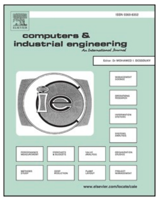
Computers & Industrial Engineering 153 (2021) 107082

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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



A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem

Fuqing Zhao [*], Lixin Zhang, Jie Cao, Jianxin Tang [*]

School of Computer and Communication Technology, Lanzhou University of Technology, Lanzhou 730050, China

A R T I C L E I N F O

Keywords:
Water wave optimization algorithm
Distributed assembly no-idle flow-shop scheduling problem
Reinforcement learning
Variable neighborhood search

A B S T R A C T

The distributed assembly no-idle flow-shop scheduling problem (DANIFSP) is a novel and considerable model, which is suitable for the modern supply chains and manufacturing systems. In this study, a cooperative water wave optimization algorithm, named CWWO, is proposed to address the DANIFSP with the goal of minimizing the maximum assembly completion time. In the propagation phase, a reinforcement learning mechanism based on the framework of the VNS is designed to balance the exploration and exploitation of the CWWO algorithm. Afterwards, the path-relinking combined with the VNS method as the modified breaking operator is introduced to enhance the capability of local search. Furthermore, a multi-neighborhood perturbation strategy in the refraction phase is applied to extract knowledge information to increase the probability of escaping the local optimal. Moreover, the comprehensive experimental program is executed to calibrate the control parameters of the CWWO algorithm and illustrate the cooperative effect of the three modified operations. The performance of the CWWO algorithm is verified on the benchmark set, and the experimental results demonstrated the stability and validity of the CWWO algorithm.

1. Introduction

With the development of the global economy, the market presents superior requirements for the manufacturing system. For instance, online response to customer order, dynamically adjust the production scheduling, tacking disrupt events and multi-task allocation. The scheduling system has become the bottleneck and key issue for the manufacturing system. In the scheduling problem, production assembly is a significant portion of the customized manufacturing system owing to its practical significance for the manufacturing systems and industrial processes. With the development of the global economy, the market presents superior requirements for the manufacturing system. For instance, online response to customer order, dynamically adjust the production scheduling, tacking disrupt events and multi-task allocation. The scheduling system has become the bottleneck and key issue for the manufacturing system. In the scheduling problem, production assembly is a significant portion of the customized manufacturing system owing to its practical significance for the manufacturing systems and industrial processes. The production systems are collectively known as the assembly flow-shop scheduling problems (ASFP) proposed by Koulamas and Kyparisis (2001). According to the two scholars, the assembly production systems demanded that a mass variety of end products is designed by utilizing the modular structures to control the production cost. Furthermore, the ASFPs are common in bluetooth headset, personal computers (Potts et al., 1995) and the illustration chart of ASFPs is shown in Fig. 1.

The distributed production environments are widely applied in real life. The distributed manufacturing systems are effective in increasing the product quality, reducing production costs and decreasing management risks (Hatami et al., 2015). In practice, the application of the distributed permutation flow-shop scheduling problem (DPFSP) (Meng et al., 2019; Naderi & Ruiz, 2010) is relatively widespread due to its challenging. The methods applied to solve the DPFSP are increasing emerged. The traditional mathematical methods, such as enumeration method, branch and bound, are appropriate for the small size and simple problems owing to its high computational complexity. The artificial intelligence methods provide several approaches and methodologies to solve scheduling problems. However, the two methods are uncontrollable in terms of timeliness and performance. Therefore, the heuristics (Campbell et al., 1970; Nawaz et al., 1983) and meta-heuristics are used to solve the complex combinatorial optimization problem. In recent years, the increasing meta-heuristics, such as artificial bee colony

* Corresponding authors.
E-mail addresses: fzhao2000@hotmail.com (F. Zhao), jxtang2011@hotmail.com (J. Tang).

https://doi.org/10.1016/j.cie.2020.107082

Available online 24 December 2020

algorithm (ABC) (Kaveh & Bakhshpoori, 2019), biogeography-based optimization algorithm (BBO) (Simon, 2008; Zhao, Qin, et al., 2018; Zhao, Qin, et al., 2019), gravitational search optimization algorithm (GSA) (Rashedi et al., 2009; Zhao, Xue, et al., 2018, 2019), water wave optimization algorithm (WWO) (Zhao et al., 2017; Zheng, 2015) collaborative optimization algorithm (COA) (Chen et al., 2019), are emerged for solving a variety of continuous optimization problems and discrete optimization problems. The operation mechanisms of these meta-heuristics are different from traditional mathematical methods. The unique algorithm theory and operation mechanism are guaranteed the exploration and exploitation of the meta-heuristics achieve a certain balance during the iterative process.

The distributed assembly permutation flow-shop scheduling problem (DAPFSP) proposed by Hatami et al. (2013) is the combination of the ASFP and the DPFSP for the various potential applications. The DAPFSP is composed of two portions, the production stage and assembly stage (Hatami et al., 2015). Since the complexity of the DAPFSP is higher than the complexity of the single shop scheduling problem, the DAPFSP is an NP-hard problem. The complexity of the DAPFSP shows that the methods adopted are efficient and time-efficient. The main task of the DAPFSP is to allocate all the jobs processed to each factory reasonably and effectively, further to determine the processing order and product assembly sequence of all the jobs in the factory (Lin et al., 2017). There are various meta-heuristics proposed to solve the DAPFSP. Hatami et al. (2015) proposed two simple heuristics and two meta-heuristics to solve the DAPFSP with sequence-dependent setup times (SDST). The two simple constructive heuristic methods (H_1 and H_2) based on the NR_1 and NR_2 (Naderi & Ruiz, 2010) were introduced in this study. Afterwards, two meta-heuristics, which are based on the variable neighborhood descent (VND) and iterated greedy (IG) algorithm, were presented to solve the DAPFSP-SDST. In the literature (Pan et al., 2019), a mixed-integer linear model, three constructive heuristics, two variable neighborhood search methods, and an iterated greedy algorithm were proposed to suit the requirements of different CPU time and solution quality. There are three variants of the discrete invasive weed optimization (DIWO), including a two-level discrete invasive weed optimization (TDIWO), a discrete invasive weed optimization with hybrid search operators (HDIWO) and an HDIWO with selection probability, proposed by Sang et al. (2018) to solve the DAPFSP with total flowtime criterion. An estimation of the distribution algorithm based on the memetic algorithm (EDAMA) (Wang & Wang, 2015) was proposed to solve the DAPFSP to minimize the maximum completion time. The exploration and exploitation of the EDA were incorporated within the framework of the MA. Furthermore, a novel selective-enhancing sampling mechanism and a critical-path-based local search strategy were proposed to enhance the search capability. Ferone et al. (2020) presented an efficient and parameter-less algorithm (BR-ILS), which is utilized as a biased-randomized iterated local search meta-heuristic. A backtracking search hyper-heuristic (BS-HH) algorithm was proposed by Lin et al. (2017) to solve the DAPFSP. In the BS-HH, ten simple and effective heuristic rules were designed to construct a set of low-level heuristics (LLHs) manipulated by the backtracking search optimization algorithm (BSA) (Civicioglu, 2013). Obviously, the DAPFSP has received considerable attention due to its complexity and the extension of the DAPFSP is gradually increased, such as the distributed assembly no-idle flow-shop scheduling problem (DANIFSP). The DANIFSP addressed in this paper, is an extension of the DAPFSP and first proposed by Shao, Dechang, et al. (2019). The DANIFSP consists of the DPFSP with no-idle constraint stage and assembly stage. Note that, there are few references about the DANIFSP.

The water wave optimization algorithm (WWO) proposed by Yu Jun Zheng (2015) is a population-based meta-heuristic algorithm. There are three operations included in the WWO, propagation operation, breaking operation, and refraction operation, respectively. In the WWO algorithm, the propagation operator is a global search behavior and has a certain ability to escape local optima, the refraction operator and breaking operator is a local search process to enhance the exploitation of the WWO algorithm (Zheng et al., 2019). After the WWO algorithm was proposed, it has been attracted the attention of various researchers to adopt the WWO algorithm for solving multitudinous problems. Zhang et al. (2019) proposed the wind-driven WWO (WDWWO) algorithm to solve challenging engineering optimization problems. Zhao, Zhang, et al. (2019)presented an improved water wave optimization algorithm with the single wave mechanism (SWWO) for the no-wait flow-shop

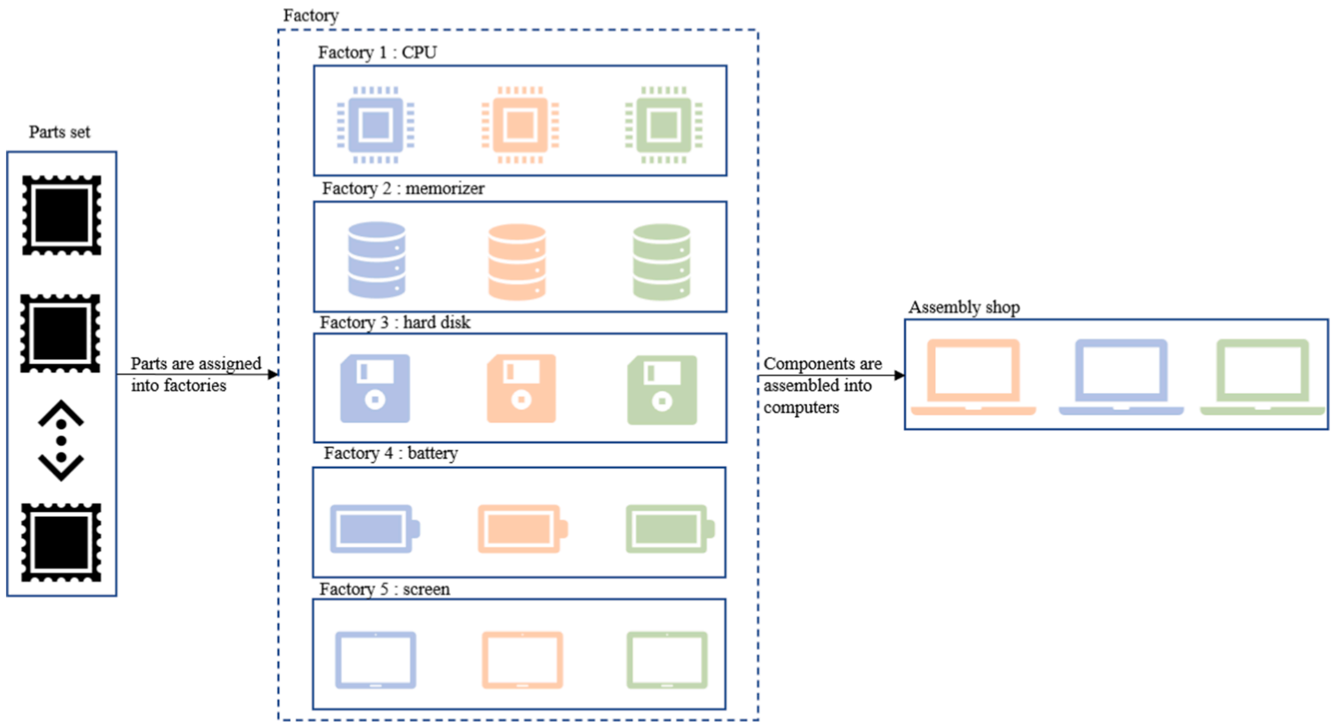


Fig. 1. The assembly process for personal computers.

scheduling problem (NWFSP). Shao et al. (2017) proposed a novel discrete water wave optimization algorithm for the blocking flow-shop scheduling problem with sequence-dependent setup times (BFSP-SDST). Additionally, the WWO algorithm was adopted to solve neural network optimization (Liu et al., 2019), the traveling salesman problem (TSP) (Wu et al., 2015), the no-idle flow-shop scheduling problem (NIFSP) (Zhao et al., 2020) and multi-objective blocking flow-shop scheduling problem (Shao, Pi, et al., 2019). In this study, the WWO algorithm is applied to address the DANIFSP and the schematic diagram is illustrated in Fig. 2. Therefore, a collaborative water wave optimization algorithm with a three-stage variable neighborhood search (CWWO) is proposed for solving the DANIFSP with minimizing the assembly completion time. The proposed algorithm is combined with the framework of the variable neighborhood search (VNS) (Hansen & Mladenović, 2005) at the three search stages, including the propagation operation based on the reinforcement learning with the VNS, the breaking operation based on path-relinking with the VNS and refraction operation based on knowledge-learning with the VNS. The contributions of this study are generalized as follows.

(1) A novel variant of the WWO algorithm that collaborated with a three-stage variable neighborhood search (VNS) and named CWWO is proposed to solve the distributed assembly no-idle flow-shop scheduling problem (DANIFSP). Moreover, the specific mixed integer programming model (MILP) is explained in this study.
(2) The reinforcement learning is embedded into the proposed algorithm. Specifically, the $Q-learning$, as the perturbation strategy in the VNS, is introduced into the propagation operation to control the wavelength and further to balance the exploration and exploitation of the proposed algorithm. The path-linking, as the portion of the breaking operation combined with the VNS, is designed to enhance the capability of the local search. Furthermore, a multi-neighborhood perturbation strategy is applied to help the proposed algorithm to escape the local optimal according to the knowledge learned from the historical optimal.
(3) In this study, benchmark set designed for the DAFSP is adopted to test the performance of the proposed algorithm and the compared algorithms. Compared with the state-of-the-art algorithms, the updated best solutions obtained by the proposed algorithm are given in the supplementary materials. Afterwards, the experimental results analyzed by certain statistical approaches are illustrated that the stability and effective of the proposed algorithm outperformed the state-of-the-arts. Additionally, owing to the DANIFSP is a novel scheduling problem, only one literature proposed the related theories and solving methods.

The remainder of the paper is organized as follows. The problem statement of DANIFSP is given in Section 2. In Section 3, the proposed algorithm is described in detail. In Section 4, several experiments are used to verify the performance of the proposed algorithm. Section 5 summarized conclusions and future work.

2. The distributed assembly no-idle flow-shop problem

The notations in this paper are described as follows.

Notations	Description
n	The number of jobs, $j = 1, 2, 3, \cdots, n$ where j is an index for jobs.
m	The number of machines, $i = 1, 2, 3, \cdots, m$ where i is an index for machines.
F	The number of factories, $f = 1, 2, 3, \cdots, F$ where f is an index for factories.
P	The number of products, $p = 1, 2, 3, \cdots, P$ where p is an index for products.
$\{O_1, O_2, .., O_P\}$	The set of the product P.
S	The population.
M_A	The assembly machine.
$tt_{j,i}$	The processing times.
t_p	The processing time of product p on the assembly stage.
C_p^n	The processing completed time of the last job belongs to$\delta(p)$
CA_p	The completed time of product p on the assembly stage.
$C_{i,j,f}$	The completed time of the job i processed on the machine j in the factory f.
$X_{j,k,f}$	A binary variable. If job j occupies the position k in factory $X_{j,k,f} = 1$. Otherwise, $X_{j,k,f} = 0$.
$C_{M_A,p}$	The completed time of product p on the assembly machine M_A.
π	Feasible scheduling and $\pi = [\pi^1, \pi^2, \cdots, \pi^F]$.
δ	The assembly sequence $\delta = [\delta(1), \delta(2), \cdots, \delta(P)]$
π^f	The sequence of jobs in factory f and $\pi^f = [\pi^f(1), \pi^f(2), \cdots, \pi^f(n^f)]$.
n^f	The total number of jobs allotted to factory f.
T	A sufficiently large positive number.
$R_{j,p}$	A binary parameter.If job j belongs to product p, $R_{j,p} = 1$. Otherwise, $R_{j,p} = 0$.
C_j	A continuous variable denotes the completed time of job j.
$C_{max}(\pi)$	The makespan.

The illustration diagram of the DANIFSP is shown in Fig. 3. The

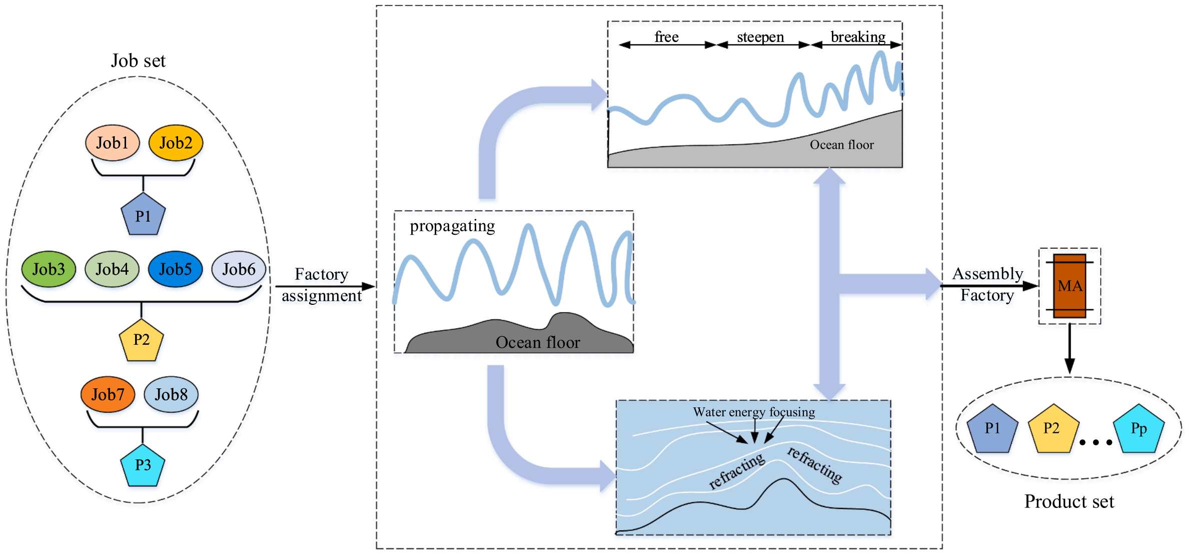


Fig. 2. The WWO algorithm applied to solve the DANIFSP.

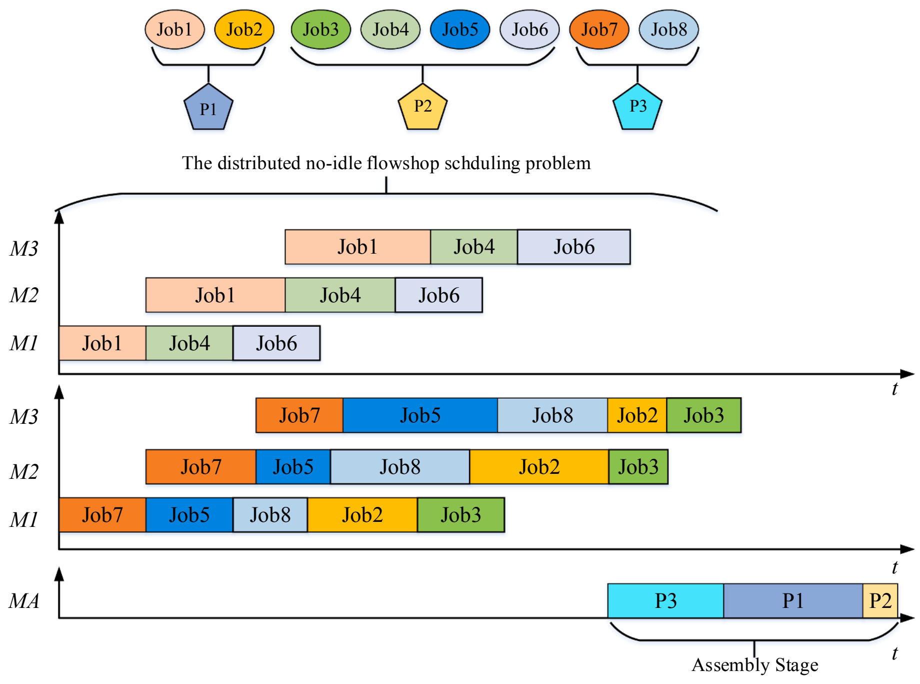


Fig. 3. The illustration diagram of the DANIFSP.

DANIFSP is described as $n$jobs $\{J_1, J_2, \cdots, J_n\}$ are allocated to $f$factories $\{F_1, F_2, \cdots, F_f\}$ with the identical m machines $\{M_1, M_2, \cdots, M_m\}$. p products $\{P_1, P_2, \cdots, P_s\}$, which are composed of jobs are assembled on the assembly machine M_A. There are two phases enclosed in the DANIFSP, the distributed flow-shop scheduling problem with no-idle constraint and the assembled product stage.

In the first phase, n jobs are allocated into f factories to process. Each factory is a no-idle flow-shop environment with $m$machines. After a job is assigned to a certain factory, all processes for the job are processed in the factory and the job is not transferred to other factories. The machines and jobs in all factories are satisfied with the characteristics of no-idle and traditional permutation flow-shop. Moreover, all jobs are mutual independence and processed at zero time. Meanwhile, the conditions are required to satisfy as follows. (1) Each job is only processed on one machine and each machine only processes one job at certain point. (2) Each operation is not stopped once it starts. (3) The setup time of the machine and the transfer time of the job are ignored. Additionally, there is an infinite buffer between the two stages of production and assembly, so is machines. In the product assembly stage, an assembly machine M_A is used to assemble the completed jobs into products. The product P_s is assembled from N_s jobs. That is, $\sum_{s=1}^{P} |N_s| = n$ represents that each job belongs to a definite product. All the jobs belong to the product P_s are assembled into the product after the processing is completed and the assembly time is t_p. The goal of DANIFSP is to determine the processing factory of the job and the processing sequence of the job in the factory to guarantee a certain production index optimal. The objective of the DANIFSP addressed in this paper is to minimize the maximum assembly completion time C_{max}. The mixed-integer linear programming model (MILP) (Shao, Dechang, et al., 2019) is given as follows.

$$\sum_{k=1,k\neq j}^{n} \sum_{f=1}^{F} X_{j,k,f} = 1 \quad j \in \{1, 2, 3, \cdots, n\} \tag{1}$$

$$\sum_{j=1,k\neq j}^{n} \sum_{f=1}^{F} X_{j,k,f} = 1 \quad j \in \{1, 2, 3, \cdots, n\} \tag{2}$$

$$X_{k,j,f} + X_{j,k,f} \le 1 \quad \forall j \in \{1, 2, 3, \cdots, n-1\}, j < k \tag{3}$$

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

$$C_{k,i,f} = C_{k-1,i,f} + \sum_{j=1}^{n} X_{j,k,f} \cdot tt_{j,i} \quad \forall k < 1, i, f \tag{5}$$

$$C_{k,i,f} \ge C_{k,i-1,f} + \sum_{j=1}^{n} X_{j,k,f} \cdot tt_{j,i} \quad \forall k, i, f \tag{6}$$

$$C_i \ge C_{k,m,f} - T(1 - X_{j,k,f}) \quad \forall j, k, f \tag{7}$$

$$CA_p \ge CA_{p-1} + t_p \tag{8}$$

$$CA_p \ge C_j + t_p - T(1 - R_{j,p}) \quad \forall j, p \tag{9}$$

$$C_{max} \ge CA_p \quad \forall p \tag{10}$$

$$X_{j,k,f} \in \{0, 1\} \quad \forall k, i, f \tag{11}$$

$$C_j \ge 0 \quad \forall j \tag{12}$$

$$CA_p \ge 0 \quad \forall p \tag{13}$$

$$C_{k,i,f} > 0 \quad \forall k, i, f \tag{14}$$

Constraint (1) guarantees that each job is only assigned to one factory and only appeared once in the factory. Constraint (2) explains that the possibility for n positions where all jobs should be allocated to each factory. Constraint (3) controls and ensures that a job is both a predecessor and successor of another job at the same time. Constraint (4) illustrates that the completed time of the first job processed on the first machine in each factory, which is ensured that the start time of the first machine in each factory is 0. Constraint (5) is the relationship between the completed times of two adjacent jobs on the identical machine in the distribution factory, which ensures that all machines have no idle time. Constraint (6) is that each job is only start to process after the previous job processed on the machine in an identical factory, which ensures that multiple jobs are not processed on one machine. Constraint (7) confirms the completed time of each job. Constraint (8) guarantees the relationship between two adjacent assembled products. Constraint (9) means that each product is only start to assemble after the jobs belong to the products are processed completely. Constraint Constraint (10) defines the maximum assembly completion time. Constraint (11) is the value range of the decision variable. Constraint (12) – Constraint (14) limit that the completed times of all processes are greater than 0.

There are two phases for calculating the makespan C_{max} of the DANIFSP. (1) The completed times of all jobs are calculated. (2) The assembly sequences are determined by the completed time of all jobs and the assembly completion times of each product are computed. The optimization objective is to minimize the maximum assembly completion time of all products. According to the literature (Shao et al., 2019), the computational process of the makespan for the DANIFSP is given as follows. Note that, $\pi^f = [\pi^f(1), \pi^f(2), \cdots, \pi^f(n^f)]$ is defined as the job sequence in the factory f, where n^f is the total number of the jobs in the factory f. $\pi_E^f = [\pi_E^f(1), \pi_E^f(2), \cdots, \pi_E^f(i)]$ is denoted as the partial sequence of the π^f and the range of i is from 1 to n^f. Additionally, the forward pass calculation (Tasgetiren et al., 2013) is adopted to calculate the completed processing time.

$$F(\pi_E^f(1), i, i+1) = tt_{\pi_E^f(1), i+1}, \quad i = 1, 2, \cdots, m-1 \tag{15}$$

$$F(\pi_E^f(j), i, i+1) = max\left\{F(\pi_E^f(j-1), i, i+1) - tt_{\pi_E^f(j), i}, 0\right\} + tt_{\pi_E^f(1), i+1}, \quad j = 2, 3, \cdots, n^f, \quad i = 1, 2, \cdots, m-1 \tag{16}$$

$$C_{\pi^f(n^f), m} = \sum_{i=1}^{m-1} F(\pi_E^f(n^f), i, i+1) + \sum_{j=1}^{n^f} tt_{\pi_E^f(j), 1} \tag{17}$$

$$C_{\pi^f(j), m} = C_{\pi^f(j+1), m} - tt_{\pi^f(j+1), m}, \quad j = n^f - 1, n^f - 2, \cdots, 1 \tag{18}$$

where $F(\pi_E^f(j), i, i+1)$ is the minimum completion difference between the last job $\pi_E^f(j)$ on the machine i and $i+1$ in Eq. (15). In Eq. (16), $C_{\pi^f(n^f), m}$ is the completion time of the last job $\pi^f(n^f)$, which is computed by the sum of the minimum completion difference with the addition of the sum of the processing time of the jobs in the factory f. $C_{\pi^f(j), m}$, which is calculated by the completed time of the job $j+1$ subtracts the processing time of the job $j+1$, represents the completion time of the other job $\pi^f(j)$ in Eq. (17). Denoted that, $\delta = [\delta(1), \delta(2), \cdots, \delta(P)]$, which is decided by the latest completion time is the assembly sequence. C_1^O are the assembly completion time of the first product and the assembly completion time for each product is calculated as follows.

$$CA_1 = C_1^n + t_1 \tag{19}$$

$$CA_p = max\left\{CA_{p-1}, C_p^n\right\} + t_p, \quad p = 2, 3, \cdots, P \tag{20}$$

$$C_{max}(\pi) = CA_P \tag{21}$$

where Eq. (19) means that the completed time of production 1 (CA_1) on the assembly stage is the sum of the C_1^n, which means the completed time of the last job belongs to production $\delta(1)$, and the processing time of product 1 (t_1) on the assembly stage. Eq. (20) represents that the completed time of product p (CA_p) on the assembly stage is the sum of the maximum value, which is between the completed time of product $p-1$ (CA_{p-1}) and the completed time (C_p^n) of the last job belongs to $\delta(p)$, and the processing time of product p (t_p) on the assembly stage. Eq. (21) is the makespan, which is the optimization objective of the DANIFSP.

3. The proposed algorithm

3.1. Initialized population and job allocation rule

The initialized method has an important influence on the population-based intelligent optimization algorithms. In order to minimize the maximum assembly completion time for the DANIFSP, not only the completed processing time of all jobs are the shortest, but also the assembly completion time is optimal, which are to consider. Therefore, a reasonable initialization method is designed to improve the quality of the population and to ensure the diversity of the population. The jobs belong to the product are required to process together as much as possible to ensure the compactness of the related jobs, and further to ensure that the product is assembled as soon as possible (Wang & Wang, 2015). Naderi and Ruiz (2010) proposed two job allocation rules named NR_1 and NR_2, which are only considered that the completed time of all jobs is optimal. However, the assembly time is consideration. During the experimential process is designed, the handling methods of the processing time and the assembly time are impact on the quality of the population. Therefore, it is necessary to consider the assembly time of the product and it is not advisable to use the sorted of the average or standard deviation of the processing time to generate the initial sequence. In this study, the method of randomly initializing the job sequence is applied to generate the initial sequence set. Afterwards, the NR_a (Shao et al., 2019) is adopted as the allocation rule to allot jobs, Furthermore, the makespan of the individual is calculated by the formulas given in Eqs. (15)–(21) from Section 2. The initialized population and the best individual are recorded. The pseudocode of the initialization is given in Procedure 1.

Procedure 1. Initialization

```
Input: $NP, n, m, f, p, assemblyset, assemblytime, processingtime$
Output: $S$
1   for $i = 1$ to $NP$ do
2       $sequence = generaterandonpermutation(job)$;
3       $job_i$ is allocated according to the job allocation rule $NR_a$;
4       $C_i$ is calculated according to the computing formulas given in Section 2;
5       $S_i$ is recorded.
```

(continued on next page)

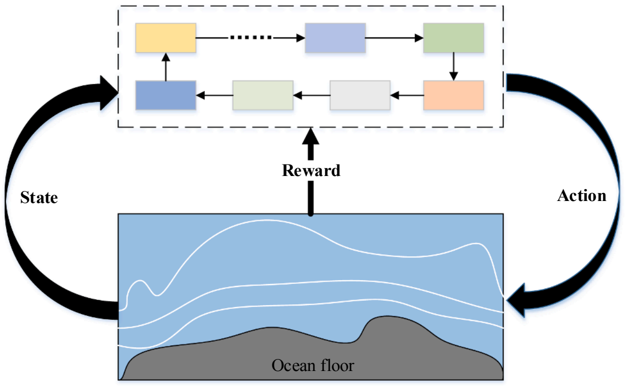


Fig. 4. The illustration for the $Q-learning$.

(*continued*)

```
6    end for
7    The best sequence and the best $C_{max}$ are recorded.
```

3.2. The propagation operation based on reinforcement learning

The goal of the propagation operation, which is a global search behavior in the WWO algorithm, is to balance the exploration and exploitation. In the WWO algorithm, each wave has three attributes, position, wavelength, and wave height. The way of generating a new wave is determined by the propagation operation and the search capability of the WWO algorithm is to a great extent impacted by the propagation operator. In the proposed algorithm, a propagation operator based on the reinforcement learning (Nussenbaum & Hartley, 2019) with the variable neighborhood search as a framework, is introduced to balance the exploration and exploitation, and further to generate the current optimal. $Q-learning$ (Watkins & Dayan, 1992), one of the reinforcement learning, is a free-model learning method, which provides a learning capability for the intelligence system in the Markov environment to select the optimal action using the experienced action. There are four elements in the $Q-learning$, including agent, the environmental state, action, and reward. The illustration plot is presented in Fig. 4 and the definition of the $Q-learning$ is shown in Eq. (22).

$$Q(S,A)\leftarrow(1-\alpha)Q(S,A)+\alpha[R(S,a)+\gamma\max_{a}Q(S',a_i)] \quad (22)$$

where α is the learning rate, γ is the discount factor. S means state and A represent action adopted in each state. R is the different reward achieved by the different actions implemented in each state. $Q(S,A)$ is regarded as the experience and memory of the current individual. $R(S,a)$ is the actual experience gained. $[R(S,a)+\gamma\max_{a}Q(S',a_i)]$ means the new achieved experience and $\max_{a}Q(S',a_i)$ represents the experience expected to be learned in the next state based on the previous experience.

In the propagation phase, a strategy to balance the exploration and exploitation is required to choose. Therefore, the $Q-learning$ is adopted to decide the position of the new generated wave and find the current optimal. Each individual is corresponding to an agent, that is a wave. The environmental state is simulated into the seabed. Under the control of the wavelength, the propagation operator is executed on each wave. When the global optimal is found, the wavelength is updated. Assumed that the optimal solution found by the current agent is better than the *hbI*, where *hbI* means the historical best individual contained the historical best sequence and the historically best C_{max}, the current agent is rewarded until the current agent gains the maximum reward. The formula of the $Q-learning$ is redefined as follows.

$$Q(S,A)\leftarrow(1-\alpha)Q(S,A)+\alpha[R(S,a)+rand^{*}\max_{a}Q(S',a_i)] \quad (23)$$

where *rand* is a random positive number from 0 to 1. The reason for γ replaced *rand* is that a self-learning mechanism is applied to control the reward obtained by the current wave. The greater *rand*, the greater role of the $\max_{a}Q(S',a_i)$. The randomness of *rand* is to help the proposed algorithm balance the global search and local search. The wavelength is used to determine the action and reward Therefore, the λ is updated by the makespan and the formula is as follows.

$$\lambda_i=\lambda_{max}-(\lambda_{max}-\lambda_{min})^{*}a^{-\frac{currentCmax(i)-minCmax}{(maxCmax-minCmax)^{*}(1+\varepsilon)}} \quad (24)$$

where *currentCmax* is the maximum assembly completion time of the current individual, *minCmax* is the current optimal, *maxCmax* is the current worst, *a* is the wavelength reduction coefficient and set to 1.0026. $\lambda_{max}=NP$ and $\lambda_{min}=1$ are the maximum wavelength and the minimum wavelength, respectively. The process of the propagation operator based on the reinforcement learning with the variable neighborhood search is given in Procedure 2 and the pseudocode about $Q-learning$ is shown in Procedure 3.

Procedure 2. Propagation

```
Input: $S_i, hbI$
Output: $searchjobIndex1, fitnesses_1$
1            $serachIndividual = S_i$;
2            $k_{max} = 3; k = 1$;
3            while $k \le k_{max}$ do
4              if $k == 1$
5                for each factory do
6                  $currentC_{max}1$ = Reinforcement Learning (Procedure 3);
7                end for
8              end if $flag = ture; bestL = hbI$;
9              while $flag$ do $flag = false$;
10               for each factory do
11                 the local search is executed on each factory;
12               end for
13               if $serachIndividual.C_{max} < bestL$ do
14                 $bestL = serachIndividual.C_{max}; flag = ture; k = 1$;
15               else $k++$; end if
16             end while
17           end while
```

As seen from Procedure 2 and Procedure 3, the propagation operation in the proposed algorithm is interpreted as that the $Q-learning$ is operated as the perturbation search strategy based on the VNS, which is realized the operation collaboration.

Procedure 3. $Q-learning$

```
Input: $serachIndividual, Q_{table}, R_{table}, \alpha, \lambda$
Output: $currentC_{max}1$
1       $currentIndividual = searchIndividual$;
2       for $i = 1$ to $n$ do
3         The action $next$ is selected from $Q_{table}$; The reward is defined according to
        the $R_{table}$;
4           for $j = 1$ to $n$ do
5             for $k = 1$ to $n$ do
6               if $flag[next] = false$ do
7                 if $Q_{table}[j,k] = 1$ do
8                   The $Q_{table}$ is updated according to the Eq. (23);
9                 else $Q_{table}[j,k] = 0$;
10              else $Q_{table}[j,k] = \lambda$;
11              end if
12        end for the sequence of $currentIndividual$ is arranged according to the new
        $Q_{table}$;
13      end for
14      The $currentC_{max}1$ is calculated.
```

3.3. The breaking operation based on path-relinking

When the current best solution is found by an individual, the breaking operator is executed as the local search in the WWO algorithm.

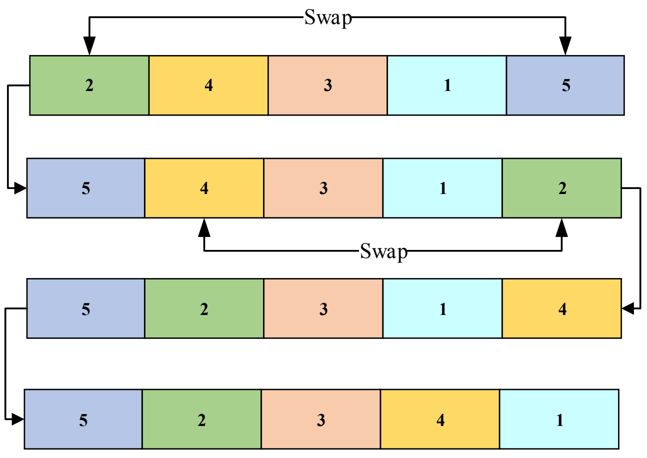


Fig. 5. The diagram of PR.

Thus, it's interpreted that the breaking operation is applied to further search behavior around the current optimal to enhance the search accuracy of the WWO algorithm. However, the local search capability provided by the breaking operator is relatively weak. Therefore, a breaking operation based on the path-relinking (PR) (Glover, 1997) combined with the framework of the VNS is designed to improve the intensive search near the current optimal solution in the proposed algorithm. The diagram of the PR is presented in Fig. 5.

The specific process for the introduced breaking operator is as follows. First, the historical optimal solution is used as a guided solution, a series of swap operations are executed between the current solution and the guided solution to generate a set of new solutions. The new solutions are evaluated to choose the optimal solution, donated as an intermediate solution. The process is named *VPath−relink* and the pseudocode is shown in Procedure 4 and the process is descriped as follows. The swap operation is executed on the history best individual and the current individual, certain middle solutions are generated by the swap operation every time. When the middle solution is different from the history best individual and the current individual, it is stored in the set *M*. All solutions are evaluated to select a desirable solution. If the makespan of the desirable solution is better than the makespans of the history best individual and the current individual, the current individual is replaced by the desirable solution.

Procedure 4. *VPath−relink*

```
Input: $serachIndividual, hbI$
Output: $currentC_{max}$
1   $serachIndividual.C_{max} = currentC_{max}1$;
2   $k_{max} = 3; k = 1$;
3   while $k \le k_{max}$ do
4     $flag = ture; bestL = currentC_{max}1$;
5     while $flag$ do
6       $flag = false$;
7       for $i = 1$ to $NP$ do
8         Find the position $k$ of $serachIndividual$ in $hbI$;
9         Swop the position of $serachIndividual$ and $hbI(k)$ to generate new $\pi'$.
10      end for
11      if $\pi'.C_{max} < bestL$ and $\pi'.C_{max} < hbI.C_{max}$ do
12        $bestL = \pi'.C_{max}; flag = ture; k = 1$;
13      else
14        $k++$;
15      end if
16    end while
17    $currentC_{max} = bestL$;
18  end while
```

Second, a variable neighborhood search strategy is performed on the intermediate solution to strengthen the intensive search and to prevent the algorithm from falling into a local optimum and improve the convergence speed of the algorithm. Finally, the individuals in the population are evaluated to choose the optimal solution. The pseudocode of the modified breaking operator is shown in Procedure 5.

Procedure 5. Breaking

```
Input: $currentC_{max}1, hbI$
Output: $searchjobIndex2, fitnesses_2$
1   $serachIndividual.C_{max} = currentC_{max}1$;
2   $k_{max} = 3; k = 1$;
3    while $k \le k_{max}$ do
4      If $k == 1$
6        $currentC_{max}2 = VPath-relink$ (Procedure 4);
7      end if
8      $flag = ture; bestL = hbI$;
9        while $flag$ do
10         $flag = false$;
11         $VNS(currentC_{max}2)$;
12           if $serachIndividual.C_{max} < bestL$ do
13             $bestL = serachIndividual.C_{max}; flag = ture; k = 1$;
14           else
15             $k++$;
```

(continued on next column)

(continued)

```
16           end if
17       end while
18   end while
```

3.4. Multi-neighborhood perturbation strategy

The refraction operator also serves as a local search process in the WWO algorithm, which is used to effectively avoid the algorithm stagnation. According to the experimental and theoretical analysis, the refraction operation has an ability to learn. In the proposed algorithm, a multi-neighborhood perturbation strategy is presented to learn the knowledge stored in the historical best information to help the proposed algorithm escape the local optimal and self-evolution. In the perturbation phase of the VNS, a *Paturain_swap_insert* strategy is introduced as the perturbation. To be specific, a factory is randomly selected and the *swap_insert* operation is performed on the jobs in the factory, further to find the current minimum makespan. The process of the *Paturain_swap_insert* is shown in Procedure 6.

Procedure 6. *Paturain_swap_insert*

```
Input: $serachIndividual$
Output: $currentC_{max}$
1   Find the factory $f$ with the best makespan;
2   Exchanged jobs from factories to generate new$\pi'$;
3   Select $a$ job from $\pi'$ randomly as $insertjob$;
4   Remove $insertjob$ from $\pi'$;
5   $bestL = serachIndividual.C_{max}$;
6   for $i = 1$ to $F$ do
7     for $k = 1$ to $n^f$ do
8       $value$ = Compute assembly time after $insertjob$ is inserted into factory $i$.
9     end for
10  end for
11  if $value < bestL$
12    $bestL = value$; record the new $\pi$;
13  end if
14  $currentC_{max} = bestL$;
```

The *Destruction−Construction* (Pan et al., 2008) as a local search method is adopted to determine the knowledge obtained from the historical best information. That is, *ds* jobs are randomly taken from all factories, and reinserted to the best possible position according to the job allocation rule to obtain the optimal solution. In short, the current water wave learns from the historical optimal solution to find the optimal solution in the neighborhood structure. The pseudocode of the multi-neighborhood perturbation strategy is shown in Procedure 7.

Procedure 7. Refraction

```
Input: $currentC_{max}1, hbI$
Output: $searchjobIndex3, fitnesses_3$
1   $serachIndividual.C_{max} = currentC_{max}1$;
2   $k_{max} = 3; k = 1$;
3   while $k \le k_{max}$ do
4     if $k == 1$
6       $currentC_{max}3 = Paturain\_insert\_swap$ (Procedure 6);
7     end if
8       $flag = ture; bestL = hbI.C_{max}$;
9       while $flag$ do $flag = false$;
10        $Destruction-Construction$;
11        if $serachIndividual.C_{max} < bestL$ do
12          $bestL = serachIndividual.C_{max}; flag = ture; k = 1$;
13        else $k++$; end if
14      end while
15  end while
```

At last, Fig. 6 shows the illustration of *Destruction−Construction* and the pseudocode of the modified refraction operation is given in Procedure 7.

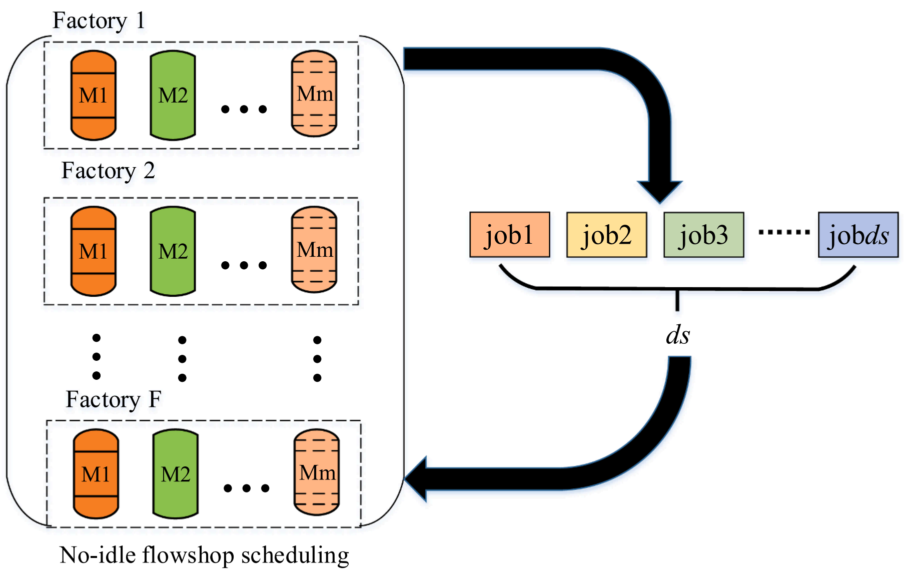


Fig. 6. The diagram of *Destruction −Construction*.

3.5. The proposed algorithm for the DANIFSP

According to the above-detailed description of the improvement operations for the cooperative water wave optimization algorithm based on the three-stage variable neighborhood search (CWWO), the operation mechanisms and overall framework of the CWWO algorithm are specifically introduced in this section. The pseudocode of the proposed algorithm is shown in Algorithm 1. Variable neighborhood search (VNS) is a heuristic algorithm. During the algorithmic search process, VNS expands the search space based on the concentrated search of the changing neighborhood structure to obtain the local optimal, and then re-intensively searches around the changing neighborhood structure according to the current optimal solution to obtain another local optimal replaces the process of the current optimal solution. The perturbation, local search, and neighborhood changes are repeatedly performed. The WWO algorithm is to perform a propagation operation on each solution in the population after the population is initialized. If the solution obtained by the propagation operation is better than the current solution and the historical optimal solution, the breaking operation is performed, otherwise, the wave height is reduced to zero, and refraction operation is executed. In the CWWO algorithm, VNS is used as an auxiliary strategy in each portion of the WWO algorithm to coordinate the proposed algorithm to achieve the algorithm collaboration and operation collaboration.

Algorithm 1. The pseudocode of the proposed algorithm

```
Input: $NP$, $\alpha$, $\lambda_{max}$, $\lambda_{min}$, $T_0$,$h_{max}$
Output: the optimal solution ($hbs.C_{max}$)
1     $[S, fitnesses]$ = Procedure 1. Initialization
2     Record the historical optimal solution $hbf$ and the historical optimal
    sequence $hbs$.
3     while the stopping termination condition do
4       for $i = 1$ to $NP$ do
5         $[searchjobIndex1, fitnesses_1]$ = Procedure 2. Propagation
6         If $fitnesses_1 < fitnesses_i$
7           $fitnesses_i = fitnesses_1, S_i = searchjobIndex1$
8           If $fitnesses_1 < hbf$
9           $hbf = fitnesses_1; hbs = \pi_1;$
10          $[searchjobIndex2, fitnesses_2]$ = Procedure 5. Breaking
11            If $fitnesses_2 < hbf$
12              $hbf = fitnesses_2; hbs = searchjobIndex2;$
13            endif
14          $h(i) = h_{max};$
15        else $h(i) -= 1$
16          If $h(i) == 0$
17            $[searchjobIndex3, fitnesses_3]$ = Procedure 7. Refraction
18              If $fitnesses_3 < hbf$
19              $hbf = fitnesses_3; hbs = searchjobIndex3;$
20              end If
21            else the first inferior solution operation ($T_0$).
22          end If
23          end If
24      Update wavelength; The second inferior solution operation ($T_0$).
25      end for
26    end while
```

In the population initialization phase, job sequences randomly generated are distributed to each factory according to the factory allocation rules to generate population sequence sets and record the current optimal solution as the historical optimal solution. In the propagation operation based on $Q-learning$, the perturbation search in VNS is improved to reinforcement learning with training and rewards. The action is determined by the wavelength and the action further determines the reward. The maximum reward is to obtain the current local optimal solution. Compared to the current optimal solution with the current solution and the historical optimal solution. If the current optimal solution is greater than the latter two, then the wave breaking operation based on Path-Relinking is performed, otherwise the wave height is reduced to zero, the refraction operation based on knowledge-learning is executed. In the improved breaking operation, the path-relinking replaces the perturbation search in the VNS, which is used to enhance the intensive search near the current optimal solution to improve the accuracy of the proposed algorithm. In the refraction stage, a multi-neighborhood perturbation strategy based on the knowledge-learning is designed. The insertion and swap of jobs are used as a perturbation strategy, and the destruction reconstruction, which utilizes the knowledge information from the historical optimal, is used as a local search strategy to effectively avoid the algorithm search stagnation. Moreover, the first acceptance criterion, simulated annealing (SA) (Bahman Naderi & Azab, 2015), is applied in the modified refraction operation to accept the neighborhood solution. Finally, after the wavelength is updated, the second acceptance criterion is used to determine whether the neighborhood solution after the specific operation is received with a certain probability. Note that, the adjusted temperature T_0 of the two acceptance criteria are identical. The flowchart of the CWWO algorithm is shown in Fig. 7.

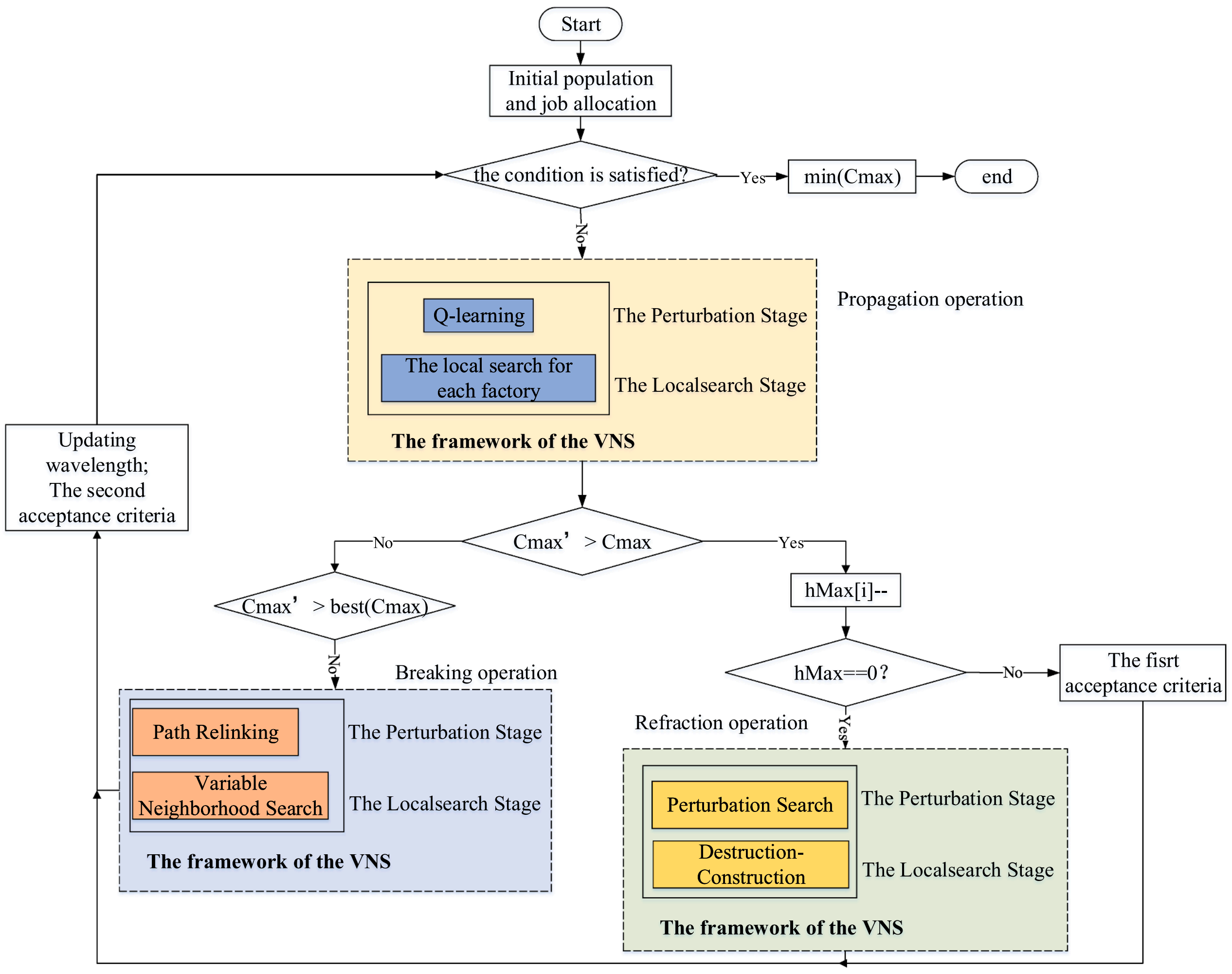


Fig. 7. The flowchart of the proposed algorithm.

4. Experiential analysis

In this section, the feasibility and effectiveness of the proposed algorithm are verified with the computational experiments, which are carried out on a personal computer (PC) with Intel (R) Core (TM) i7-6700 CPU 3.4 GHz and 8.00 GB memory with the Windows Server 2012 Operating System. In this paper, the average relative percentage deviation (ARPD) and the standard deviation (SD) are adopted to evaluate the performance of the CWWO and the compared algorithms. The calculation equation of the ARPD is given in in *Eq.*(25). All compared algorithms and methods are coded with Java and evaluated with the same termination criterion $T_{max} = n*m*f$ milliseconds. Furthermore, the Minitab 18 statistical software and MATLAB are applied to analyze the experimental results and generate statistical graphs.

$$ARPD = \frac{1}{R}\sum_{i=1}^{R}\frac{C_{max}(i) - C_{opt}}{C_{opt}}*100 \tag{25}$$

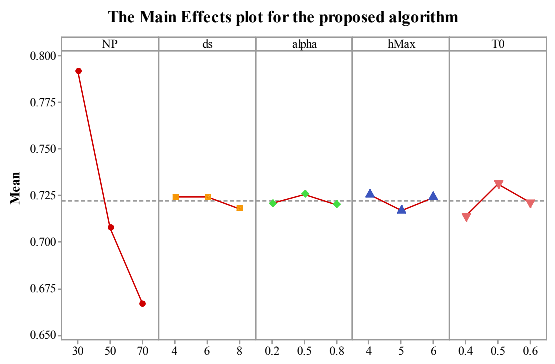
where the C_i is the solution obtained by given algorithm tested the *ith* run of a certain instance, the C_{opt} means the best solution found so far, R is the number of run times the algorithm on a single case.

4.1. The control parameters analysis

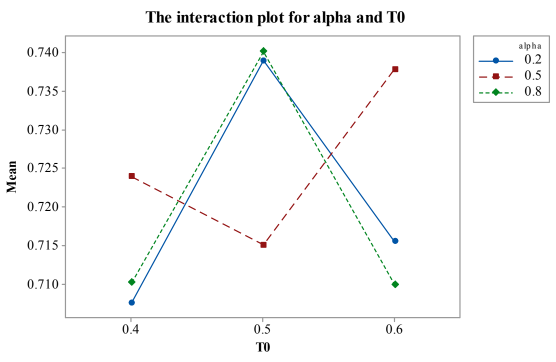
In this section, the design of experiments method (DOE) (Jones, 2010) is applied to analyze the control parameters on the impact of the proposed algorithm in detail. The specific parameters include population size (*NP*), the destruction size (*ds*), the wave height (*hMax*), the learning rate (*alpha*), and the adjusted temperature (T_0). The levels of the chosen parameters are listed as follows: $NP = \{30, 50, 70\}$, $ds =$

Table 1
The ANOVA results of Parameters for the proposed algorithm.

Source	Sum of squares	Degrees of freedom	Mean Square	F-ratio	p-Value
NP	0.65655	2	0.32827	159.2	**0**
ds	0.00205	2	0.00103	0.5	0.6087
alpha	0.00149	2	0.00075	0.36	0.6967
hMax	0.00355	2	0.00178	0.86	0.4244
T_0	0.01255	2	0.00628	3.04	**0.0499**
*NP*ds*	0.01531	4	0.00383	1.86	0.1197
*NP*alpha*	0.00609	4	0.00152	0.74	0.5668
*NP*hMax*	0.00244	4	0.00061	0.3	0.8807
$NP*T_0$	0.01062	4	0.00265	1.29	0.2765
*ds*alpha*	0.00604	4	0.00151	0.73	0.5708
*ds*hMax*	0.00251	4	0.00063	0.3	0.8748
$ds*T_0$	0.0014	4	0.00035	0.17	0.9539
*alpha*hMax*	0.01101	4	0.00275	1.33	0.2585
$alpha*T_0$	0.02536	4	0.00634	3.07	**0.0175**
$hMax*T_0$	0.00477	4	0.00119	0.58	0.6783
Error	0.3959	192	0.00206		
Total	1.15765	242			



(a) The Main effects plot

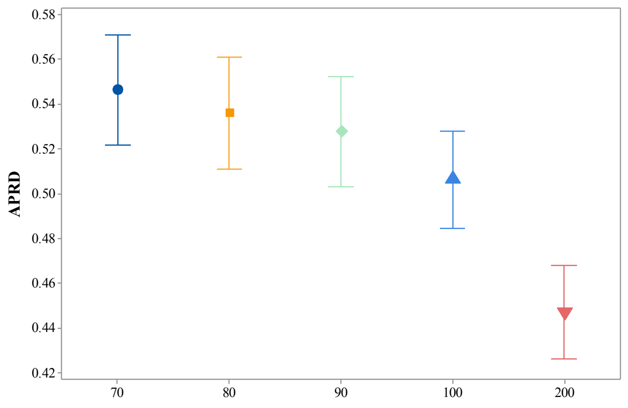


(b) The Interaction plot

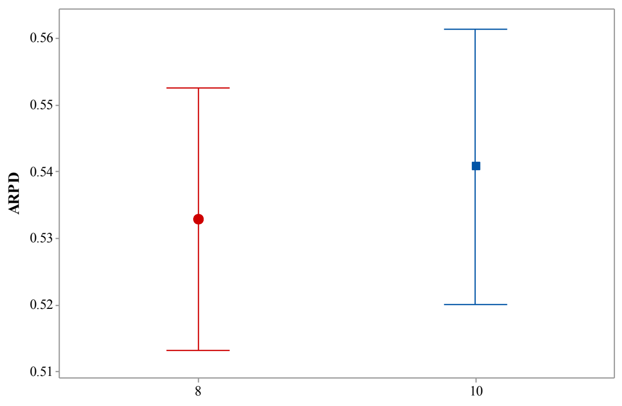
Fig. 8. Parameters analysis plot for the proposed algorithm.

$\{4,6,8\}$, *hMax* $= \{4,5,6\}$, *alpha* $= \{0.2, 0.5, 0.8\}$, $T_0 = \{0.4, 0.5, 0.6\}$. All parameter combinations are in total of 3*3*3*3*3 = 243. To avoid biased or overfitting results, the new test suit is generated to verify the control parameters. The results of the control parameters are carried out on 48 instances randomly generated, including $n = \{100, 200\}$, $m = \{5, 10\}$, $F = \{4, 6\}$, $P = \{30, 40\}$. The processing time is randomly generated from 1 to 99. Each combination includes three instances and runs 5 times. The stopping termination criterion is set to $T_{max} = n*m*f$ms. The experimental results are investigated by the multifactor analysis of variance (ANOVA), which demonstrates whether interactions between the parameters are significant (Afifi & Azen, 1972).

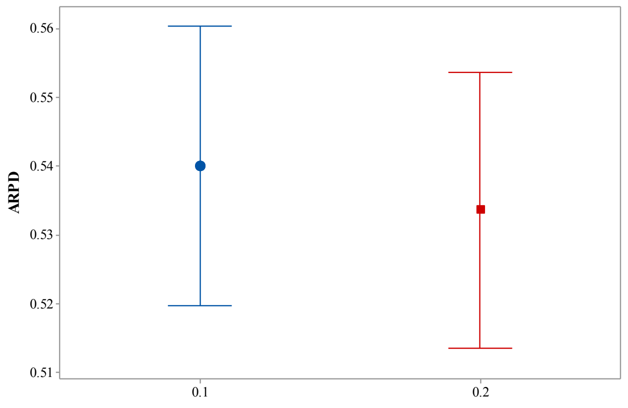
As seen from Table 1, the population size (*NP*) and the adjusted temperature (T_0) are the most significant parameters for the proposed algorithm owing to the two $p-value$ are smaller than 0.05. Moreover, there is an interaction effect between α and T_0. Combined with Fig. 8, Fig. 8 shows the parameters analysis plot for the proposed algorithm. In Fig. 8(a), *NP* is set to 70, which is illustrated that the experimental results of the proposed algorithm are optimal. When the *NP* is set to 30, the results are the worst. Therefore, the small population size leads to lose the diversity of the population in the process of iterations. If the population size is large, a lot of search time is wasted in the search process. For the adjusted temperature, the smaller the initial value, the better the



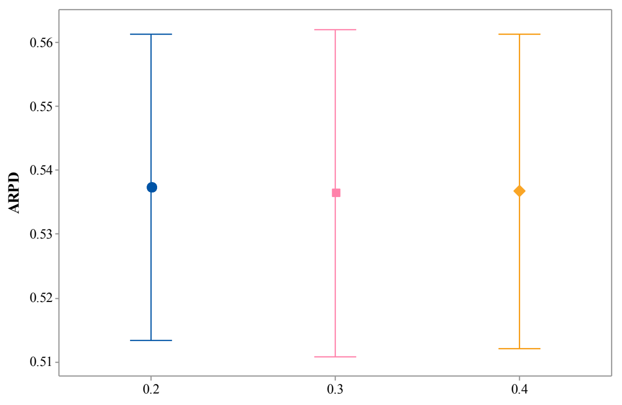
(a) Interval plot and 95% confidence level intervals of *NP*



(b) Interval plot and 95% confidence level intervals of *ds*



(c) Interval plot and 95% confidence level intervals of *alpha*



(d) Interval plot and 95% confidence level intervals of T_0

Fig. 9. Interval plot with 95% confidence level intervals of certain parameters.

neighborhood solution received. Afterwards, the interaction effect of $alpha^{*}T_0$ declared that the modified propagation operator is considerably thorough for the searching solution to choose the best solution in the population. Therefore, the satisfactory combination of the control parameter is $NP = 70$, $ds = 8$, $alpha = 0.2$, $hMax = 5$ and $T_0 = 0.4$.

Although the optimal parameter configuration has been given in Fig. 8, it's not affirmed that whether the maximum values of *NP* and *ds*, and the minimum values of *alpha* and T_0 are applied to assist the proposed algorithm with obtaining the promising performance. Therefore, an additional experiment is executed to find the best parameter values. In the following experiment, the four parameters are set as follows, $NP = \{70,80,90,100,200\}$, $ds = \{8,10\}$, $alpha = \{0.1,0.2\}$, $T_0 = \{0.2, 0.3, 0.4\}$. Hence, a total of $5*2*2*3 = 60$ parameter configurations are tested. The new instances are generated according to the Section 4.1 and each instance runs 5 times. The total of the experimental results is $5*2*2*3*48*5 = 14,400$. Afterwardss, the results of the additional experiment are also analyzed by the ANOVA technique. The analysis results are exhibited in the interval plot with 95% confidence level in Fig. 9. For the population size, when $NP = 200$, the performance of the proposed algorithm performs well. However, the run time is fairly slow. Therefore, the recommendatory population size is $NP = 100$. In a summary, the best parameter configuration is given as follows. $ds = 8$, $alpha = 0.2$, $T_0 \leq 0.4$.

4.2. Performance analysis for each component

As presented in Section 3.5, the three operators in the CWWO algorithm are refined to enhance the performance of the proposed algorithm. Therefore, there are three dominant strategies in the CWWO, the propagation operation with reinforcement learning (*CWWOnop*), the breaking operation based on path-relinking (*CWWOnob*), and the multi-neighborhood perturbation strategy (*CWWOnor*), respectively. To analyze each component of the CWWO, the simulation experiment results are carried out on 48 instances randomly generated, including $n = \{100,200\}$, $m = \{5,10\}$, $F = \{4,6\}$, $P = \{30,40\}$, each combination has three instances. The processing time is randomly generated from 1 to 99. The stopping termination criterion is set to $T_{max} = n*m*f$ms. The comparison results among the four compared algorithms are shown in Table 2.

From Table 2, the overall average ARPD of the CWWO algorithm is less than the overall average ARPD of the other algorithms and the ARPD values of the CWWO algorithm for each instance are superior to the ARPD values of the three compared algorithms. When the number of machines is 5, the number of factories is 4, the number of productions is 30 and the number of jobs is 100, it's seen that the ARPD of the *CWWOnop* is the largest, which demonstrates that the propagation operation with reinforcement learning (*CWWOnop*) has the greatest impact on the CWWO algorithm. The reason is that when the CWWO algorithm is executed on the small-scale problem, the propagation operation plays a critical role to balance the exploration and exploitation of the CWWO algorithm. When the improvement propagation operation is removed, the loss for the ability of the exploration is leading

Table 2
The comparison results among the CWWO, *WOnop*, *CWWOnob*, and *CWWOnor*.

Category	Number	CWWO	*CWWOnop*	*CWWOnob*	*CWWOnor*
Machine	5	**0.422**	0.838	0.743	0.785
	10	**0.483**	0.745	0.732	0.817
Factory	4	**0.461**	0.872	0.725	0.801
	6	**0.444**	0.711	0.751	0.801
Production	30	**0.323**	0.481	0.426	0.431
	40	**0.599**	1.102	1.049	1.171
Jobs	100	**0.492**	0.962	0.918	0.956
	200	**0.413**	0.621	0.538	0.647
Average		**0.455**	0.791	0.735	0.801

to the CWWO algorithm caught in the local search. When the number of machines is 10, the number of factories is 6, the number of productions is 40 and the number of jobs is 200, the ARPD of the *CWWOnor* is the largest. Therefore, the refraction operation has the greatest influence on the CWWO algorithm. The enhanced refraction operation removed leads to that the CWWO algorithm has the stagnation of the search. On most occasions, the influence of the breaking operation based on path-relinking on the CWWO algorithm is the smallest. According to the above analysis, the influence of the three improved strategies on the CWWO algorithm is different under different circumstances. However, the enhanced refraction operation has the greatest influence on the CWWO algorithm from the entire results.

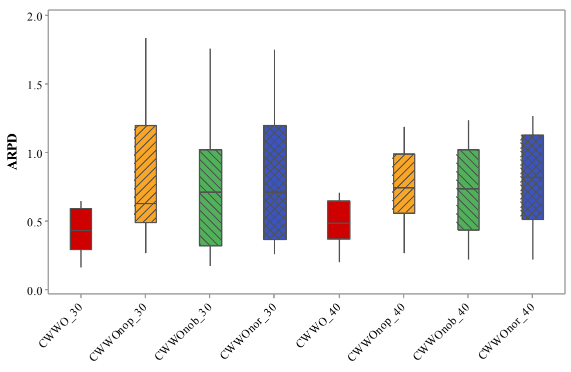
In Fig. 10, the box-plots of the four algorithms are presented under different circumstances. Fig. 10(a) demonstrates that when the number of the productions is different, the impact of the *CWWOnor* on the proposed algorithm is obvious, the *CWWOnob*is secretary and the least impact is *CWWOnop*. In the Fig. 10(b), the influence of the *CWWOnor* on the proposed algorithm is outstanding when the number of factories is 4 and 6. For the number of factories is 4, the impact of *CWWOnob* on the proposed algorithm is secretary, *CWWOnop* is last. However, the influence of the *CWWOnop* on the proposed algorithm is secretary. For Fig. 10(c), when the number of the machine is 5, the three operators have no significant impact on the proposed algorithm. While the number of the machine is 10, the performance of the proposed algorithm is enhanced by the collaboration of the three operators. When the number of jobs is 100 (Fig. 10(d)), the performance of the proposed algorithm is better than the number of jobs is 200, which is declared that the three operators the impact of the three operations on the proposed algorithm is greatest when the number of jobs is 100. Combined with the above analysis and the no free lunch (NFL) (Wolpert, 1997), it's explained that the three operators are the three operators have different effects on the proposed algorithm in different situations, and to a certain extent, the performance of the proposed algorithm is improved by the three refined operators.

The pie chart for the three components is shown in Fig. 11. According to Fig. 11, it's clearly seen that the improved refraction operation has a significant impact on the CWWO algorithm.

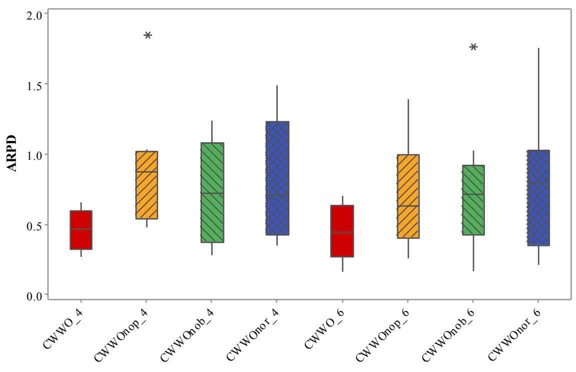
Moreover, the interval plot of interaction among the CWWO algorithm, *CWWOnop*, *CWWOnob* and *CWWOnor*with 95% confidence interval is shown in Fig. 12. It's seen that the impact of the three operations on the performance of the proposed algorithm is comparable. Meanwhile, the influence of improved refraction operation is distinct. Combined with Table 2, Figs. 11 and 12, the proposed algorithm is to effectively balance the exploration and exploitation, which elucidates the effectiveness of all employed improvement strategies, and shows that the effect of the three strategies of cooperation is ″$1 + 1 + 1 > 3$″.

4.3. The comparative experimental analysis

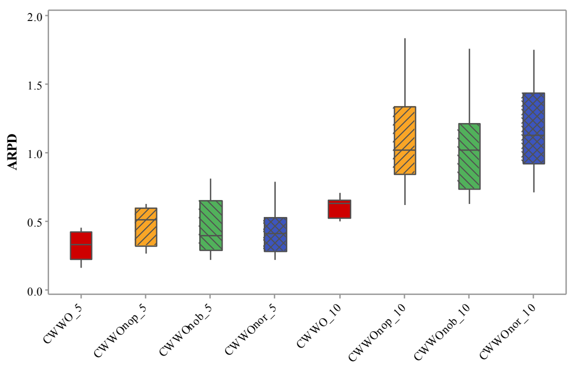
In this paper, the computational experiments are executed with a large benchmark set proposed for the DAPFSP (Hatami et al., 2013). The benchmark set included the number of jobs $n = \{100, 200, 500\}$, the number of machines $m = \{5,10,20\}$, the number of factories $F = \{4,6,8\}$, the number of products $P = \{30,40,50\}$. There are 10 test instances for each combination and each combination runs 10 times. The proposed algorithm is compared with four state-of-the-art algorithms, including EDAMA (Wang & Wang, 2015), HBBO (Lin & Zhang, 2016), HILS (Shao, Dechang, et al., 2019), and HVNS (Shao, Dechang, et al., 2019). The parameters setting is strictly following all original details given in the original paper to closely reach published performance and the method of the makespan is calculated by Shao, Dechang, et al. (2019). The ARPD values of all compared algorithms grouped according to machines, factories, products, and jobs are listed in Table 3. The interval interaction plots for all compared algorithms with different conditions are given in Fig. 13. Furthermore, the best, mean, worst and standard deviation values of the proposed algorithm are placed in the supplementary



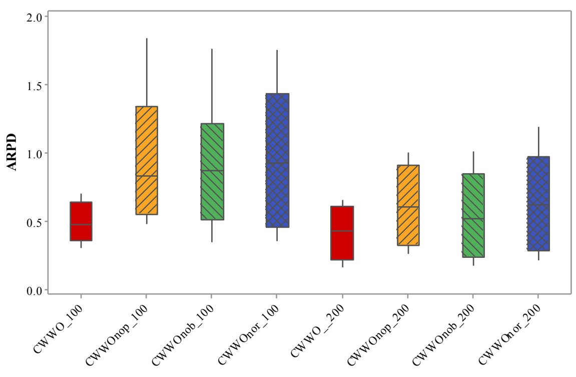
(a) The box-plot for the different number of productions



(b) The box-plot for the different number of factories



(c) The box-plot for the different number of machines



(d) The box-plot for the different number of jobs

Fig. 10. The box-plots for the CWWO, *CWWOnop*, *CWWOnob*, and *CWWOnor*.

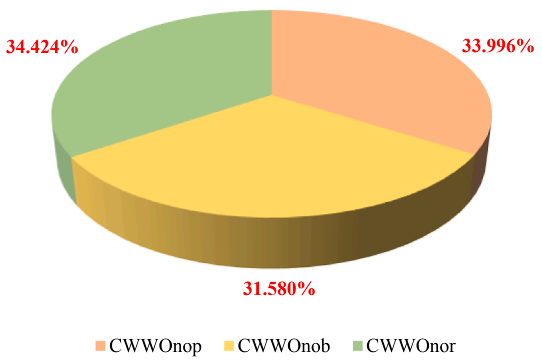


Fig. 11. The pie chart for the three components.

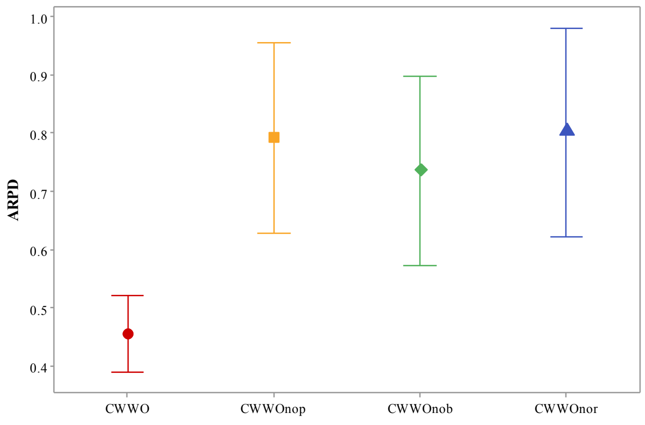


Fig. 12. The interval plot of interaction among the CWWO, *CWWOnop*, *CWWOnob*, and *CWWOnor*.

materials.

From Table 3, the concrete analyses have been listed as follows.

(1) The average ARPD of the proposed algorithm is 0.617, which is smaller than the average APRD of the other algorithms. However, when the number of the machine is 5, the performance of the HILS is better than the proposed algorithm and the other compared algorithms.
(2) When the $m = 10$ and $m = 20$, the ARPD of the proposed algorithm is gradually increasing and smaller than the ARPD of the other compared algorithms. For the number of productions, when $P = 40$, the performance of the proposed algorithm is satisfactory.
(3) With increasing for the number of factories and jobs, the ARPD of the proposed algorithm is successively decreasing, which explains that the more the number of factories and jobs, the more stable the compared algorithms. Obviously, the performance of the proposed algorithm precedes other compared algorithms.

In accordance with Fig. 13, the influences of machines, jobs, productions, and factories for the proposed algorithm are further analyzed.

(1) As seen from Fig. 13(a), the interval plot of interaction for the different number of productions with a 95% confidence interval

Table 3
The ARPD values of all compared algorithms.

Category	Number	CWWO	HBBO	EDAMA	HILS	HVNS
Machine	5	0.396	1.436	1.139	**0.328**	0.609
	10	**0.624**	1.598	1.367	0.801	0.891
	20	**0.845**	1.565	1.301	1.563	1.345
Factory	4	**0.763**	2.009	1.622	1.106	1.154
	6	**0.578**	1.461	1.201	0.861	0.903
	8	**0.525**	1.127	0.985	0.724	0.788
Production	30	**0.646**	1.730	1.372	0.877	1.065
	40	**0.609**	1.546	1.278	0.908	0.950
	50	**0.611**	1.321	1.158	0.906	0.830
Job	100	**0.786**	1.084	0.989	1.208	1.050
	200	**0.586**	1.217	1.030	0.816	0.875
	500	**0.430**	2.297	1.789	0.667	0.921
Average		**0.617**	1.533	1.269	0.897	0.948

is to explain the significant difference among all compared algorithms, which is illustrated that the stability of the proposed algorithm is superior to the other algorithm.

(2) According to Fig. 13(b), the performance of the proposed algorithm is desirable in all compared algorithms under the condition of the different number of factories.

(3) Fig. 13(c) presents that the more the number of the machines, the greater the significant difference among all compared algorithms.

(4) The performance of the proposed algorithm is stability with the increasing number of jobs from Fig. 13(d).

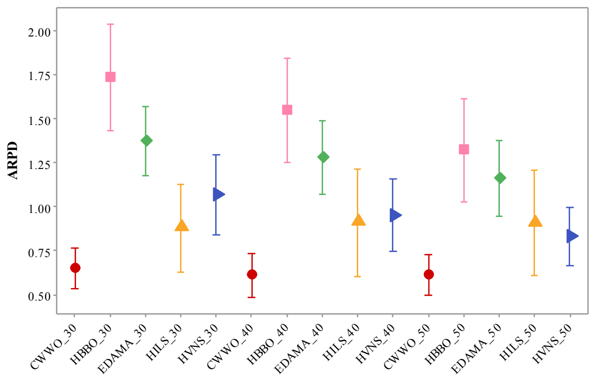
Finally, Fig. 14 illustrates that the interval interaction plot of the proposed algorithm is under the other compared algorithms with a 95% confidence interval, which states that the CWWO algorithm outperformed the four compared algorithms.

Meanwhile, the standard deviation (SD) as an evaluation criteria is employed to assess the performance of the CWWO algorithm. The SD is a statistical indicator to reflect the dispersion degree of a set of data. The smaller the SD, the more clustered the data. The SD values of all compared algorithms are presented in Table 4. As shown in Table 4, the average SD value of the CWWO algorithm is 29.654, which is the smallest. However, when the number of the machine is 5, the SD value of the CWWO algorithm is larger than the SD value of the HILS algorithm, which illustrated that the performance of the HILS algorithm is better than the performance of the CWWO algorithm. For the other occasions, the SD value of the CWWO algorithm are smaller than the compared algorithms, which declared that the stability of the CWWO algorithm is better than the four algorithms.

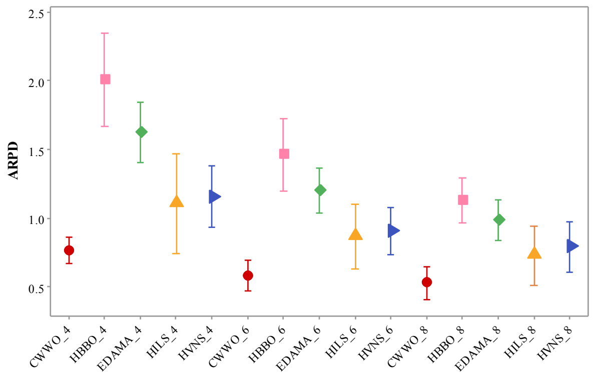
Afterwards, the interval plot with the SD of all compared algorithms is shown in Fig. 15. The interval plot of the CWWO algorithm is under the interval plot of the four algorithms with a 95% confidence interval. Although the stability and interval difference of the CWWO algorithm are superior to the HILS algorithm, the interval plot of the CWWO algorithm has an interaction with the interval plot of the HILS algorithm, which is illustrated that the CWWO algorithm has no significant difference with the HILS algorithm. However, the CWWO algorithm has significant difference with the other three compared algorithms. To sum up, the stability and effectiveness of the CWWO algorithm outperformed the four compared algorithms.

4.4. *The statistical analysis for all compared algorithms*

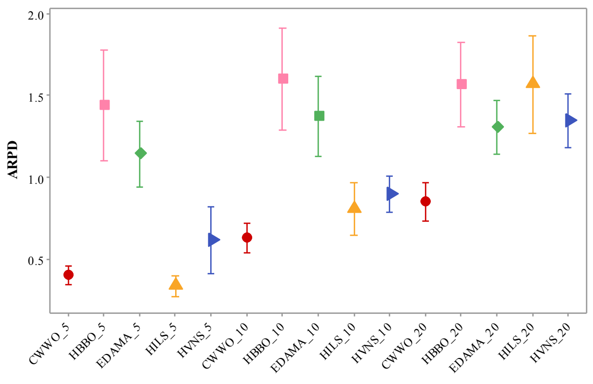
The statistical test plays a vital role in the experimental analysis. In



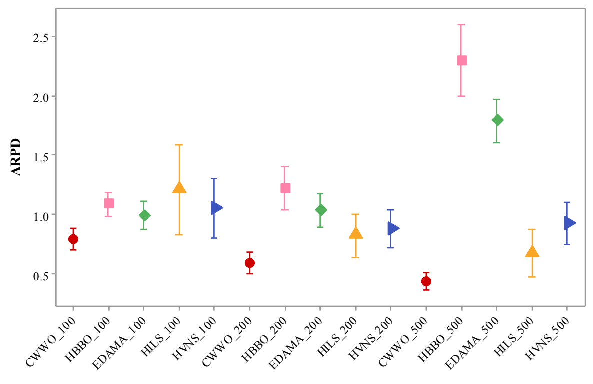
(a) The interval plot of different productions.



(b) The interval plot of different factories.



(c) The interval plot of different machines.



(d) The interval plot of different jobs

Fig. 13. The plots of the different conditions for all compared algorithms.

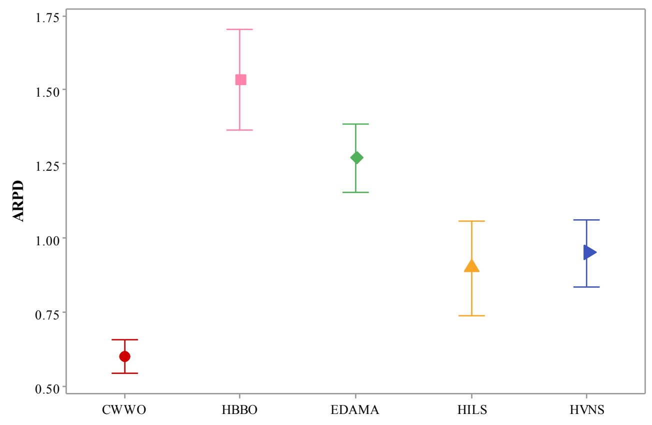


Fig. 14. The interval plot of all compared algorithms with the APRD.

Table 4
The standard deviation values of the compared algorithms.

Category	Number	CWWO	HBBO	EDAMA	HILS	HVNS
Machine	5	17.637	102.325	77.837	**16.928**	44.969
	10	**28.081**	114.748	97.897	35.312	42.897
	20	**43.245**	117.818	96.334	82.581	70.869
Factory	4	**40.040**	148.473	115.609	48.202	68.676
	6	**26.550**	107.840	86.018	43.288	47.031
	8	**22.373**	78.579	70.441	43.331	43.027
Production	30	**32.439**	126.749	96.637	44.372	59.585
	40	**28.917**	111.692	90.868	43.111	52.970
	50	**27.607**	96.451	84.563	47.338	46.180
Job	100	**19.602**	26.336	24.071	24.406	20.471
	200	**23.325**	55.605	47.267	37.783	39.557
	500	**46.036**	252.950	200.730	72.633	98.707
Average		**29.654**	111.631	90.689	44.940	52.912

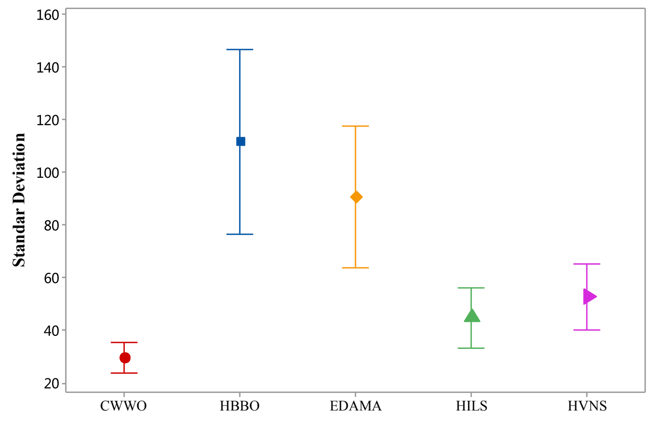


Fig. 15. The Interval plot of all compared algorithms with the SD.

Table 5
The results obtained by the Wilcoxon's test in different significance levels.

CWWO vs.	$R+$	$R-$	Z	$p-value$	$\alpha =$ 0.05	$\alpha =$ 0.01
HBBO	304929.00	23526.00	−21.123	4.8673E-99	**Yes**	**Yes**
EDAMA	297684.00	30711.00	−20.036	2.6993E-89	**Yes**	**Yes**
HILS	233747.00	93898.00	−10.517	7.208E-26	**Yes**	**Yes**
HVNS	266926.00	61529.00	−15.418	1.2407E-53	**Yes**	**Yes**

Table 6
The mean ranks for the compared algorithms obtained by the Friedman'test.

Algorithms	Mean Rank	Chi-Square	*p*-Value
CWWO	1.950	738.358	1.7222E-158
HBBO	3.880		
EDAMA	3.520		
HILS	2.670		
HVNS	2.950		
Crit.Diff.	0.202		
Crit.Diff.	0.243		

this section, the statistical analysis is carried out to verify the results of all compared algorithms with a 95% confidence and detect the significant difference among the compared algorithms. Therefore, the Wilcoxon's test (García et al., 2008) is adopted to detect the validity of the compared algorithms and the statistical results are presented in Table 5. The CWWO algorithm is regarded as the control algorithm. According to the Table 5, the $R+$ represents that the CWWO algorithm outperformed the four compared algorithms. "Yes" indicates that significant difference is existed between the control algorithm and the compare algorithm. The statistical test results obtained by the Wilcoxon's test with $\alpha = 0.05$ and $\alpha = 0.01$ illustrates that the CWWO algorithm has more significant advantage than the four compared algorithms.

Meanwhile, the Friedman's test (García et al., 2008) is further applied to verify significant difference among all compared algorithms. The mean ranks of all compared algorithms obtained by the Friedman's test are demonstrated in Table 6. According the statistical test results in Table 6, the $p-value$ is 1.7222E−158 and the $\chi^2(2)$ is 738.358 with $\alpha =$ 0.05 and $\alpha = 0.01$. The mean rank of the CWWO algorithm is the smallest in the five compared algorithms. Moreover, the additional Bonferroni-Dunn's method is serviced as a post hoc procedure to calculate the critical difference (CD computed according to Eq. (26)) and evaluate the significance level for all compared algorithms. In Eq. (26), k is the number of algorithms used to compared and N is the number of instances. In this experimental procedure, k is set to 5 and N is set to 810. According to Table B.16 (two tailed $\alpha(2)$) of Zar (2010), when $\alpha = 0.01$, $q_a = 3.091$ and $\alpha = 0.05$, $q_a = 2.576$.

$$CD = q_a\sqrt{\frac{k(k+1)}{6N}} \tag{26}$$

The results of the Bonferroni-Dunn's method are shown in Fig. 16 and the CWWO algorithm is considered as the control algorithm. It's obvious that the CWWO algorithm has significant difference with the compare algorithms. From the above analysis, the CWWO algorithm is significantly superior to the four compared algorithms.

According to the above experimental discussion and statistical test

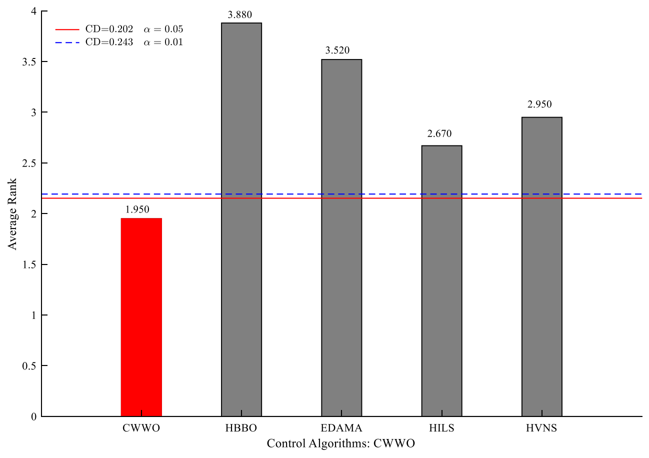


Fig. 16. Graphical representation of the Bonferroni-Dunn's test.

analysis, the conclusions are expressed as follows. (1) the ability of the exploitation of the CWWO algorithm is effectively enhance by the different strategies in terms of the components analysis. (2) the improved propagation operation has an advantage to balance the exploration and exploitation according the reward and experience of the wavelength. According to the experimental analysis of the three components, the neighborhood structure information in the breaking stage has been not taken advantage led to that the performance of PR is not fully developed. (3) in the refraction stage, the knowledge stored in the historical best information has been taken full advantage to escape the local optimal. However, the running time of the CWWO algorithm is longer than the HILS algorithm and HVNS algorithm. In general, the proposed algorithm is an effective method to address the DANIFSP with the objective of minimizing the maximum assembly completion time.

4.5. *Algorithm complexity analysis*

In this section, the time complexity of the proposed algorithm is analyzed. The proposed algorithm, which is a meta-heuristic belong to an iteration approach, has a requirement to take time to obtain a near optimal solution. If the time complexity of distributing one job is $O(1)$, the time complexity of problem, which has n jobs, is $O(n)$ (Jia et al., 2018). In this study, the proposed algorithm consists of four portions, including the initialization allocation and three main operations. For the initialization phase, the time complexity of distributing jobs into each factory is $O(n^2f)$, where f is the number of factories. For the propagation operator, the time complexity of modified propagation operation is $O(n^3f)$. The time complexity for the breaking operation is $O(NP)$. In the refraction phase, the time complexity is $O(n^*f)$. Therefore, the whole-time complexity of the CWWO algorithm is shown as follows.

$$O(n,f,p,k,NP) = O(k)^*\left[O\left(n^2f\right)+O\left(n^3f\right)+O(NP)+O(n^*f)\right]^*O(p)$$

$$\approx O(k)^*O\left(n^3f\right)^*O(p)$$

$$\approx O\left(kn^3fp\right)$$

where k is the iterations and p means the number of production.

5. Conclusion

This study proposed a collaborative water wave optimization algorithm (CWWO) based on the three-stage variable neighborhood search with minimizing the maximum assembly time as the optimization goal to address the distributed assembly no-idle flow-shop scheduling problem (DANIFSP). The conclusions about the paper are shown as follows.

(1) The $Q-$*learning* controlled by the wavelength, as a perturbation method based on the framework of the VNS, is introduced in the propagation operation to generate the new solution to balance the exploration and exploitation of the CWWO algorithm. In the breaking phase, the path-relinking mechanism assisted by the VNS is designed to improve the local search ability of the proposed algorithm. In the refraction stage, a multi-neighborhood perturbation strategy is performed on the current optimal solution obtained by the knowledge from the historical information to help the CWWO algorithm jump out the local optimal and keep the vitality of searching.
(2) For the experimental analysis, the relevant parameters in the proposed algorithm are analyzed by the DOE method and ANOVA method. The optimum parameter combination is given to help the proposed algorithm obtain the optimal. Afterwards, the effectiveness of the three major operations in the CWWO algorithm is verified to illustrate the influence of the operation collaboration. The analyses of each component indicated that the modified strategies have great impact on the performance of the proposed algorithm.
(3) The Wilcoxon's test and Friedman's test are adopted to declare the significant difference among all compared algorithms. Furthermore, the experimental results confirmed that the effectiveness and stability of the CWWO algorithm outperformed the other compared algorithms referred to this paper.

Considering the future research, the additional constraint conditions, such as sequence-dependent setup time, worker allocation, green energy resource, and multi-task allocation are added to the distributed flow-shop problem with assembly procedure. For the solving method, the prior knowledge information and data hidden in the algorithm are fully utilized to improve information utilization and operating efficiency. Moreover, the essence of the distributed flow-shop problem and algorithms used are key for addressing the problem.

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.

Acknowledgement

This work was financially supported by the National Natural Science Foundation of China under grant numbers 62063021. It was also supported by the Key Research Programs of Science and Technology Commission Foundation of Gansu Province (2017GS10817), Lanzhou Science Bureau project (2018-rc-98), Public Welfare Project of Zhejiang Natural Science Foundation (LGJ19E050001), respectively.

Appendix A. Supplementary material

Supplementary data to this article can be found online at https://doi.org/10.1016/j.cie.2020.107082.

References

Afifi, A.A. & Azen, S. (1972). The analysis of variance.
Campbell, H. G., Dudek, R. A., & Smith, M. L. (1970). A heuristic algorithm for the n job, m machine sequencing problem. *Management Science, 16*(10), 630–630.
Chen, J.-F., Wang, L., & Peng, Z.-P. (2019). A collaborative optimization algorithm for energy-efficient multi-objective distributed no-idle flow-shop scheduling. *Swarm and Evolutionary Computation, 50*.
Civicioglu, P. (2013). Backtracking search optimization algorithm for numerical optimization problems. *Applied Mathematics and Computation, 219*(15), 8121–8144.
Ferone, D., Hatami, S., González-Neira, E. M., Juan, A. A., & Festa, P. (2020). A biased-randomized iterated local search for the distributed assembly permutation flow-shop problem. *International Transactions in Operational Research, 27*(3), 1368–1391.
García, S., Molina, D., Lozano, M., & Herrera, F. (2008). 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.
Glover, F. (1997). Tabu search and adaptive memory programming—Advances, applications and challenges. In R. S. Barr, R. V. Helgason, & J. L. Kennington (Eds.), *Interfaces in computer science and operations research: Advances in metaheuristics, optimization, and stochastic modeling technologies* (pp. 1–75). Boston, MA: Springer, US.
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.
Hatami, S., Ruiz, R., & Andrés-Romano, C. (2013). The distributed assembly permutation flowshop scheduling problem. *International Journal of Production Research, 51*(17), 5292–5308.
Hatami, S., Ruiz, R., & Andrés-Romano, C. (2015). Heuristics and metaheuristics for the distributed assembly permutation flowshop scheduling problem with sequence dependent setup times. *International Journal of Production Economics, 169*, 76–88.
Jia, Y.-H., Chen, W.-N., Yuan, H., Tianlong, G., Zhang, H., Gao, Y., & Zhang, J. (2018). An intelligent cloud workflow scheduling system with time estimation and adaptive ant colony optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 1–16.
Jones, R. (2010). Design and Analysis of Experiments (fifth edition). Quality & Reliability Engineering International, 18(2), 163–163.

Kaveh, A. & Bakhshpoori, T. (2019). Artificial Bee Colony Algorithm.
Koulamas, C., & Kyparisis, J. G. (2001). The three-stage assembly flowshop scheduling problem. *Computers & Operations Research, 28*(7), 689–704.
Lin, J., Wang, Z.-J., & Li, X. (2017). A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem. *Swarm and Evolutionary Computation, 36*, 124–135.
Lin, J., & Zhang, S. (2016). An effective hybrid biogeography-based optimization algorithm for the distributed assembly permutation flow-shop scheduling problem. *Computers & Industrial Engineering, 97*, 128–136.
Liu, A., Peng, L., Weiliang, S., Deng, X., Li, W., Zhao, Y., & Liu, B. (2019). Prediction of mechanical properties of micro-alloyed steels via neural networks learned by water wave optimization. *Neural Computing and Applications*.
Meng, T., Pan, Q. K., & Wang, L. (2019). A distributed permutation flowshop scheduling problem with the customer order constraint. *Knowledge-Based Systems, 184*, 17.
Naderi, B., & Azab, A. (2015). An improved model and novel simulated annealing for distributed job shop problems. *International Journal of Advanced Manufacturing Technology, 81*(1–4), 693–703.
Naderi, B., & Ruiz, R. (2010). The distributed permutation flowshop scheduling problem. *Computers & Operations Research, 37*(4), 754–768.
Nawaz, M., Enscore, E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega, 11*, 91–95.
Nussenbaum, K., & Hartley, C. A. (2019). Reinforcement learning across development: What insights can we draw from a decade of research? *Developmental Cognitive Neuroscience, 40*.
Pan, Q. K., Gao, L., Li, X. Y., & Jose, F. M. (2019). Effective constructive heuristics and meta-heuristics for the distributed assembly permutation flowshop scheduling problem. *Applied Soft Computing, 81*, 16.
Pan, Q. K., Tasgetiren, M. F., & Liang, Y. C. (2008). A discrete differential evolution algorithm for the permutation flowshop scheduling problem. *Computers & Industrial Engineering, 55*(4), 795–816.
Potts, C. N., Sevastjanov, S. V., Strusevich, V. A., Vanwassenhove, L. N., & Zwaneveld, C. M. (1995). The 2-stage assembly scheduling problem: Complexity and approximation. *Operations Research, 43*(2), 346–355.
Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: A gravitational search algorithm. *Information Sciences, 179*(13), 2232–2248.
Sang, H.-Y., Pan, Q.-K., Wang, P., Han, Y.-Y., Gao, K.-Z., & Duan, P. (2018). Effective invasive weed optimization algorithms for distributed assembly permutation flowshop problem with total flowtime criterion. *Swarm and Evolutionary Computation, 44*.
Shao, W., Dechang, P., & Zhongshi, S. (2019). Local search methods for a distributed assembly no-idle flow shop scheduling problem. *IEEE Systems Journal*, 1–12.
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*.
Shao, Z., Pi, D., & Shao, W. (2019). A novel multi-objective discrete water wave optimization for solving multi-objective blocking flow-shop scheduling problem. *Knowledge-Based Systems, 165*, 110–131.
Simon, D. (2008). Biogeography-based optimization. *Ieee Transactions on Evolutionary Computation, 12*(6), 702–713.
Tasgetiren, M. F., Pan, Q. K., Suganthan, P. N., & Buyukdagli, O. (2013). A variable iterated greedy algorithm with differential evolution for the no-idle permutation flowshop scheduling problem. *Computers & Operations Research, 40*(7), 1729–1743.
Wang, S.-Y., & Wang, L. (2015). An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 46*, 1–1.
Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning, 8*(3–4), 279–292.
Wolpert, D. (1997). Macready: No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation - TEC*.
Wu, X. B., Liao, J., & Wang, Z. C. (2015). Water wave optimization for the traveling salesman problem. *International conference on intelligent computing*.
Zar, J. (2010). *Biostatistical analysis*. New Jersey: Prentice-Hall.
Zhang, J. Z., Zhou, Y. Q., & Luo, Q. F. (2019). Nature-inspired approach: A wind-driven water wave optimization algorithm. *Applied Intelligence, 49*(1), 233–252.
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*.
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.
Zhao, F., Qin, S., Zhang, Y., Ma, W., Zhang, C., & Song, H. (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., 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., Xue, F., Zhang, Y., Ma, W., Zhang, C., & Song, H. (2019). A discrete gravitational search algorithm for the blocking flow shop problem with total flow time minimization. *Applied Intelligence, 49*(9), 3362–3382.
Zhao, F., Zhang, L., Liu, H., Zhang, Y., Ma, W., Zhang, C., & Song, H. (2019). An improved water wave optimization algorithm with the single wave mechanism for the no-wait flow-shop scheduling problem. *Engineering Optimization, 51*(10), 1727–1742.
Zhao, F., Zhang, L., Zhang, Y., Ma, W., Zhang, C., & Song, H. (2020). A hybrid discrete water wave optimization algorithm for the no-idle flowshop scheduling problem with total tardiness criterion. *Expert Systems with Applications, 146*, 113–166.
Zheng, Y. J. (2015). Water wave optimization: A new nature-inspired metaheuristic. *Computers & Operations Research, 55*, 1–11.
Zheng, Y. J., Lu, X. Q., Du, Y. C., Xue, Y., & Sheng, W. G. (2019). Water wave optimization for combinatorial optimization: Design strategies and applications. *Applied Soft Computing, 83*, 16.