An improved water wave optimization algorithm with the single wave mechanism for the no-wait flow-shop scheduling problem

Fuqing Zhao, Lixin Zhang, Huan Liu, Yi Zhang, Weimin Ma, Chuck Zhang & Houbin Song

To cite this article: Fuqing Zhao, Lixin Zhang, Huan Liu, Yi Zhang, Weimin Ma, Chuck Zhang & Houbin Song (2018): An improved water wave optimization algorithm with the single wave mechanism for the no-wait flow-shop scheduling problem, Engineering Optimization, DOI: 10.1080/0305215X.2018.1542693

To link to this article: https://doi.org/10.1080/0305215X.2018.1542693

View supplementary material

Published online: 26 Nov 2018.

Submit your article to this journal

Article views: 21

View Crossmark data

ENGINEERING OPTIMIZATION
https://doi.org/10.1080/0305215X.2018.1542693

An improved water wave optimization algorithm with the single wave mechanism for the no-wait flow-shop scheduling problem

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

[a]School of Computer and Communication Technology, Lanzhou University of Technology, Lanzhou, People's Republic of China; [b]School of Mechanical Engineering, Xijin University, Xi'an, People's Republic of China; [c]School of Economics and Management, Tongji University, Shanghai, People's Republic of China; [d]H. Milton Stewart School of Industrial & Systems Engineering, Georgia Institute of Technology, Atlanta, GA, USA

ABSTRACT
In this article, a water wave optimization algorithm with a single wave mechanism, called single water wave optimization (SWWO), is proposed to solve the no-wait flow-shop scheduling problem (NWFSP) with the objective of minimizing the makespan. In the proposed SWWO, an improved Nawaz–Enscore–Ham (NEH) heuristic is applied to construct a high-quality initial candidate. In the propagation operation, a self-adaptive block-shift operation is employed. In the breaking operation, a variable neighbourhood search operation is utilized to explore the local optimal solution. According to the schema theory as presented in genetic algorithms, a crossover operation is adopted as the refraction operation. Finally, the computational results based on several benchmarks and statistical performance comparisons are presented. The experimental results demonstrate the effectiveness and efficiency of the proposed SWWO for solving the NWFSP.

ARTICLE HISTORY
Received 10 July 2018
Accepted 28 October 2018

KEYWORDS
Water wave optimization (WWO); block-shift operation; variable neighbourhood search (VNS); no-wait flow-shop scheduling problem; makespan

1. Introduction

The flow-shop scheduling problem (FSP) has been a focus of extensive study in the operations research literature over the past several decades. As an important branch of the FSP, the no-wait flow-shop scheduling problem (NWFSP) is widely used in modern manufacturing and production systems, *e.g.* in the chemical industry (Rajendran 1994), food processing (Hall and Sriskandarajah 1996), the pharmaceutical industry (Raaymakers and Hoogeveen 2000) and steel rolling (Aldowaisan and Allahverdi 2012), and in some advanced manufacturing systems, including robotic cells (Agnetis 2000), flexible manufacturing systems (Wang, Xing, and Bai 2005), just-in-time production systems (Shabtay 2012), surgery scheduling (Wang *et al.* 2015), minimizing makespan in a two-stage flow shop (Huang *et al.* 2017), and two-criteria total tardiness and makespan (Allahverdi, Aydilek, and Aydilek 2018). For technological reasons, in the manufacturing industries no in-process waiting is allowed between any two consecutive machines, such that once the processing of a job starts, subsequent processing is continuously carried out on all machines with no interruptions until the processes have been completed. It has been verified that the NWFSP with more than two machines is strongly NP hard, *i.e.* the NWFSP becomes more complicated and difficult to solve with increasing problem size.

CONTACT Fuqing Zhao Fzhao2000@hotmail.com

Supplemental data for this article can be accessed at https://doi.org/10.1080/0305215X.2018.1542693

Since the NWFSP has a wide range of engineering applications, scholars in the field have conducted in-depth studies on NWFSPs and proposed various scheduling methods. These methods are divided into two categories: exact methods and heuristic algorithms. Exact solutions are obtained by the exact methods in the optimization process; however, the computational complexity and storage of the algorithms increases with increasing problem size. Heuristics are divided into two categories: constructive heuristics and metaheuristics. Constructive heuristics consume less time to obtain the best solutions, but the solution quality of the specific problems usually does not satisfy the requirements of the problem. Metaheuristics achieve satisfactory solutions in a reasonable computational time, and include the self-adaptive harmony search particle swarm optimization algorithm (Zhao *et al.* 2015), a hybrid algorithm based on the self-adaptive gravitational search algorithm–differential evolution (Zhao *et al.* 2018b) and a two-stage differential biogeography-based optimization algorithm (Zhao *et al.* 2019). In addition, metaheuristics have been utilized for solving the FSP, including the genetic algorithm (França, Tin Jr, and Buriol 2006), particle swarm optimization (Pan, Tasgetiren, and Liang 2008), iterative local search algorithm (Wang, Li, and Wang 2008), differential evolution (Qian *et al.* 2009), tabu search (Wang, Li, and Qian 2010; Ding *et al.* 2015a), discrete harmony search algorithm (Gao, Pan, and Li 2011), hybrid water flow algorithm (Tran and Ng 2013), hunting search metaheuristic algorithm (Naderi, Khalili, and Arshadi Khamseh 2014), chaotic local search bacterial foraging optimization (Zhao *et al.* 2016), hybrid harmony search algorithm (Zhao *et al.* 2017), teaching–learning-based optimization (Shao, Pi, and Shao 2018a), estimation of distribution algorithm with path relinking for the blocking FSP (Shao, Pi, and Shao 2018a) and discrete water wave optimization (DWWO) (Zhao *et al.* 2018a). Although various methods have been proposed and applied to solve the NWFSP, practitioners and researchers are still pursuing efficient methods.

Inspired by the shallow water wave models, the water wave optimization (WWO) algorithm was proposed by Zheng (2015). Three kinds of water wave evolution mechanism, namely propagation, refraction and breaking, are simulated according to the models. WWO and its variations have been applied to solve various problems, such as the travelling salesman problem (Wu, Liao, and Wang 2015), economic dispatch problem (Balamurugan, Siva, and Lakshminarasimman 2017), permutation flow-shop scheduling problem (Yun *et al.* 2016) and blocking flow-shop scheduling problem (Shao, Pi, and Shao 2018b). WWO tends to converge at a late stage with excessive same-candidate solutions in the population from the framework of WWO. In the framework of WWO, the water wave is preferentially propagated in a desirable direction by dynamically adjusting the wavelength, the wave height of each water wave and the quality of the water wave. Therefore, the effect of population on the algorithm is neglected in the design of the algorithm, and a WWO algorithm based on a single water wave algorithm, called single water wave optimization (SWWO), is proposed to solve the NWFSP. In the proposed algorithm, an improved Nawaz–Enscore–Ham (NEH) mechanism is introduced to construct a high-quality initial solution. As the propagation operation, a block-shift operation is employed to dynamically adjust the neighbourhood space. For the breaking operation, a variable neighbourhood search (VNS) operation is introduced to explore the local optimal solution. A crossover operation is applied as the refraction operation, allowing the current sequence to learn from the historically optimal sequence.

As a strongly NP-hard problem, under the condition of satisfying $m \geq 3$ (where m is the number of machines), it is difficult for the NWFSP to develop an exact algorithm to find an optimal solution within a reasonable time. The direct motivation for SWWO is to continually explore efficient algorithms to solve the NWFSP in certain manufacturing environments. Therefore, a new metaheuristic algorithm is proposed to solve the NWFSP and provide high-quality solutions within a reasonable time. The contributions of this article are summarized as follows.

- An NEH algorithm based on the coefficient of variation (NEH_COV) is proposed to improve the quality of the initial solution.

- According to the adaptability of WWO, the size of the block is dynamically adjusted. A modified block-shift operation (MBSO), which is an efficient insertion strategy, is introduced into the framework of WWO.
- VNS is introduced as the breaking operation into the framework of the SWWO. The modified variable neighbourhood search (MVNS) is proposed in this article to ensure that the optimal solution is found. However, the breaking operation is executed when the current optimal solution is searched.

The remainder of the article is organized as follows. Section 2 provides the problem formulation on NWFSP with the objective of minimizing the makespan. In Section 3, the SWWO is proposed, with four operators described in detail. In Section 4, several experiments are presented to verify the performance of the proposed algorithm. Section 5 summarizes conclusions and future work.

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

The NWFSP is described as follows. n jobs $\{J_1, J_2, \ldots, J_\mathrm{n}\}$ are processed on m machines $\{M_1, M_2, \ldots, M_\mathrm{m}\}$, and the processing routes of all jobs on m machines are the same.

- At a certain moment, a job is only processed on one machine.
- A machine processes one job at a time.
- Each job is processed without any waiting time between consecutive operations.

In this article, the objective is to find an optimal job permutation, denoted as $C_{\max}$, which has the minimum makespan. A sequence to be scheduled is expressed as $\pi = [\pi(1), \pi(2), \ldots, \pi(k), \ldots, \pi(n)]$. 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 the completion time distance between adjacent jobs ($\pi(i-1)$ and $\pi(i)$). Because of the no-wait constraint, a completion time distance between each pair of jobs is fixed in a certain problem instance. The $D(\pi(i-1), \pi(i))$ is calculated by Equation (1):

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

The makespan of sequence π is obtained as Equation (2):

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

To obtain a potential expression and simplify the calculation process, a virtual job $\pi(0)$ is introduced at the beginning of permutation π. The processing time of $\pi(0)$ is set to zero on all machines. Therefore, sequence π is replaced by $\pi' = [\pi(0), \pi(1), \ldots, \pi(k), \ldots \pi(n)]$, and set $D(\pi(0),\pi(1)) = \sum_{k=1}^{m} p(\pi(1),k)$. The computational formula of the 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 aimed for is calculated by Equation (4):

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

3. The SWWO algorithm

3.1. Initial population

The NEH algorithm is an effective constructive heuristic algorithm for solving the NWFSP. In the discrete harmony search algorithm (Gao, Pan, and Li 2011), the initial sequence is generated in descending order of standard deviation (SD) of the processing times in all machines. The method of initializing a sequence has been shown to produce a superior solution. The SD and the coefficient of variation (COV) are indicators measuring the dispersion of a data set. When the degree of variation of two or more data sets is compared, if the dimensions of each data set are the same, the SD is directly applied to compare them. If the data sets of different groups have different dimensions, the COV is used for comparison. In a variety of production scheduling environments, the average time per job processed on all machines is not guaranteed to have the same value. Therefore, using the COV is more accurate than using the SD to measure the dispersion of n data sets.

In combination with the original NEH strategy, the total processing time and the COV are considered to obtain the initial sequence. The square of the COV and the total processing time are used as measures to increase the weight of influence of the COV. The formulae used to calculate the indicator are listed in Equations (5)–(8). The pseudo-code of the NEH_COV algorithm is given in Algorithm 1. Table 1 shows the results of the new initial sequence approach. A modified Nawaz–Enscore–Ham

Algorithm 1 Pseudo-code of the NEH_COV algorithm

```
1 Generate an initial sequence by sorting the jobs according to the non-increasing order of CVT_i given in Equation (8). Denote
  the sequence as $\pi = [\pi(1) \quad \pi(2) \quad \cdots \quad \pi(n)]$.
2 Select the first two jobs in $\pi$ and evaluate the two possible sub-sequences. The better sub-sequence is chosen as the current
  sequence.
3 For $k = 3$ to $n$ do
4     Take the job $\pi_k$ and find the best sub-sequence by placing it in all possible positions of the sub-sequence that have been
      already scheduled.
5 End For
Output: the sequence $\pi$
```

Table 1. Comparison of the Nawaz–Enscore–Ham (NEH) algorithm, modified NEH (MNEH) and NEH based on coefficient of variation (NEH_COV).

	ARPD (%)			Average improvement ratio (%)	Average improvement ratio (%)
$n \times m$	NEH	MNEH	NEH_COV	NEH_COV vs NEH	NEH_COV vs MNEH
20 × 5	0.303	0.421	0.331	−0.028	0.09
20 × 10	0.558	0.99	0.493	0.065	0.497
20 × 20	0.329	0.872	0.536	−0.207	0.336
50 × 5	1.916	0.753	0.741	1.175	0.012
50 × 10	0.989	0.579	0.188	0.801	0.391
50 × 20	1.55	0.591	0.439	1.111	0.152
100 × 5	0.661	0.232	0.264	0.397	−0.032
100 × 10	0.876	0.531	0.146	0.73	0.385
100 × 20	0.124	0.695	0.277	−0.153	0.418
200 × 10	0.523	0.213	0.159	0.364	0.054
200 × 20	0.532	0.129	0.248	0.284	−0.119
500 × 20	0.188	0.11	0.128	0.06	−0.018
Average	NA	NA	NA	4.599	2.166

Note: ARPD = average relative percentage deviation; NA = not applicable.

(MNEH) algorithm was presented in Ding *et al.* (2015a). NEH_COV is a modified initial sequence method. The performance of the algorithm is improved by 2.166% on average.

$$\mathrm{AVG}_i = \frac{1}{m}\sum\nolimits_{j=1}^{m} p_{ij} \tag{5}$$

$$\mathrm{STD}_i = \sqrt{\frac{1}{m-1}\sum_{j=1}^{m}(p_{ij}-\mathrm{AVG}_i)^2} \tag{6}$$

$$\mathrm{CV}_i = \frac{\mathrm{STD}_i}{\mathrm{AVG}_i} \tag{7}$$

$$\mathrm{CVT}_i = \mathrm{CV}_i^{\,2} \times \sum\nolimits_{j=1}^{m} p_{ij} \tag{8}$$

3.2. Propagation operator

At each generation, a water wave is propagated once. The propagation range of a water wave is determined by the current wavelength (λ, $1 < \lambda < \lambda_{\max}$). Based on the block-shift operation (Ding *et al.* 2015b), the MBSO is employed in SWWO as the propagation operation. The adaptive tuning strategy for block size is named the k-job insert. In the NWFSP, the no-wait constraint motivates researchers to develop speed-up techniques for evaluating candidate solutions for basic neighbourhood operations (swap, insert). The k-job insert as an extension of the basic insert operation is also applied to speed-up techniques. Figure 1(c) illustrates the k-job insert. Equation (11) describes the application of the speed-up method to calculate the increment. The extension achieves high-efficiency candidate solutions. However, the importance of the k-job insert strategy has been neglected in the field. The k-job insert is utilized in Ding *et al.* (2015b) as the main body of the algorithm. In the proposed algorithm, the parameter k is set as the parameter wavelength λ in the WWO, so that the λ-job-insert operation is perfectly integrated into the framework of SWWO. Let the current wavelength of the water wave be λ. The pseudo-code of the propagation operator is given in Algorithm 2.

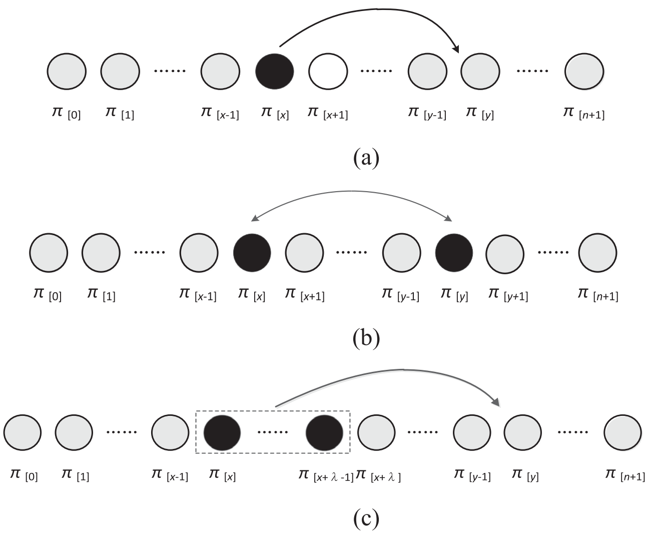


Figure 1. Three types of job move: (a) basic insert move; (b) basic swap move; (c) λ-insertion move.

Algorithm 2 Pseudo-code of the propagation operator

Input: current water wave: sequence π; wavelength of $\pi: \lambda$

```
1   $\Delta C^*_{\max} = +\infty$
2   For $i = 1:n-\lambda+1$
3     For $j = 1:n+1$
4       IF $(i > j \| i < j - r)$
5         Calculate the makespan increment $\Delta C_{\max}$ of inserting job block
          $[\pi(i), \pi(i+1), \ldots, \pi(i+\lambda-1)]$ into position $j$.
6         IF$(\Delta C_{\max} < \Delta C^*_{\max})$
7           $\Delta C^*_{\max} = \Delta C_{\max}$.
8           $x = i$
9           $y = j$
10        END IF
11      END IF
12    END FOR
13  END FOR
Insert the job block at position $x$ into position $y$, get the sequence $\pi'$
```

Output: sequence π'

Basic insert move:

$$\Delta_{\pi(x,y)} = D_{[y-1][x]} + D_{[x][y]} - D_{[y-1][y]} + D_{[x-1][x+1]} - D_{[x-1][x]} - D_{[x][x+1]} \tag{9}$$

Basic swap move:

$$\Delta_{\pi(x,y)} = \begin{cases} D_{[y-1][x]} + D_{[x][y+1]} + D_{[x-1][y]} + D_{[y][x+1]} - D_{[y-1][y]} \\ \qquad -D_{[y][y+1]} - D_{[x-1][x]} - D_{[x][x+1]}, & \text{if}\, x < y \\ D_{[y-1][x]} + D_{[x][y]} + D_{[y][x+1]} - D_{[y-1][y]} - D_{[y][x]} - D_{[x][x+1]}, & \text{if}\, x = y+1 \end{cases} \tag{10}$$

λ-Insertion move:

$$\Delta_{\pi(x,y)} = D_{[y-1][x]} + D_{[x+\lambda\ -\ 1][y]} - D_{[y-1][y]} + D_{[x-1][x+\lambda]} - D_{[x-1][x]} - D_{[x+\lambda\ -\ 1][x+\lambda]} \tag{11}$$

3.3. Refraction operator

There is only one water wave in the SWWO; a new solution is obtained by crossing the historical optimal solution with the current solution. This operation is called the refraction operation. The crossover operation is depicted in Figure 2. The pseudo-code of the refraction operator is given in Algorithm 3.

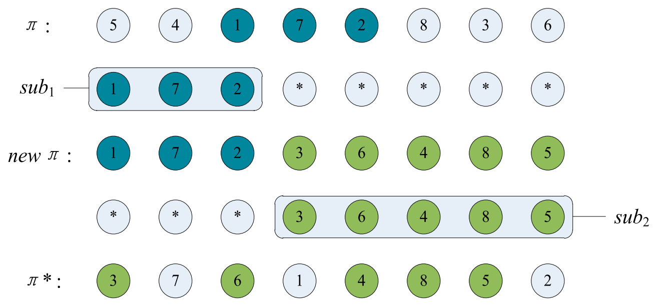


Figure 2. Crossover operation.

Algorithm 3 Pseudo-code of the refraction operator

```
Input: current water wave: sequence $\pi = [\pi(1) \quad \pi(2) \quad \cdots \quad \pi(n)]$; the best wave $\pi^*$
1  Generate two positions $P_1$ and $P_2$ randomly, $1 \le P_1 \le P_2 \le n$.
2  $sub_1 = [\pi(P_1) \quad \pi(P_1+1) \quad \cdots \quad \pi(P_2)]$.
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 $\pi$
```

3.4. Breaking operator

In SWWO, a breaking operation is performed when the historical optimal solution is found by the propagation operation. SWWO performs a deep local search near the historical best solution by running a VNS to simulate the breaking operation.

VNS is a metaheuristic method proposed by Hansen and Mladenovic (2001) to exploit the systematic change in the neighbourhood space to expand the search. A general VNS is presented by Kirlik and Oguz (2012) to solve minimize the total weighted tardiness in the presence of a sequence-dependent set-up. In this article, the terminating mechanism of the VNS is modified as an MVNS. In the MVNS, the termination condition is set to the $k_{\max}$ search without finding a desirable solution. Since the propagation operation is based on an extension of a basic insert operation, the swap operation is executed before the insert operation when the breaking operation is performed. Therefore, set N_1 as the swap neighbourhood structure and N_2 as the insert neighbourhood structure. Figure 1(a) and (b) illustrate the basic insertion operation and the basic swap operation, respectively. The pseudo-code of the breaking operator is outlined in Algorithm 4.

3.5. The framework of SWWO

SWWO is an efficient self-adaptation framework, which changes the search range of the algorithm by constantly adjusting the wavelength λ and the wave height h. In the initialization phase of the

Algorithm 4 Pseudo-code of the breaking operator

```
Input: current water wave: sequence $\pi$
1  Set the swap neighbourhood structure as $N_1$, the insert neighbourhood structure as $N_2$, set $l = 1$ and $t = 1$.
2  While $t <= 2$
3      IF $l == 1$
4          Find the best neighbour solution $\pi'$ of $\pi$ in $N_l(\pi)$.
5          IF $C_{\max}(\pi') < C_{\max}(\pi)$ then
6              Set $\pi = \pi'$.
7          ELSE
8              Set $t = t + 1$.
9          END
10         Set $l = 2$.
11     End If
12     IF $l == 2$
13         Find the best neighbour solution $\pi'$ of $\pi$ in $N_l(\pi)$.
14         IF $C_{\max}(\pi') < C_{\max}(\pi)$ then
15             Set $\pi = \pi'$.
16         ELSE
17             Set $t = t + 1$.
18         END
19             Set $l = 1$.
20     End If
21 End While
Output: sequence $\pi$
```

algorithm, λ and h are set to the maximum values, $\lambda_{\max}$ and $h_{\max}$, respectively. The inferior solution is accepted with a probability of

$$p = \exp\left(\frac{C_{\max}(\pi) - C_{\max}(\pi')}{T}\right) \tag{12}$$

where T is a constant temperature. Following the suggestions of Osman and Potts (1989), the temperature is set as follows:

$$T = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}}{10 \times n \times m} \times T_0 \tag{13}$$

where T_0 is a parameter to be adjusted. The maximum computational time is the termination condition adopted. The pseudo-code of the complete SWWO is given in Algorithm 5. A graphical description is shown in Figure 3.

Algorithm 5 Pseudo-code of SWWO

```
1  Initial parameters: temperature $T$, maximum wavelength $\lambda_{max}$, $=$ maximum wave height $h_{max}$.
2  Generate an initial sequence using the NEH_COV algorithm.
3  Evaluate the initial sequence $\pi$ and assign $\pi$ to the best sequence $\pi^*$.
4  While the termination condition is not met do
5    Propagate $\pi$ to a new $\pi'$.
6    If $C_{\max}(\pi') < C_{\max}(\pi)$ then
7      If $C_{\max}(\pi') < C_{\max}(\pi^*)$ then
8          Break $\pi'$ and replace it with the result.
9          $\pi^* = \pi'$.
10       End If
11       $\pi = \pi'$.
12       If $\lambda > 2$ then
13     $\lambda = \lambda - 1$;
14       Else
15       $\lambda = 2$
16       End If
17       Else
18         If $h == 0$ then
19           Refract $\pi'$ and replace it with the result.
20           Update $h$ and $\lambda$ to $h_{max}$ and $\lambda_{max}$, respectively.
21           Else
22           $h = h - 1$.
23           Set $\pi = \pi'$ with the probability in Equation (12).
24           If $\lambda < \lambda_{\max}$ then
25             $\lambda = \lambda + 1$;
26           Else
27             $\lambda = \lambda_{\max}$
28           End If
29           End If
30       End If
31   End While
32   Return $\pi^*$
```

4. Numerical experiments

The SWWO for the NWFSP was coded using MATLAB®. The simulation experiments were carried out on a personal computer with an Intel® Core™ i7-6700 central processing unit (CPU) at 3.4 GHz and 8 GB of memory, with a Windows Server 2012 Operating System. Owing to limited space, the parametric analysis of SWWO, analysis of neighbourhood structures, computational complexity for SWWO and evaluation criteria are described in the online Supplementary material. The

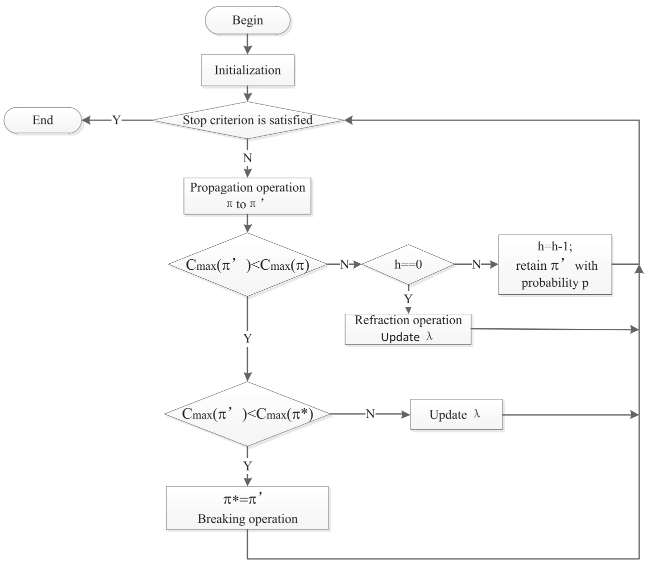

Begin
Initialization
Stop criterion is satisfied
End
Y
N
Propagation operation π to π'
Cmax(π')<Cmax(π)
N
h==0
N
h=h-1; retain π' with probability p
Y
Refraction operation Update λ
Y
Cmax(π')<Cmax(π*)
N
Update λ
Y
π*=π'
Breaking operation

Figure 3. Framework of the single water wave optimization (SWWO) algorithm.

performance of the SWWO compared with other algorithms is tested on three benchmark sets, consisting of three sets of instances: small-scale, Carlier's instances (Carlier 1978); medium-scale, Reeve's instances (Reeves 1995); and large-scale, Taillard's instances (Taillard 1993).

4.1. Performance analysis of each component of SWWO

In this section, the contribution of each component of the proposed algorithm is investigated and analysed. Four main components contribute to the proposed SWWO: the initial population, propagation operation, refraction operation and breaking operation. The four main components of SWWO are implemented to verify the key contributions. These components are included in SWWO without initial population (SWWO Noi), SWWO without propagation operation (SWWO Nop), SWWO without refraction operation (SWWO Nor) and SWWO without breaking operation (SWWO Nob). Each algorithm is run 10 times on each instance. The computational results and the best solutions are shown in Table 2. To enable a fair comparison, the termination criteria of all algorithms are uniformly identified as the same maximum CPU time.

Table 2. Computational results of all components of single water wave optimization (SWWO).

Instance	SWWO Nop	SWWO Nor	SWWO Nob	SWWO Noi
20 × 5	4.054	0.000	0.000	0.000
20 × 10	3.546	0.000	0.000	0.001
20 × 20	3.286	0.000	0.000	0.000
50 × 5	6.448	0.404	0.143	0.172
50 × 10	5.014	0.066	0.096	0.089
50 × 20	4.363	0.004	0.047	0.076
100 × 5	7.545	0.418	0.261	0.267
100 × 10	5.841	0.484	0.296	0.285
100 × 20	5.032	0.585	0.264	0.244
500 × 5	6.534	0.275	0.360	0.440
500 × 10	5.583	0.400	0.328	0.430
500 × 20	5.708	0.185	0.688	6.012
Average	5.246	0.235	0.207	0.668

Note: Nop = without propagation operation; Nor = without refraction operation; Nob = without breaking operation; Noi = without initial population.

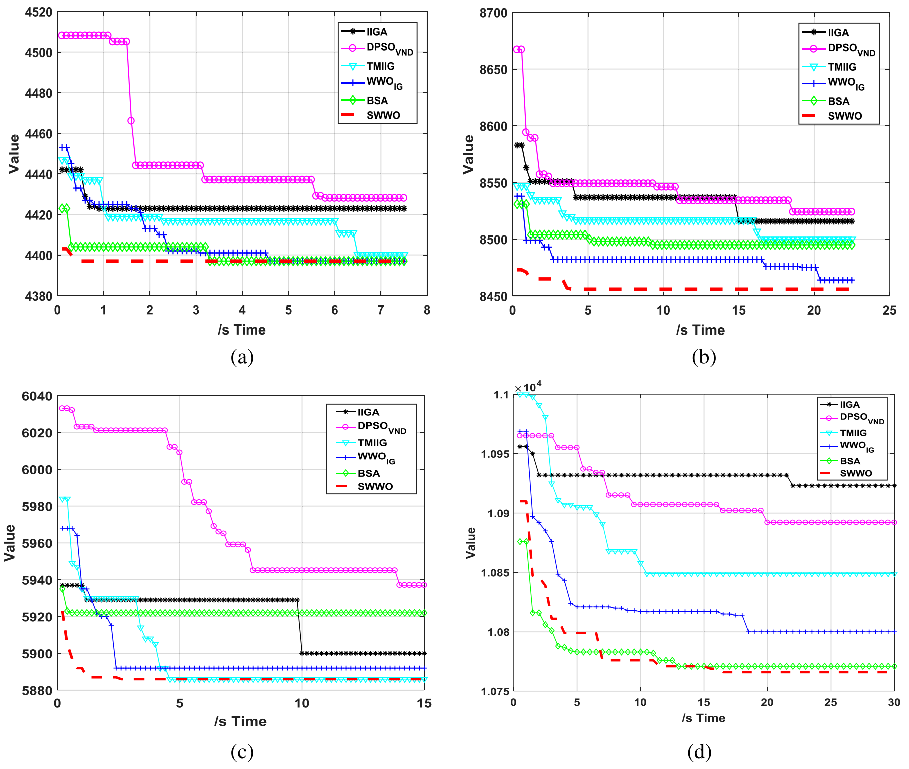
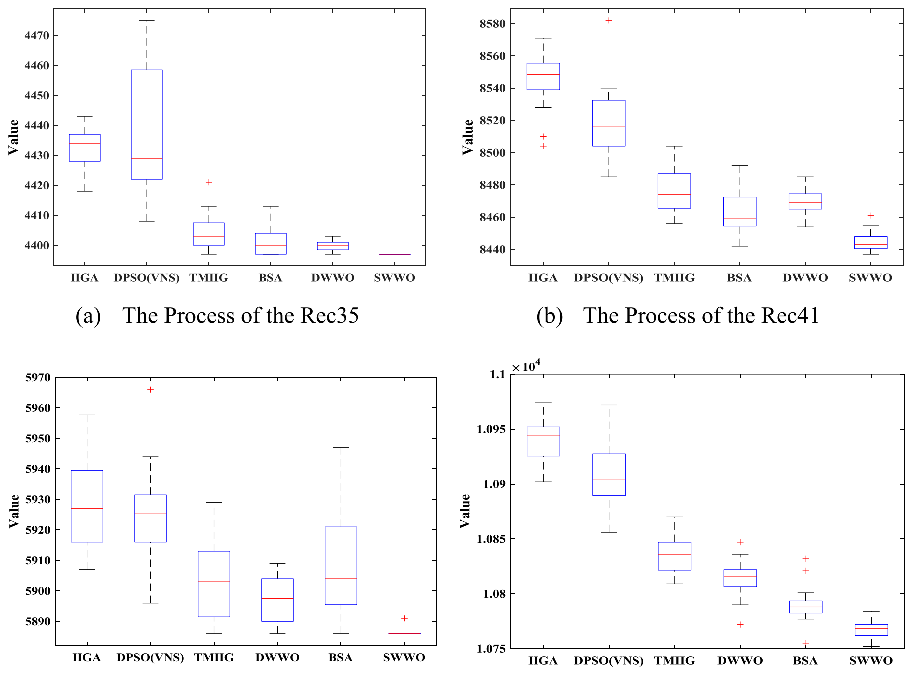


Figure 4. Convergence curves: (a) process of Rec35; (b) process of Rec41; (c) process of Ta055; (d) process of Ta090. IIGA = improved iterated greedy algorithm; DPSO$_{VND}$ = discrete particle swarm optimization–variable neighbourhood descent; TMIIG = tabu-mechanism improved iterated greedy algorithm; WWO$_{IG}$ = water wave optimization with iterated greedy; BSA = block-shifting simulated annealing algorithm; SWWO = single water wave optimization.

Table 2 shows the results after removing each component of SWWO. If the result is the worst solution among the four components of SWWO, it is certain that this part has the most significant impact on the algorithm, and thus this part is most important in the algorithm. From Table 2, the propagation operation is the most important to SWWO. According to the comparison of the results of SWWO and DWWO from Figures 4 and 5, it is not difficult to find that the convergence performance and stability of SWWO are better than those of DWWO. The performance of SWWO benefits from the combination of the self-adaptability of WWO with the depth search of the block-shift operator. In addition, the single wave mechanism is an innovative scheme in this article. DWWO is based on a population, while SWWO has a single sequence. The results in Table 5 show that the SWWO outperforms the DWWO.

4.2. Comparison of SWWO with existing algorithms

This section presents a comprehensive experimental evaluation and comparison of SWWO with other powerful methods. The SWWO is compared with other algorithms, including the improved iterated greedy algorithm (IIGA) (Pan, Wang, and Zhao 2008), hybrid discrete particle swarm optimization–variable neighbourhood descent (DPSO$_{VND}$) (Pan, Tasgetiren, and Liang, 2008), tabu-mechanism improved iterated greedy algorithm (TMIIG) (Ding *et al.* 2015a), block-shifting simulated annealing algorithm (BSA) (Ding *et al.* 2015b) and DWWO (Zhao *et al.* 2018a). In the following experiment, instances of different sizes are used to test the performance of the SWWO. To ensure the fairness of comparative experiments, all the comparison algorithms have been reimplemented in MATLAB. The results of all the comparison algorithms running at the same time are compared on



(a) The Process of the Rec35 (b) The Process of the Rec41

(c) The Process of the Ta055 (d) The Process of the Ta090

Figure 5. Boxplots for the instances: (a) process of Rec35; (b) process of Rec41; (c) process of Ta055; (d) process of Ta090. IIGA = improved iterated greedy algorithm; DPSO(VNS) = discrete particle swarm optimization–variable neighbourhood search; TMIIG = tabu-mechanism improved iterated greedy algorithm; BSA = block-shifting simulated annealing algorithm; DWWO = discrete water wave optimization; SWWO = single water wave optimization.

the same platform. In Carlier's and Reeve's instances, the maximum computational time of each compared algorithm is set as ($n/2 \times m \times 30$) ms, since the algorithms have different convergence rates and different search intensities. In Taillard's instances, the maximum running time of three different sizes ($n/2 \times m \times \tau$, $\tau = 30, 60, 90$) is adopted as the stopping criterion of the algorithms.

4.2.1. Comparison for Carlier's instances

Carlier's instances have small scale but long processing times. Table 3 shows the performance of all the algorithms in solving Carlier's problem. The result of SWWO is the same as IIGA, DPSOVNS, TMIIG and DWWO. The average relative percentage deviation (ARPD) and the SD

Table 3. Comparison of results based on Carlier's benchmark set.

		IIGA		DPSO$_{VND}$		TMIIG		BSA		DWWO		SWWO	
Instance	$n \times m$	ARPD	SD	ARPD	SD	ARPD	SD	ARPD	SD	ARPD	SD	ARPD	SD
Car1	11 × 5	0	0	0	0	0	0	0	0	0	0	**0**	**0**
Car2	13 × 4	0	0	0	0	0	0	0	0	0	0	**0**	**0**
Car3	12 × 5	0	0	0	0	0	0	0.041	8.199	0	0	**0**	**0**
Car4	14 × 4	0	0	0	0	0	0	0.656	56.467	0	0	**0**	**0**
Car5	10 × 6	0	0	0	0	0	0	2.453	158.863	0	0	**0**	**0**
Car6	8 × 9	0	0	0	0	0	0	0	0	0	0	**0**	**0**
Car7	7 × 7	0	0	0	0	0	0	0	0	0	0	**0**	**0**
Car8	8 × 8	0	0	0	0	0	0	0	0	0	0	**0**	**0**

Note: IIGA = improved iterated greedy algorithm; DPSO$_{VND}$ = discrete particle swarm optimization–variable neighbourhood descent; TMIIG = tabu-mechanism improved iterated greedy algorithm; BSA = block-shifting simulated annealing algorithm; DWWO = discrete water wave optimization; SWWO = single water wave optimization; ARPD = average relative percentage deviation; SD = standard deviation.

values of all the algorithms are 0. However, the BSA is inferior to the other algorithms in terms of convergence and robustness. In BSA, the main body is the block-shift operator. In SWWO, the propagation operation is based on the block-shift operator. The difference is that the size of the job block and the insertion position of the job block in BSA are random, whereas the block-shift operator is incorporated into the propagation operation in SWWO. The size of the job block is determined by the current wavelength, and the insertion position is the best of all the insertion methods. This is the main reason why SWWO is better than BSA in solving Carlier's problem.

4.2.2. Comparison for Reeve's instances

The results for Reeve's instances are shown in Table 4. The ARPD and SD values for all instances in SWWO are superior to the other comparison algorithms. The ARPD and SD values of the first 18 instances are 0, which indicates that the proposed algorithm quickly converges to the optimal solutions in solving various cases and the SWWO is very stable in solving Reeve's problems. The results show that the convergence performance and stability of SWWO are better than state-of-the-art algorithms for the NWFSP. Figure 4(a) and (b) show the convergence curves of the six comparative algorithms in solving Rec35 and Rec41, respectively. SWWO has faster convergence velocity and higher accuracy than the other algorithms, and also has the ability to escape from local optima quickly. Figure 5(a) and (b) show the boxplots of the six comparative algorithms in solving Rec35 and Rec41, respectively. The boxplot directly shows the discrete degree of the test data, which reflects the stability of SWWO. From Table 4, owing to the randomness of the main body of BSA, BSA does not obtain a desirable solution compared to the other algorithms in solving Reeve's problem. The superior performance of SWWO comes from the combination of the self-adaptability of WWO with the depth search of the block-shift operator.

Table 4. Comparison of results based on Reeve's benchmark set.

		IIGA		$DPSO_{VND}$		TMIIG		BSA		DWWO		SWWO	
Instance	$n \times m$	ARPD	SD	ARPD	SD	ARPD	SD	ARPD	SD	ARPD	SD	ARPD	SD
Rec01	20 × 5	0	0	0	0	0	0	0	0	0	0	**0**	**0**
Rec03	20 × 5	0	0	0	0	0	0	0	0	0	0	**0**	**0**
Rec05	20 × 5	0	0	0.033	1.118	0	0	0.033	0.5	0	0	**0**	**0**
Rec07	20 × 10	0	0	0	0	0	0	0.033	0.471	0	0	**0**	**0**
Rec09	20 × 10	0	0	0	0	0	0	0.22	6.185	0	0	**0**	**0**
Rec11	20 × 10	0	0	0	0	0	0	0.08	3.354	0	0	**0**	**0**
Rec13	20 × 15	0	0	0	0	0	0	0.301	0.471	0	0	**0**	**0**
Rec15	20 × 15	0	0	0	0	0	0	0.059	3.354	0	0	**0**	**0**
Rec17	20 × 15	0	0	0	0	0	0	0	0	0	0	**0**	**0**
Rec19	30 × 10	0.105	2.16	0.064	4.099	0.064	4.099	0.228	6.576	0.129	5.185	**0**	**0**
Rec21	30 × 10	0.183	2.967	0.242	5.047	0.053	2.141	0.201	6.074	0	0	**0**	**0**
Rec23	30 × 10	0.093	3.202	0.123	4.955	0	0	0.315	7.566	0	0	**0**	**0**
Rec25	30 × 15	0.014	1.118	0.093	1.491	0	0	0.023	1.863	0	0	**0**	**0**
Rec27	30 × 15	0.16	4.717	0.34	11.686	0.267	4.099	0.981	2.427	0.019	1.106	**0**	**0**
Rec29	30 × 15	0	0	0	0	0.228	7.5	0.36	3.287	0	0	**0**	**0**
Rec31	50 × 10	0.828	3.727	0.751	11.441	0.255	5.164	0.023	1.826	0.302	2.236	**0**	**0**
Rec33	50 × 10	1.454	8.654	1.123	20.846	0.622	9.674	0.426	9.856	0.509	4.646	**0**	**0**
Rec35	50 × 10	0.769	10.637	1.141	23.843	0.159	5.508	0.129	4.497	0.159	4.32	**0**	**0**
Rec37	75 × 20	1.384	16.077	1.076	26.742	0.591	6.472	0.212	9.504	0.63	16.631	**0.056**	**5.346**
Rec39	75 × 20	1.253	18.626	1.07	20.78	0.469	15.446	0.398	14.671	0.388	12.671	**0.089**	**6.292**
Rec41	75 × 20	1.23	15.91	0.857	20.188	0.551	14.626	0.257	18.98	0.351	7.993	**0.045**	**3.023**

Note: IIGA = improved iterated greedy algorithm; $DPSO_{VND}$ = discrete particle swarm optimization–variable neighbourhood descent; TMIIG = tabu-mechanism improved iterated greedy algorithm; BSA = block-shifting simulated annealing algorithm; DWWO = discrete water wave optimization; SWWO = single water wave optimization; ARPD = average relative percentage deviation; SD = standard deviation.

4.2.3. Comparison for Taillard's instances

For Taillard's problem, the different maximum run times ($n/2 \times m \times \tau$, $\tau = 30, 60, 90$) are set as the termination conditions of the algorithms. Table 5 shows the running results of all the comparison algorithms when τ is 30, 60 and 90. SWWO performs better than IIGA, DPSO$_{\text{VND}}$, TMIIG and DWWO in all instances. However, in comparison with BSA, SWWO is inferior to BSA in the case of 200 and 500 jobs. In BSA, the size of the block is a random number. In SWWO, the parameter λ in the λ-job-insert operation is determined by the maximum wavelength. As a parameter, the maximum wavelength is able to adapt to all instances. Choosing the right parameters is a very important issue in all algorithms. In general, the parameters have been chosen only to fit most instances, and for some instances the desired result is not obtained. However, if a random number is utilized to determine a certain parameter in each iteration, a high-efficiency solution is acquired in certain instances. Figure 4(c) and (d) show the convergence graphs of Ta55 and Ta90. It is obvious that the convergence rate of SWWO is faster than other algorithms. Figure 5(c) and (d) suggest that the SWWO still has superior stability when solving large-scale problems. Because all the comparison algorithms have the same operating environment, the proposed SWWO is better than the other comparison algorithms.

Table 5. Comparison of results based on Taillard's benchmark set.

		IIGA		DPSO$_{\text{VND}}$		TMIIG		BSA		DWWO		SWWO	
$n \times m$	τ	ARPD	SD	ARPD	SD	ARPD	SD	ARPD	SD	ARPD	SD	ARPD	SD
20 × 5	30	0.004	0.122	0.023	0.392	0	0	0.024	0.693	0	0	0	0
20 × 5	60	0.001	0.037	0.017	0.353	0	0	0.013	0.298	0	0	0	0
20 × 5	90	0	0	0.002	0.075	0	0	0.009	0.298	0	0	0	0
20 × 10	30	0.008	0.373	0.004	0.132	0	0	0.192	3.429	0	0	0	0
20 × 10	60	0.006	0.298	0.004	0.125	0	0	0.158	1.368	0	0	0	0
20 × 10	90	0.004	0.186	0.004	0.037	0	0	0.144	1.806	0	0	0	0
20 × 20	30	0.027	0	0.004	0.189	0.013	0.4	0.114	2.746	0	0	0.008	0.33
20 × 20	60	0.018	0.377	0.004	0.189	0.009	0.377	0.065	1.455	0	0	0	0
20 × 20	90	0.013	0.4	0.007	0.179	0.004	0.298	0.082	1.289	0	0	0	0
50 × 5	30	1.493	7.957	0.933	7.294	0.335	4.064	0.12	2.07	0.64	4.723	0.025	1.224
50 × 5	60	1.428	8.348	0.748	5.992	0.271	3.492	0.073	1.456	0.549	5.339	0.024	0.975
50 × 5	90	1.489	6.456	0.597	4.85	0.239	2.942	0.056	1.429	0.556	3.873	0.014	0.584
50 × 10	30	0.748	7.264	0.66	8.286	0.207	4.264	0.201	4.685	0.174	3.405	0.024	1.078
50 × 10	60	0.687	7.076	0.507	8.464	0.128	2.575	0.112	3.815	0.132	3.542	0.014	0.558
50 × 10	90	0.744	5.731	0.461	7.7	0.13	3.045	0.128	3.591	0.107	2.718	0.018	0.689
50 × 20	30	0.627	9.221	0.591	14.338	0.214	6.388	0.313	13.384	0.114	4.108	0.013	0.695
50 × 20	60	0.582	10.222	0.431	12.344	0.162	4.64	0.233	9.576	0.08	3.818	0.008	0.472
50 × 20	90	0.612	8.782	0.412	12.019	0.119	3.294	0.234	11.236	0.047	2.509	0.005	0.366
100 × 5	30	3.334	9.707	2.314	15.129	1.069	9.651	0.078	2.99	2.653	14.514	0.246	5.651
100 × 5	60	3.325	12.876	1.878	13.952	0.903	10.206	0.059	2.479	2.505	14.176	0.202	5.577
100 × 5	90	3.281	17.305	1.745	16.327	0.769	9.879	0.055	2.255	2.417	15.308	0.185	5.191
100 × 10	30	1.978	11.215	1.626	20.044	0.734	12.182	0.126	6.602	0.704	12.159	0.147	5.634
100 × 10	60	2.027	14.326	1.389	19.679	0.652	12.284	0.095	5.762	0.61	11.422	0.147	5.169
100 × 10	90	2.032	12.63	1.279	20.087	0.581	9.983	0.065	3.634	0.558	9.434	0.121	5.687
100 × 20	30	1.703	21.205	1.272	22.79	0.648	16.557	0.257	11.012	0.455	11.016	0.103	8.58
100 × 20	60	1.82	18.651	1.076	20.392	0.563	14.297	0.205	8.247	0.408	7.802	0.116	9.345
100 × 20	90	1.764	18.417	0.958	18.655	0.494	15.379	0.184	6.76	0.35	9.575	0.097	7.227
200 × 10	30	3.363	27.496	3.278	33.112	1.851	30.8	0.083	8.13	1.893	26.697	0.771	17.595
200 × 10	60	3.457	22.152	2.867	38.202	1.613	25.839	0.053	5.899	1.693	26.072	0.691	18.828
200 × 10	90	3.496	24.841	2.569	25.923	1.506	24.744	0.066	6.968	1.609	23.74	0.657	17.231
200 × 20	30	2.63	31.363	2.446	44.766	1.205	29.625	0.113	14.159	0.986	28.065	0.383	15.945
200 × 20	60	2.709	35.617	2.095	44.946	1.097	21.888	0.097	14.131	0.842	23.512	0.361	23.117
200 × 20	90	2.766	37.33	1.964	43.39	1.055	29.287	0.091	12.257	0.807	26.689	0.364	21.61
500 × 20	30	4.113	75.816	4.223	63.298	2.835	62.028	0.087	30.007	2.133	55.439	1.768	61.097
500 × 20	60	4.188	73.937	3.901	82.127	2.583	57.583	0.08	26.265	1.83	51.729	1.536	66.695
500 × 20	90	4.277	73.937	3.769	70.513	2.531	70.475	0.078	23.202	1.694	64.684	1.504	64.124

Note: IIGA = improved iterated greedy algorithm; DPSO$_{\text{VND}}$ = discrete particle swarm optimization–variable neighbourhood descent; TMIIG = tabu-mechanism improved iterated greedy algorithm; BSA = block-shifting simulated annealing algorithm; DWWO = discrete water wave optimization; SWWO = single water wave optimization; ARPD = average relative percentage deviation; SD = standard deviation.

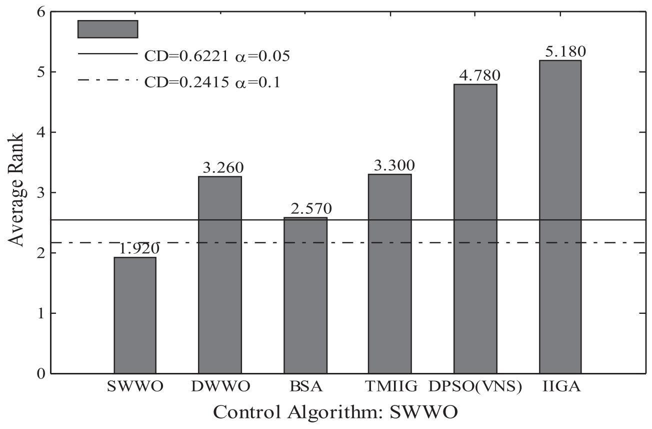


Figure 6. Graphical representation of the Bonferroni–Dunn's test. SWWO = single water wave optimization; DWWO = discrete water wave optimization; BSA = block-shifting simulated annealing algorithm; TMIIG = tabu-mechanism improved iterated greedy algorithm; DPSO(VNS) = discrete particle swarm optimization–variable neighbourhood search; IIGA = improved iterated greedy algorithm; CD = critical difference.

To detect significant differences between SWWO and the five competitors, the Friedman's test (García *et al.* 2009) is utilized to rank the algorithms. The SWWO has the best ranking among the six algorithms. To evaluate the significance level of all the algorithms, an additional Bonferroni–Dunn's method is applied as a *post hoc* procedure to calculate the critical difference (CD) (Equation 14) for comparing their differences with $\alpha = 0.05$ and $\alpha = 0.1$:

$$\mathrm{CD} = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{14}$$

where parameters k and N are the number of algorithms and the number of instances, respectively. In the experimental evaluations, $k = 6$ and $N = 120$. When $\alpha = 0.1$, q_α is 2.327, and when $\alpha = 0.05$, q_α is 2.576, from Table B.16 [two-tailed $\alpha(2)$] of Zar (1999). Tables 4 and 5 present the non-parametric test results of SWWO and the comparison algorithms in solving Reeve's and Taillard's problems. The results of Bonferroni–Dunn's test are given in Figure 6. The SWWO and comparison algorithms have significant differences at $\alpha = 0.05$ and $\alpha = 0.1$.

According to the experiments and comparisons with other algorithms, the proposed SWWO is an efficient, effective and robust algorithm for solving NWFSPs. The superiority of the proposed SWWO can be mainly attributed to the following aspects; (1) SWWO is a single sequence rather than being based on population, so the impact of population is ignored in the design of the algorithm to improve its performance; (2) k-job insert obtains more desirable candidate solutions than the basic interpolation strategy; (3) the speed-up methods adopted effectively enhance the searching efficiency of algorithm; (4) the MVNS further improves the quality of the solutions; and (5) statistical analysis is used to illustrate the superiority of SWWO in terms of convergence and stability. These satisfactory outcomes validate the advantages of SWWO and demonstrate that it is efficient and stable for solving NWFSPs.

5. Conclusions and future work

In this article, a novel SWWO algorithm, based on the framework of WWO, is applied to solve the NWFSP with the objective of minimizing the makespan. First, a neglected but efficient search mechanism named the block-shift mechanism is introduced into the propagation operation. Secondly, the crossover operation is applied to obtain the optimal solution in SWWO. Thirdly, the VNS operation is introduced into the breaking operation to increase the local search capability of the algorithm. Finally, the computational results and comparisons based on three benchmark sets suggest the effectiveness of SWWO for the NWFSP.

Several issues deserve more in-depth study in the future. First, as an adaptive framework, SWWO constantly adjusts the size of the parameters to change the search space. However, setting reasonable upper and lower bounds for the parameters according to the scale of the problem is still a key issue. Secondly, the convergence and performance of SWWO should be analysed from a theoretical perspective. Moreover, the SWWO could be utilized in other fields.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was financially supported by the National Natural Science Foundation of China [grant number 61663023]. It was also supported by the Key Research Programs of Science and Technology Commission Foundation of Gansu Province [2017GS10817], Wenzhou Public Welfare Science and Technology Project [G20170016], and the General and Special Program of the Postdoctoral Science Foundation of China, the Science Foundation for Distinguished Youth Scholars of Lanzhou University of Technology [grant numbers 2012M521802, 2013T60889 and J201405, respectively].

ORCID

Fuqing Zhao http://orcid.org/0000-0002-7336-9699

References

Agnetis, A. 2000. "Scheduling No-Wait Robotic Cells with Two and Three Machines." *European Journal of Operational Research* 123 (2): 303–314.

Aldowaisan, Tariq, and Ali Allahverdi. 2012. "Minimizing Total Tardiness in No-Wait Flowshops." *Foundations of Computing and Decision Sciences* 37 (3): 149–162.

Allahverdi, Ali, Harun Aydilek, and Asiye Aydilek. 2018. "No-Wait Flowshop Scheduling Problem with Two Criteria; Total Tardiness and Makespan." *European Journal of Operational Research* 269 (2): 590–601.

Balamurugan, R., M. Siva, and L. Lakshminarasimman. 2017. "Water Wave Optimization Algorithm for Solving Multi-area Economic Dispatch Problem." *International Journal of Computer Applications* 167 (5): 19–27.

Carlier, Jacques. 1978. "Ordonnancements à contraintes disjonctives." *RAIRO - Operations Research* 12 (4): 333–350.

Ding, Jian Ya, Shiji Song, Jatinder N. D. Gupta, Rui Zhang, Raymond Chiong, and Cheng Wu. 2015a. "An Improved Iterated Greedy Algorithm with a Tabu-Based Reconstruction Strategy for the No-Wait Flowshop Scheduling Problem." *Applied Soft Computing* 30: 604–613.

Ding, Jian Ya, Shiji Song, Rui Zhang, Siwei Zhou, and Cheng Wu. 2015b. "A Novel Block-Shifting Simulated Annealing Algorithm for the No-Wait Flowshop Scheduling Problem." In *Proceedings of the 2015 IEEE Congress on Evolutionary Computation*, 2768–2774. Sendai, Japan: IEEE.

França, P. M., G. Tin Jr, and L. S. Buriol. 2006. "Genetic Algorithms for the No-Wait Flowshop Sequencing Problem with Time Restrictions." *International Journal of Production Research* 44 (5): 939–957.

Gao, Kai Zhou, Quan Ke Pan, and Jun Qing Li. 2011. "Discrete Harmony Search Algorithm for the No-Wait Flow Shop Scheduling Problem with Total Flow Time Criterion." *The International Journal of Advanced Manufacturing Technology* 56 (5–8): 683–692.

García, Salvador, Daniel Molina, Manuel Lozano, and Francisco Herrera. 2009. "A Study on the Use of Non-Parametric Tests for Analyzing the Evolutionary Algorithms' Behaviour: A Case Study on The CEC'2005 Special Session on Real Parameter Optimization." *Journal of Heuristics* 15 (6): 617–644.

Hall, Nicholas G., and Chelliah Sriskandarajah. 1996. "A Survey of Machine Scheduling Problems with Blocking and No-Wait in Process." *Operations Research* 44 (3): 510–525.

Hansen, Pierre, and Nenad Mladenovic. 2001. "Variable Neighborhood Search: Principles and Applications." *European Journal of Operational Research* 130 (3): 449–467.

Huang, J. D., J. J. Liu, Q. X. Chen, and N. Mao. 2017. "Minimizing Makespan in a Two-Stage Flow Shop with Parallel Batch-Processing Machines and Re-entrant Jobs." *Engineering Optimization* 49 (6): 1010–1023.

Kirlik, Gokhan, and Ceyda Oguz. 2012. "A Variable Neighborhood Search for Minimizing Total Weighted Tardiness with Sequence Dependent Setup Times on a Single Machine." *Computers & Operations Research* 39 (7): 1506–1520.

Naderi, B., Majid Khalili, and Alireza Arshadi Khamseh. 2014. "Mathematical Models and a Hunting Search Algorithm for the No-wait Flowshop Scheduling with Parallel Machines." *International Journal of Production Research* 52 (9): 2667–2681.

Osman, I. H., and C. N. Potts. 1989. “Simulated Annealing for Permutation Flow-Shop Scheduling.” *Omega* 17 (6): 551–557.
Pan, Quan Ke, M. Fatih Tasgetiren, and Yun Chia Liang. 2008. “A Discrete Particle Swarm Optimization Algorithm for the No-Wait Flowshop Scheduling Problem.” *Computers & Operations Research* 35 (9): 2807–2839.
Pan, Quan Ke, Ling Wang, and Bao Hua Zhao. 2008. “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.
Qian, B., L. Wang, R. Hu, D. X. Huang, and X. Wang. 2009. “A DE-Based Approach to No-Wait Flow-Shop Scheduling.” *Computers & Industrial Engineering* 57 (3): 787–805.
Raaymakers, W. H. M., and J. A. Hoogeveen. 2000. “Scheduling Multipurpose Batch Process Industries with No-Wait Restrictions by Simulated Annealing.” *European Journal of Operational Research* 126 (1): 131–151.
Rajendran, Chandrasekharan. 1994. “A No-Wait Flowshop Scheduling Heuristic to Minimize Makespan.” *Journal of the Operational Research Society* 45 (4): 472–478.
Reeves, Colin R. 1995. “A Genetic Algorithm for Flowshop Sequencing.” *Computers & Operations Research* 22 (1): 5–13.
Shabtay, Dvir. 2012. “The Just-in-Time Scheduling Problem in a Flow-Shop Scheduling System.” *European Journal of Operational Research* 216 (3): 521–532.
Shao, Zhongshi, Dechang Pi, and Weishi Shao. 2018a. “Estimation of Distribution Algorithm with Path Relinking for the Blocking Flow-Shop Scheduling Problem.” *Engineering Optimization* 50 (5): 894–916.
Shao, Zhongshi, Dechang Pi, and Weishi Shao. 2018b. “A Novel Discrete Water Wave Optimization Algorithm for Blocking Flow-Shop Scheduling Problem with Sequence-Dependent Setup Times.” *Swarm and Evolutionary Computation* 40: 53–75.
Taillard, E. 1993. “Benchmarks for Basic Scheduling Problems.” *European Journal of Operational Research* 64 (2): 278–285.
Tran, Trung Hieu, and Kien Ming Ng. 2013. “A Hybrid Water Flow Algorithm for Multi-objective Flexible Flow Shop Scheduling Problems.” *Engineering Optimization* 45 (4): 483–502.
Wang, Chuyang, Xiaoping Li, and Qian Wang. 2008. “Iterative Local Search Algorithm for No-Wait Flowshop Scheduling Problems to Minimize Makespan.” In *12th International Conference on Computer Supported Cooperative Work in Design*, 908–912. Xi’an: IEEE.
Wang, Chuyang, Xiaoping Li, and Qian Wang. 2010. “Accelerated Tabu Search for No-Wait Flowshop Scheduling Problem with Maximum Lateness Criterion.” *European Journal of Operational Research* 206 (1): 64–72.
Wang, Yu, Jiafu Tang, Zhendong Pan, and Chongjun Yan. 2015. “Particle Swarm Optimization-Based Planning and Scheduling for a Laminar-Flow Operating Room with Downstream Resources.” *Soft Computing* 19 (10): 2913–2926.
Wang, Zhenbo, Wenxun Xing, and Fengshan Bai. 2005. “No-Wait Flexible Flowshop Scheduling with No-Idle Machines.” *Operations Research Letters* 33 (6): 609–614.
Wu, Xiao Bei, Jie Liao, and Zhi Cheng Wang. 2015. “Water Wave Optimization for the Traveling Salesman Problem.” In *The International Conference on Intelligent Computing (ICIC 2015)*, Vol. 9225, 137–146. Fuzhou: Springer.
Yun, Xin, Xiaoyi Feng, Lyu Xin, Shouyang Wang, and Bo Liu. 2016. “A Novel Water Wave Optimization Based Memetic Algorithm for Flow-Shop Scheduling.” In *Proceedings of the 2016 IEEE Congress on Evolutionary Computation*, 1971–1976. Vancouver, BC: IEEE.
Zar, J. H. 1999. *Biostatistical Analysis*. Upper Saddle River, NJ: Prentice Hall.
Zhao, Fuqing, Yang Liu, Zhongshi Shao, Xin Jiang, Chuck Zhang, and Junbiao Wang. 2016. “A Chaotic Local Search Based Bacterial Foraging Algorithm and Its Application to a Permutation Flow-Shop Scheduling Problem.” *International Journal of Computer Integrated Manufacturing* 29 (9): 962–981.
Zhao, Fuqing, Yang Liu, Yi Zhang, Weimin Ma, and Chuck Zhang. 2017. “A Hybrid Harmony Search Algorithm with Efficient Job Sequence Scheme and Variable Neighborhood Search for the Permutation Flow Shop Scheduling Problems.” *Engineering Applications of Artificial Intelligence* 65: 178–199.
Zhao, Fuqing, Huan Liu, Yi Zhang, Weimin Ma, and Chuck Zhang. 2018a. “A Discrete Water Wave Optimization Algorithm for No-Wait Flow Shop Scheduling Problem.” *Expert Systems with Applications* 91: 347–363.
Zhao, Fuqing, Yang Liu, Chuck Zhang, and Junbiao Wang. 2015. “A Self-Adaptive Harmony PSO Search Algorithm and Its Performance Analysis.” *Expert Systems with Applications* 42 (21): 7436–7455.
Zhao, Fuqing, Shuo Qin, Yi Zhang, Weimin Ma, Chuck Zhang, and Houbin Song. 2019. “A Two-Stage Differential Biogeography-Based Optimization Algorithm and its Performance Analysis.” *Expert Systems with Applications* 115: 329–345.
Zhao, Fuqing, Feilong Xue, Yi Zhang, Weimin Ma, Chuck Zhang, and Houbin Song. 2018b. “A Hybrid Algorithm Based on Self-Adaptive Gravitational Search Algorithm and Differential Evolution.” *Expert Systems with Applications* 113: 515–530.
Zheng, Yu-Jun. 2015. “Water Wave Optimization: A New Nature-Inspired Metaheuristic.” *Computers & Operations Research* 55: 1–11.