algorithm for the no-idle permutation flowshop scheduling problem with the total tardiness criterion. *Applied Mathematical Modelling, 37*, 6758–6779.
Tasgetiren, M., Eliiyi, U., Öztop, H., Kizilay, D., & Pan, Q.-.K. (2018). An energy-efficient single machine scheduling with release dates and sequence-dependent setup times.
Ying, K.-C., Lin, S.-W., Cheng, C.-Y., & He, C.-D. (2017). Iterated reference greedy algorithm for solving distributed no-idle permutation flowshop scheduling problems. *Computers & Industrial Engineering, 110*, 413–423.
Zhao, F. Q., Qin, S., Zhang, Y., Ma, W. M., Zhang, C., & Song, H. B. (2019). A hybrid biogeography-based optimization with variable neighborhood search mechanism for no-wait flow shop scheduling problem. *Expert Systems with Applications, 126*, 321–339.
Zhao, F. Q., Xue, F. L., Zhang, Y., Ma, W. M., Zhang, C., & Song, H. B. (2019). A discrete gravitational search algorithm for the blocking flow shop problem with total flow time minimization. *Applied Intelligence, 49*, 3362–3382.
Zhao, F., Liu, H., Yi, Z., Ma, W., & Zhang, C. (2017). A discrete water wave optimization algorithm for no-wait flow shop scheduling problem. *Expert Systems with Applications, 91*, 347–363.
Zhao, F., Qin, S., Zhang, Y., Ma, W., & Song, H. (2018). A two-stage differential biogeography-based optimization algorithm and its performance analysis. *Expert Systems with Applications, 115*, 329–345.
Zhao, F., Xue, F., Zhang, Y., Ma, W., Zhang, C., & Song, H. (2018). A hybrid algorithm based on self-adaptive gravitational search algorithm and differential evolution. *Expert Systems with Applications, 113*, 515–530.
Zhao, F., Zhang, L., Liu, H., Zhang, Y., Ma, W., & Zhang, C. (2019). An improved water wave optimization algorithm with the single wave mechanism for the no-wait flow-shop scheduling problem. *Engineering Optimization, 51*, 1727–1742.
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).