A Wale Optimization Algorithm for Distributed Flow Shop with Batch Delivery

Qinghua Li
Shandong Normal University
Junqing Li (lijunqing@lcu-cs.com)
Shandong Normal University https://orcid.org/0000-0002-3617-6708
Xinjie Zhang
Shandong Normal University
Biao Zhang
Liaocheng University

Research Article

Keywords: distributed flow shop, batch delivery, wale optimization algorithm, local search.

Posted Date: July 27th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-741608/v1

Version of Record: A version of this preprint was published at Soft Computing on August 21st, 2021. See the published version at https://doi.org/10.1007/s00500-021-06099-0.

A Wale Optimization Algorithm for Distributed Flow Shop with Batch Delivery

Abstract: In this study, a distributed flow shop scheduling problem with batch delivery constraints is investigated. The objective is to minimize the makespan and energy consumptions simultaneously. To this end, a hybrid algorithm combining the wale optimization algorithm (WOA) with local search heuristics is developed. In the proposed algorithm, each solution is represented by three vectors, namely a job scheduling sequence vector, batch assignment vector, and a factory assignment vector. Then, an efficient neighborhood structure is applied in the proposed algorithm to enhance search abilities. Furthermore, the simulated annealing algorithm and clustering method are embedded to improve the global search abilities of the algorithm. Finally, 30 instances are generated based on realistic application to test the performance of the algorithm. After detailed comparisons with three efficient algorithms, i.e., ABC-Y, ICA-K, and $IWOA_{NS}$, the superiority of the proposed algorithm is verified.

Key words: distributed flow shop; batch delivery; wale optimization algorithm; local search.

1 Introduction

With the rapid development of manufacturing, many enterprises began to consider multiple factories working at the same time which formed distributed scheduling (DS). In DS, the distributed flow shop scheduling problem (DFSP) is the most widely studied [1-13], where many types of constraints have been researched, such as sequence-dependent setup times [1,8], no-wait [2,12], no-idle [4], stochastic time [6], release time [10], and random machines breakdowns [13]. In addition, many types of objectives, such as makespan, total cost, and average tardiness [3], total tardiness [5], earliness/tardiness [11], are also minimized. These studies considered different types of constraints and objectives in DFSP, which can be closer to the reality. However, most jobs should be assembled into different groups of products, and deliver to different companies, where the assembly constrains should be taken as a challenging factor.

Nowadays, batch delivery has been used in various fields [14-16]. Wang et al. [17] considered order selection and assignment in the distributed problems. Yin et al. [18] integrated production and batch delivery scheduling that processed and delivered jobs together in batches. Qi et al. [19] studied a two-agent scheduling problem with batch delivery. Basir et al. [20] presented a batch delivery system on a two-stage assembly flowshop. Noroozi et al. [21] considered a third-party logistics distribution, where production scheduling and batch delivery were combined. Jiang et al [22] studied the scheduling problem to deliver the products to the customers in batches. Kong et al. [23] proposed a Just-in-Time strategy to precast construction in a batch delivery problem. Kazemi et al. [24] considered batching delivery with assembly flow shop scheduling. The batch delivery procedures have also been researched by Agnetis et al. [25] and Wang et al. [26]. However, less literature has considered DFSP with batch delivery constraints.

Recent years, many types of meta-heuristics have been developed for solving different optimization problems [27-45]. Basir et al. [27] presented a bi-level improved genetic algorithm to solve the two-stage assembly flow shop scheduling problem with batch delivery system. Peng et al. [29] developed an improved artificial bee colony algorithm for a steelmaking casting process. Liao et al. [31] introduced a particle swarm optimization algorithm for hybrid flow shops. Several meta-heuristics have also been developed for the permutation flow shop scheduling problem, such as a population-based tabu search [34], a hybrid whale algorithm [43]. For the distributed permutation flow shop scheduling problem, Gao et al. [32] developed an efficient tabu search algorithm. In addition, the whale optimization algorithm (WOA), as an efficient swarm intelligent algorithm, has also been applied for many optimization problems [41-45].

Based on the above discussed optimization problems and meta-heuristics, we develop a hybrid algorithm combining the wale optimization algorithm (WOA) with local search heuristics to solve the distributed flow shop scheduling problem with batch delivery constraints (DFSP-BD). The main contributions are as follows: (1) a hybrid algorithm combining the wale optimization algorithm (WOA) with local search heuristics is developed; (2) each solution is represented by two vectors, namely a job scheduling sequence vector, and a two-dimensional vector to record the factory assignment, and product assignment, respectively; (3) an efficient neighborhood structure is applied in the proposed algorithm to enhance search abilities; and (4) a simulated annealing algorithm and clustering method are embedded, to improve the global search abilities of the algorithm.

The remainder of this paper is organized as follows. Section 2 gives the problem description. Section 3 introduces the related algorithms. Section 4 describes the proposed algorithm with all of the components. The computational results and comparisons are reported in Section 5. Finally, the last section presents the concluding remarks and future research directions.

2 Problem description

The DFSP-BD is a typical realistic optimization problem, which combines DFSP and batch delivery to customer process. Therefore, two charging tasks should be solved, i.e., schedule jobs in the processing stage, and assign jobs in the batch delivery stage. Fig. 1 shows a realistic example for the considered problem.

In processing stage, first, a set of jobs $\{J_1, J_2, J_3 ..., J_n\}$ are randomly assigned to a set of factories $\{F_1, F_2, F_3 ..., F_f\}$. Then, at each factory, the assigned jobs will be processed through the same machine sequence. In the batch delivery stage, each job is assigned to a certain batch, where all the jobs in the same batch can be delivered to certain customers. The assumptions are described as follows:

- All machines and jobs are ready at 0 time;

- Each machine can process only one job at a time;
- Each job can be processed at exactly one machine at a time;
- Processing overlap is not permitted, i.e., all operations belonging to the same jobs should be processed one by one.
- Each job should be assigned to exactly one batch.
- All jobs belonging to the same batch should be delivered at the same time.
- Each job should be assigned to exactly one factory.

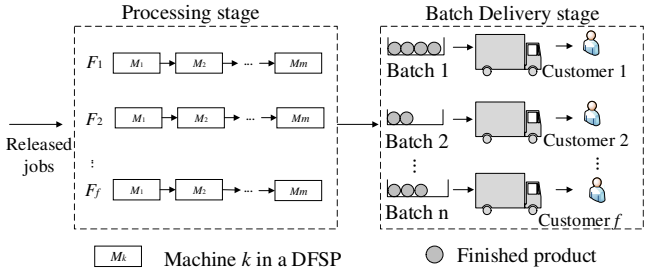

Fig.1 Illustration of a realistic DFSP-BD problem.

2.1 Problem formulation

The notations and decision variables are given in Table 1.

Table 1 Notations and decision variables

Index	
j	job index, j=1, 2, …, n
i	machine index, i=1, 2…, m
f	factory index, f=1, 2, …, F
l, s	batch type
v	speed, v=1, 2, …, S
n	the number of jobs
m	the number of machines
F	the number of factories
p	the number of the batch
S	the number of speeds
pt_s	standard batch delivery time of batch
t_{ji}	standard process time of jobs
M	a large number
$ppf_{i,v}$	The *EC* per unit time of machine i in factory f running at speed v
spf_i	The *EC* of the machine i at stand-by mode per unit time in factory f
PPP	Unit operation energy consumption of machine in batch delivery stage
SPP	Standby energy consumption of machine in batch delivery stage
$X_{k,j,f}$	In the factory f, job j is processed immediately after job k
$Y_{j,f}$	The job j is in factory f
$c_{j,i,f}$	The completion time of job i on machine j in the factory f
$Z_{l,s}$	Binary value set to l, if l to be delivery just before each product s
$H_{j,i,v}$	Binary value set to 1, if the processing speed of job j is v on machine m
$Fsv_{s,v}$	Binary value set to 1, if the processing speed of product s on the machine is v
$p_{j,i}$	The actual processing time
pp_s	The actual batch delivery time
c_f	The end time of factory f
CA_s	The total completion times of processing and batch delivery
$c_{\max}$	The maximum completion times
PEC	The processing energy consumption
SEC	The standby energy consumption
EC	The total energy consumption

Minimize:

$$w*C_{\max}+(1-w)*EC \quad (1)$$

Subject to:

$$\sum_{v=1}^{S} H_{j,i,v}=1 \quad (2)$$

$$\sum_{v=1}^{S} F_{s,v}=1 \quad (3)$$

$$p_{j,i}=t_{j,i}*\sum_{v=1}^{S} H_{j,i,v}/v \quad (4)$$

$$pp_s=pt_s*\sum_{v=1}^{S} F_{s,v}/v \quad (5)$$

$$\sum_{f=1}^{F} X_{k,j,f}=1, k=1,...n, k\neq j \quad (6)$$

$$X_{j,j,f}=0 \quad (7)$$

$$\sum_{k=1}^{n} X_{k,j,f}+X_{j,k,f}\le 2*Y_{j,f} \quad (8)$$

$$\sum_{f=1}^{F} X_{k,j,f}+X_{j,k,f}\le 1 \quad (9)$$

$$\sum_{f=1}^{F}\sum_{j=1}^{n} X_{k,j,f}\le 1 \quad (10)$$

$$\sum_{j=1}^{n} Y_{j,f}\ge 1 \quad (11)$$

$$\sum_{f=1}^{F} Y_{j,f}=1 \quad (12)$$

$$\sum_{j=1}^{n} X_{0,j,f}=1 \quad (13)$$

$$c_{j,1,f}\ge p_{j,1}-M*(1-Y_{j,f}) \quad (14)$$

$$c_{j,i,f}\ge c_{j,i-1,f}+p_{j,i}-M*(1-Y_{j,f}) \quad (15)$$

$$c_{j,i,f}\ge c_{j,i,f}+p_{j,i}+(X_{k,j,f}-1)*M \quad (16)$$

$$c_f\ge c_{j,m,f}-M*(1-Y_{j,f}) \quad (17)$$

$$\sum_{l=1}^{p} Z_{l,s}=1 \quad (18)$$

$$\sum_{s=1}^{p} Z_{l,s}\le 1 \quad (19)$$

$$Z_{l,s}+Z_{s,l}\le 1 \quad (20)$$

$$CA_s\ge c_{j,m,f}+(G_{j,s}-1)*M+pp_s+(Y_{j,f}-1)*M \quad (21)$$

$$CA_s\ge CA_l+pp_s+M*(Z_{l,s}-1) \quad (22)$$

$$c_{j,i,f}\ge 0 \quad (23)$$

$$PEC=\sum_{j=1}^{n}\sum_{i=1}^{m}\sum_{v=1}^{S} H_{j,i,v}*P_{j,i}*ppf_{i,v}+\sum_{s=1}^{p}\sum_{v=1}^{S} pp_s*Fsv_{s,v}*ppp_v \quad (24)$$

$$SEC=\sum_{f=1}^{F}\sum_{i=1}^{m}(c_f-\sum_{j=1}^{n} p_{j,i}*Y_{j,f})*spf_i+(c_{\max}-CA_1-\sum_{s=2}^{p} pp_s) \quad (25)$$

$$EC = PEC + SEC \tag{26}$$

The objective function (1) is to minimize the weighted sum of makespan and total energy consumption. Constraint (2) indicates that each job has a unique processing speed on each machine. Constraint (3) indicates that the processing speed of each product s on the machine is unique. Constraint (4) calculates the processing speed of each job on each machine. Constraint (5) calculates the processing speed of each batch on each machine. Constraints (6) - (10) restrict that two jobs belonging to the same factory should have a unique processing sequence relationship. The constraint (11) means that each factory allocates at least one job. Constraint (12) indicates that any job can only be assigned to one factory. Constraint (13) restricts that each factory has at least one job. Constraint (14) calculates the makespan of the first operation of each job. Constraint (15) indicates that there is no overlap between processes. Constraint (16) indicates that the processing between two adjacent jobs in each factory is not allowed to overlap. Constraint (17) calculates the makespan of each factory. Constraint (18) ensures that there is only one previous product (l) to be delivered just before each product s. Constraint (19) ensures that no more than one product (s) can be processed after each product. Constraint (20) checks that a product to be delivered cannot be both a predecessor and successor of another product to be delivered at the same time. Constraint (21) indicates that every product s will not start the batch delivery stage until all work on the last machine (M) has been completed. Constraint (22) restricts no overlap between the batches, determines that if the batch s is to be delivery immediately after the batch l, the operation of the batch l must end before the operation of the batch s begins. Constraint (23) limits the range of decision variables. Constraints (24-26) calculate the total energy consumption.

2.3 Problem illustration

Given a simple DFSP-BD problem, there are 2 factories, 7 jobs, and 2 machines in each factory. All the jobs should be delivered to 2 customers with 3 batches. Table 2 gives the processing times and batch deliver times, and Fig. 2 shows the resulted Gantt chart for the example.

Table 2. Processing times for the example.

Customer	Job	Processing time		Batch delivery time
		M_1	M_2	
C_1	J_1	4	6	5
	J_2	2	6	
	J_7	3	5	4
	J_6	5	2	
C_2	J_4	5	3	6
	J_3	6	4	
	J_5	4	2	

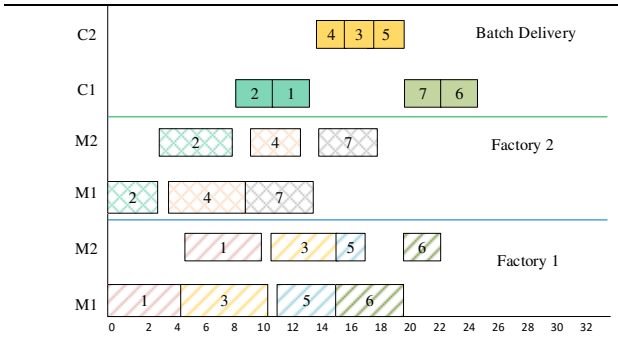

Fig. 2 Gantt chart for the example.

It can be seen from Fig. 2 that: (1) four jobs including J_1, J_3, J_5, and J_6 are processed in the first factory, while the following three jobs, i.e., J_2, J_4, and J_7, have been assigned to the second factory; and (2) all the jobs belonging to the same batch should be assembled into a batch to deliver to the given customer. For example, J_1 and J_2 are assigned to batch 1 to dispatch to customer 1.

3 The canonical WOA

The WOA algorithm, proposed by Mirjalili and Lewis [35], is inspired by the process of whales to prey food. In the canonical WOA, two typical procedures including bubbling and encircling are embedded to perform the searching tasks.

3.1 Framework of the canonical WOA

The framework of WOA is described in Algorithm 1.

Algorithm 1 The whale optimization algorithm (WOA)
Input: a population
Output: the best solution

```
initial Population X_i (i =1, 2…, n)
compute the search agent fitness value
X* = search agent of the lowest fitness value
While (t < the max iterations)
    for each search agent
        if ( p < 0.5 ) then
            if (|A|<1 ) then
                X_{t+1} = X*_t − A·D
            else if (|A|≥1 ) then
                X_{t+1} = X*_rand − A·D
            end
        else if ( p ≥ 0.5 ) then
            X_{t+1} = D'·e^{bl}·cos(2πl)+X*_t
        end
    end
    Repair the search agents
    compute the fitness of each search agent and update X*
end
```

3.2 Bubbling and encircling procedure

Whales swim around their prey and update the location of the search agent according to the best location of the search agent. Encircling the prey mechanism can be defined as follows:

$$D = | K \cdot X^*_t - X_t | \tag{27}$$

$$X_{t+1} = X^*_t - A \cdot D \tag{28}$$

$$A = 2a \cdot r - a \tag{29}$$

$$K = 2 \cdot r \tag{30}$$

where t is the current iteration number, X^*_t is the current best solution, X_t is the position vector and K is a coefficient. D is a distance ranging between X^*_t and X_t. A is randomly selected between $[-a, a]$, and r is a random number from $[0,1]$.

The whale attacks the prey by spiral upgrading way and the process can be defined as follows:

$$X_{t+1} = D' \cdot e^{bl} \cdot cos(2\pi l) + X^*_t \tag{31}$$

$$D' = | X^*_t - X_t | \tag{32}$$

where D' is a distance value between X^*_t and X_t, b is a

constant to define the shape of the logarithmic spiral, l is a value range between $[-1,1]$.

The probability of encircling the prey and spiral bubble-net attacking are 50%, respectively. The model is as follows:

$$X_{t+1} = \begin{cases} X^*_t - A \cdot D & if \ \ p < 0.5 \\ D' \cdot e^{bl} \cdot cos(2\pi l) + X^*_t & if \ \ p \geq 0.5 \end{cases} \quad (33)$$

3.3 Exploration phase

The exploration phase can be defined as:

$$D = | K \cdot X^*_{rand} - X_t | \quad (34)$$

$$X_{t+1} = X^*_{rand} - A \cdot D \quad (35)$$

where X^*_{rand} is a random whale individual which is selected from the current population.

4 The proposed algorithm

4.1 Solution representation

In DFSP-BD, we used three vectors to represent each solution, which is shown in Fig.3. The first vector, named factory assignment vector, assigns each job to a certain factory. The second vector, named scheduling vector, arranges the processing order of the jobs in the assigned factory. The last vector assembles several different jobs into the given batch. As shown in Fig.3, different colors represent different products. J_1 and J_2 belong to P_2, J_3 and J_4 belong to P_1, the rest of jobs belong to the P_3. Four jobs, i.e., J_1, J_3, J_5, *and* J_6 are processing in F_1, and three jobs, i.e., J_2, J_4, *and* J_7 are processing in F_2.

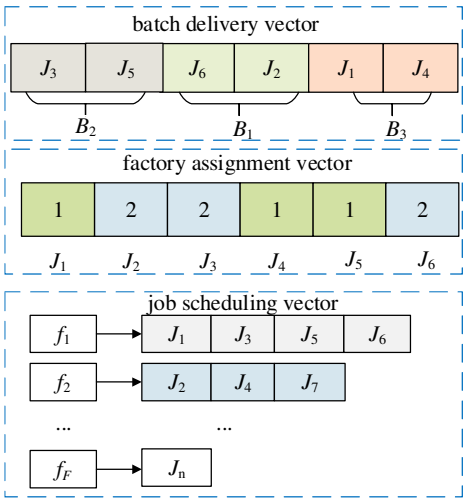

Fig. 3 Solution representation in DFSP-BD

4.2 Neighborhood structures

To balance the global and local search abilities, four types of neighborhood structures are developed. The neighborhood structure is given in Algorithm 2.

4.2.1 Swap different products in a batch

This method aims to swap different products in a randomly selected batch. The detailed steps are as follows: (1) first, randomly select a batch and two products (e.g., P_5 and P_7); and (2) select better positions for the selected products with the minimum completion time. Fig. 4 shows an example to swap two products in a selected batch.

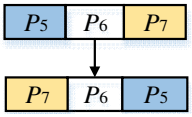

Fig. 4 Swap different products in a batch

4.2.2 Swap different jobs in a factory

This method aims to swap different jobs in a randomly selected factory. The detailed steps are as follows: (1) select the factory with the maximum completion time as the critical factory; (2) and randomly select two jobs in the selected factory; and (3) swap the two selected jobs and update the current solution if the newly-generated solution is better. Fig. 6 shows the swap procedure of this approach.

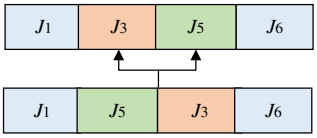

Fig. 5 Swap different jobs in a factory

Algorithm 2 Local search strategy
Input: two position *A* and *B*
Output: the best sequence *S*

```
The initialized number R, R = rand () %4
case 0: Job insertion approach
    Set job scheduling in each factory f1, f2 and f3
    if jobs scheduling in each factory >1 do
        Insert jobs of f1 into f2 and f3
    end if
break
case 1: Swap different products in a batch
    Randomly select a batch scheduling b1
    if b1>1 do
        looks for two good insert position, and swap the two
        products in b1
    end if
break
case 2:  Swap different jobs in a factory
    if f1>1 do
        Step1. randomly select a factory f1, and selected two
        positions p1 and p2 from the job scheduling
        Step2. swap the two jobs in the two positions
    end if
break
case 3:  Swap different jobs in different factories
    if f1 and f2>1 do
        Step1. randomly select two factory f1 and f2, selected two
        positions p1 and p2 from the job scheduling
        Step2. swap the two jobs in the two positions
    end if
break
```

4.2.3 Swap different jobs in different factories

This method aims to swap different jobs in different factories. The detailed steps are as follows: (1) randomly select two jobs from two different factories; and (2) swap the two selected jobs and update the current solution if the newly-generated solution is better. Fig. 7 shows the procedure of this approach.

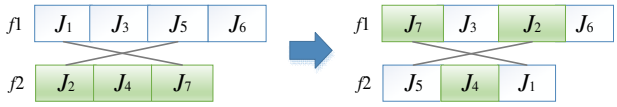

Fig. 6 Swap different jobs in different factories

4.2.4 Job insertion approach

The job insertion approach aims to delete jobs from the critical factory and insert them into other factories. The

detailed steps are as follows: (1) firstly, select a factory with the maximum completion time as the critical factory; (2) secondly, insert all jobs of the critical factory into other factories; and (3) during the insertion of deleted jobs to other factories, the newly-generated solutions are evaluated and the best one will be selected to update the current individual.

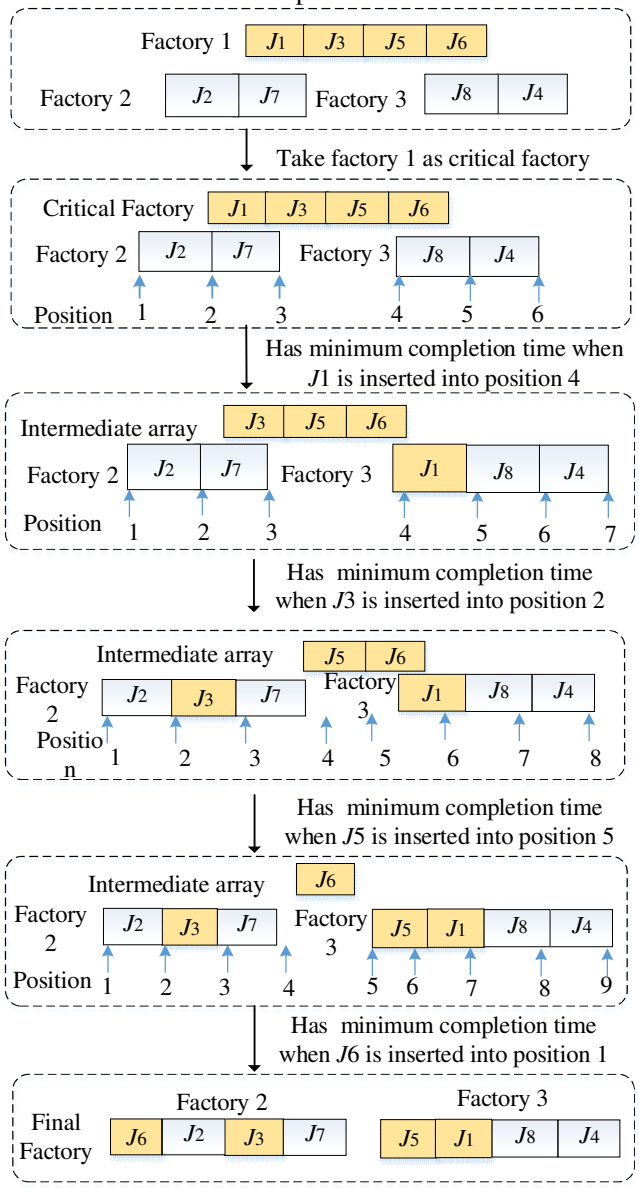

Fig. 7 Insert jobs in factories

4.3 K-means discretization exploitation phase

The *K-means* discretization method is an iterative clustering analysis algorithm and it is applied to divide the population into several sub-populations. In exploitation phase, bubble-net attacking of the whale as a cluster of sub-populations, which can find locally optimal solution in each cluster. The main steps of *K-means* discretization method are as follows:

Step 1. Divide the data into K groups, and K center individuals are randomly selected as the initial clustering centers.

Step 2. Calculate the distance between each individual and each seed clustering center, and each individual is assigned to the nearest clustering center. Cluster centers and the individuals assigned to them represent a cluster. The distance is calculated by $D_i = \sqrt{(S_i - C_i)^2}$, Where S_i is a dimension of solution and C_i is a dimension of center, D_i is the distance between S_i and C_i .

Step 3. Each cluster center of the cluster will be recalculated according to the existing individuals in the cluster.

Step 4. The above process will be repeated until a termination condition is met.

4.4 SA-based local search acceptance criterion

The SA-based acceptance criterion is embedded to enhance the global search abilities of the proposed algorithm. The detailed steps are given as follows:

Step 1. Compared the values of the neighboring solution X_n and the best solution found so far X_{best} , if $E(X_n) < E(X_{best})$, replace X_{best} with X_n directly.

Step 2. if $E(X_n) > E(X_{best})$, accept X_n with probability of $P(x_{best}, x_n, T) = e^{E(x_{best})-E(x_n)/T}$, where T stands for temperature, which determines the probability of acceptance. The calculation process of T is shown in formula (36), and the parameter decreases continuously during the iteration.

$$Temperature = T \cdot \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}}{n \cdot m \cdot 10} \tag{36}$$

4.5 Framework of the proposed algorithm

The framework of the proposed algorithm is described in Algorithm 3.

Algorithm 3 The improved whale optimization algorithm (IWOA$_{NS}$)
Input: an initialized population P
Output: the best solution S

```
The initialized population F is divided into k subpopulations by K-means algorithm i.e. P= {p1,p2,…,pk} (c.f. subsection 4.3)
Calculated the fitness value i for each subpopulation pi
While the termination condition is not satisfied
    if ( p < 0.5 ) then
        Perform the WOA with encircling the prey stage (c.f. subsection 3.1)
        if (| A|< 1 ) then
            Perform the SA-based local search (c.f. subsection 4.4)
        else if (| A|≥ 1 ) then
            Perform the WOA with searching for the prey (c.f. subsection 3.2)
        end
    else if ( p ≥ 0.5 ) then
        Perform the WOA with spiral bubble-net attacking stage (c.f. subsection 3.1)
        Local search is performed and SA is embedded (c.f. subsection 4.3)
    end
end
Perform the SA-based local search acceptance criterion (c.f. subsection 4.5)
end
Output the best solution found so far
```

5 Numerical experiments

5.1. Experimental instances

To test the efficiency of the IWOA$_{NS}$ in solving DFSP-BT, we generate 30 different scales of instances, where in the number of jobs is {20,50,80,100,200}, the number of machines is {2,5,8}, and the number of factories is {2, 5}, respectively.

5.2. Effectiveness of CPLEX

For evaluating the performance of $IWOA_{NS}$, the exact solver IBM ILOG CPLEX 12.7.1 is used to calculate the MIP model. The comparison experiment results are shown in Table 3. For the first column is problem scale which contains the number of jobs, machines and factories. The best value of each instance is illustrated in second column, the next two columns are minimum fitness value of $IWOA_{NS}$ and CPLEX, and the last two columns are percentage deviation difference obtained by each algorithm with respect to the corresponding optimal value, the calculation formula is given at (37):

$$dev = (f_c - f_b) / f_b \times 100\% \qquad (37)$$

where f_c represents the best solution generated by $IWOA_{NS}$ or CPLEX; f_b represents the best solution between $IWOA_{NS}$ and CPLEX.

From the Table 3, it can be observed that: (1) under 6 instances, $IWOA_{NS}$ obtains a higher solution quality, while CPLEX has worse performance; and (2) compared with the last row, the average of $IWOA_{NS}$ is less than CPLEX.

Table 3 Comparison between CPLEX and $IWOA_{NS}$

Ins	Best	Min fitness		dev	
		$IWOA_{NS}$	CPLEX	$IWOA_{NS}$	CPLEX
ins_5_5_2	294.81	294.81	299.60	**0.00**	0.016
ins_5_5_3	221.00	221.00	246.20	**0.00**	0.014
ins_6_5_2	309.19	309.19	310.60	**0.00**	0.005
ins_5_5_3	292.86	292.86	315.40	**0.00**	0.076
ins_8_4_2	320.47	320.47	335.60	**0.00**	0.047
ins_8_4_3	298.23	298.23	301.20	**0.00**	0.009
avg	289.43	289.43	301.43	**0.00**	0.028

5.3. Sensitivity analysis

In this subsection, to analysis the sensitivity, we set two experiments: performance changes under different parameters and performance of the algorithm at different scales. Details are shown below.

5.3.1 Performance changes under different parameters

In this paper, there are two parameters, namely, the temperature (T), and population group numbers (N). We set five levels of two parameters to obtain experimental data, and analyze for a better group of parameters.

As shown in Table 4, the different values of the two parameters are combined. Thus, the influence of these two parameters on the performance of the algorithm is analyzed using DOE's Taguchi method. An orthogonal array L16 (42) is used to analyze the parameters at four-factor levels. Each parameter group is run independently 30 times to get the average value, which is the response variable (RV). The RV for the 16 groups of parameters is listed in Table 4, and the parameters are analyzed by the line graph in Fig.8. From Table 5 and line graph Fig.8, when the parameter T is at the first level and N is at the first level, the result is the best, and the parameters T and N were set to 0.1 and 0.9, respectively.

Table 4 The parameter values.

Parameter	Values			
	1	2	3	4
T	0.1	0.3	0.5	0.7
N	3	6	9	10

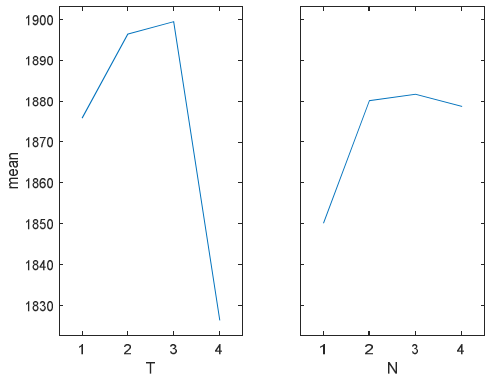

Fig.8 The factor level trend of the two parameters.

Table 5 Orthogonal array and RV values.

Experiment number	Factor		RV
	T	N	
1	0.1	3	1822.76
2	0.1	6	1892.69
3	0.1	9	1893.34
4	0.1	10	1894.67
5	0.3	3	1892.02
6	0.3	6	1891.24
7	0.3	9	1908.07
8	0.3	10	1894.51
9	0.5	3	1900.18
10	0.5	6	1899.58
11	0.5	9	1898.97
12	0.5	10	1899.16
13	0.7	3	1827.29
14	0.7	6	1825.13
15	0.7	9	1826.56
16	0.7	10	1826.73

5.3.2 Performance of the algorithm at different scales

In this section, in order to show clearly and directly the superiority of the $IWOA_{NS}$ in solving DFSP-BD, we construct experiments to compare the three algorithms on DFSP-BD.

To verify the performance of $IWOA_{NS}$ in solving DFSP with batch delivery constraints, the other two typical algorithms were compared with the $IWOA_{NS}$: ABC algorithm (ABC-Y) was proposed by Yurtkuran in 2018; ICA algorithm (ICA-K) was proposed by Kazemi in 2017, respectively. The main reasons for selecting these compare algorithms are as follows: (1) for the $IWOA_{NS}$ algorithm, first, in the initialization part of the $IWOA_{NS}$ algorithm, neighborhood structures are proposed, and a local search strategy based on whale swarm optimization algorithm is used to enhance the local search abilities; then, the convergence of the algorithm is analyzed, and it is proved that the algorithm has convergence both locally and globally; (2) for the ICA-K algorithm, which include initialization, assimilation, revolution, and colony exchange phases. In ICA-K, one or more colonies of the weakest empires are generally moved to other empires through the empire competition mechanism, which has robust local search ability and faster convergence speed, but it is easier to fall into the local optimization than the WOA; (3) for the ABC-Y algorithm, it can find high-quality honey source with high efficiency in any environment, and also can adapt to the change of environment, there are three main parts of the ABC-Y algorithm: the onlooker bee, the employed bee and the scout bee, where the employed bees and onlookers are used to perform exploitation tasks and the scout bees are designed for performing exploration tasks.

The results are recorded in Table 6 which show that the performance of all three contrast algorithms on scale 20 is far

superior to other scales. Moreover, except for the scale 20, the algorithm outperforms the other comparative algorithms on any other scale.

5.4. Effectiveness of Neighborhood structures

For test the efficiency of the IWOA with local search strategy ($IWOA_{NS}$), we compared $IWOA_{NS}$ and the IWOA without local search strategy (IWOA) in Table 7. Through two algorithms, 30 instances are run independently for 30 times, and the maximum, minimum and average values are obtained. And then obtained the best value and the dev.

As shown in Table 7, the first column is problem scale which contains the number of jobs, machines and factories. The best value of each instance is presented in second column, the next two columns are minimum fitness value of two algorithms, and the last two columns are the values of percentage deviation obtained by each algorithm with respect to the corresponding optimal value, the calculation formula is given at (37).

Table 7 Comparisons of $IWOA_{NS}$ and IWOA

Ins	Best	Min fitness		dev	
		$IWOA_{NS}$	IWOA	$IWOA_{NS}$	IWOA
20-2-2	432.54	432.54	544.54	**0.00**	20.57
20-2-5	507.62	507.62	662.04	**0.00**	23.32
20-2-8	613.81	613.81	763.42	**0.00**	19.60
20-5-2	369.47	369.47	369.49	**0.00**	0.01
20-5-5	467.78	467.78	467.84	**0.00**	0.01
20-5-8	569.82	569.82	589.89	**0.00**	3.40
50-2-2	1014.74	1014.74	1336.34	**0.00**	24.07
50-2-5	1043.07	1043.07	1434.31	**0.00**	27.28
50-2-8	1172.63	1172.63	1655.01	**0.00**	29.15
50-5-2	849.88	849.88	942.69	**0.00**	9.85
50-5-5	987.63	987.63	1114.89	**0.00**	11.41
50-5-8	1061.40	1061.40	1253.54	**0.00**	15.33
80-2-2	1674.59	1674.59	2306.60	**0.00**	27.40
80-2-5	1651.03	1651.03	2531.86	**0.00**	34.79
80-2-8	1755.69	1755.69	2443.77	**0.00**	28.16
80-5-2	1334.64	1334.64	1601.08	**0.00**	16.64
80-5-5	1465.17	1465.17	1666.84	**0.00**	12.10
80-5-8	1531.57	1531.57	1920.47	**0.00**	20.25
100-2-2	2097.55	2097.55	2808.03	**0.00**	25.30
100-2-5	2091.65	2091.65	2994.88	**0.00**	30.16
100-2-8	2180.19	2180.19	3229.81	**0.00**	32.50
100-5-2	1631.66	1631.66	1910.11	**0.00**	14.58
100-5-5	1750.76	1750.76	2009.31	**0.00**	12.87
100-5-8	1809.80	1809.80	2199.73	**0.00**	17.73
200-2-2	4439.25	4439.25	6032.87	**0.00**	26.42
200-2-5	4251.25	4251.25	6216.12	**0.00**	31.61
200-2-8	4272.08	4272.08	6072.29	**0.00**	29.65
200-5-2	3385.87	3385.87	4409.08	**0.00**	23.21
200-5-5	3331.87	3331.87	4315.98	**0.00**	22.80
200-5-8	3461.11	3461.11	4496.65	**0.00**	23.03
avg	1773.54	1773.54	2343.32	**0.00**	20.44

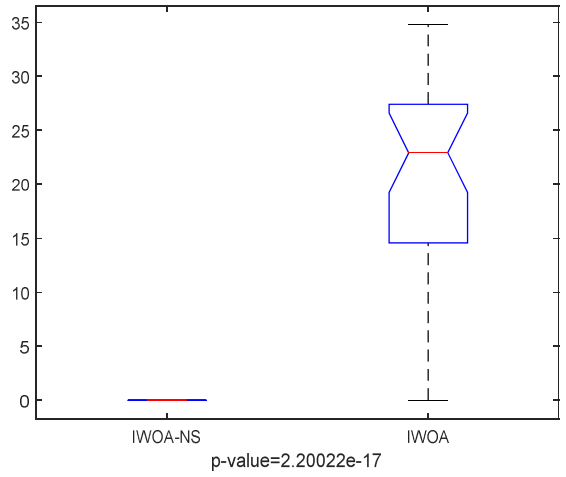

Fig.9 Means and 95% LSD interval for compared $IWOA_{NS}$ and IWOA

It can be seen from Table 7: (1) for 30 instances, $IWOA_{NS}$ have 30 best solutions, while IWOA only have 3 best solutions; and (2) compared with the last row, the average of $IWOA_{NS}$ is far less than IWOA.

To verify the performance with compared IWOA, an ANOVA chart is generated. Fig.9 presents compared result with dev values of the two compared algorithms. The average dev value of 30 instances is calculated. Under the 95% confidence interval, if p-value <0.05, the performance of the algorithm is significantly better than other algorithms, according to the multi-factor analysis of variance (ANOVA) theory. Through the Fig.6 can be seen, the p-value is 2.20022e-17 which is far less than 0.05, showing the proposed algorithm has better performance.

5.5. Comparisons with other efficient algorithms

The experimental results are shown in Table 8. The first column represents the scale of 30 instances. The second column is the optimal value of each example run; the next three columns give the best solution of each algorithm after 30 independent experiments. The last three columns are the deviations calculated by each algorithm. It can be seen from Table 8 that: (1) Among the 30 instances, the $IWOA_{NS}$ algorithm has 28 optimal solution, while the ABC-Y algorithm has only 8 optimal solutions and the ICA-K algorithm has only 1 optimal solution. (2) Compared with the last line, the mean values of $IWOA_{NS}$ and ICA-K are close to each other, and both are far less than the ABC-Y algorithm.

In order to further prove the superiority of the $IWOA_{NS}$, Fig.10 illustrates the compared result of ABC-Y algorithm and $IWOA_{NS}$. From the p-value=1.07668e-06 which is far less than 0.05, showing the proposed algorithm has better performance.

Four instances were randomly selected to further analyze the performance of $IWOA_{NS}$ and ICA-K. And the convergence curve which drawn based on the experimental data as shown in Fig. 11. The results show that $IWOA_{NS}$ is far superior to ICA-K. As shown in these convergence curves, the $IWOA_{NS}$ shows better convergence abilities for the considered the DFSP with batch delivery constraint.

As shown in Fig.12, the Gantt chart contains two customers and four batches, and has 20 jobs, 8 machines, 5 factories. Each rectangle corresponds to a job, and the color represent different batches.

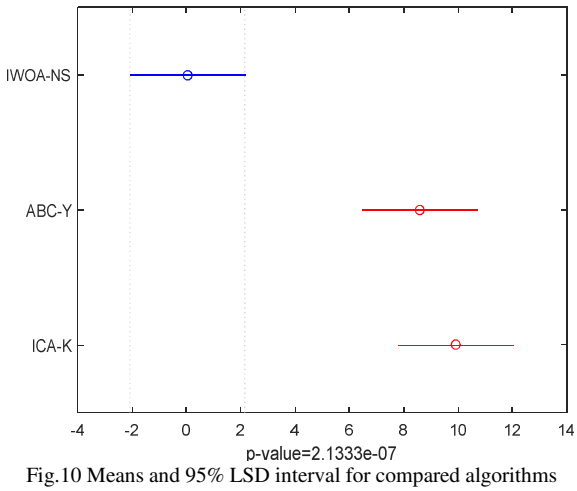

Fig.10 Means and 95% LSD interval for compared algorithms

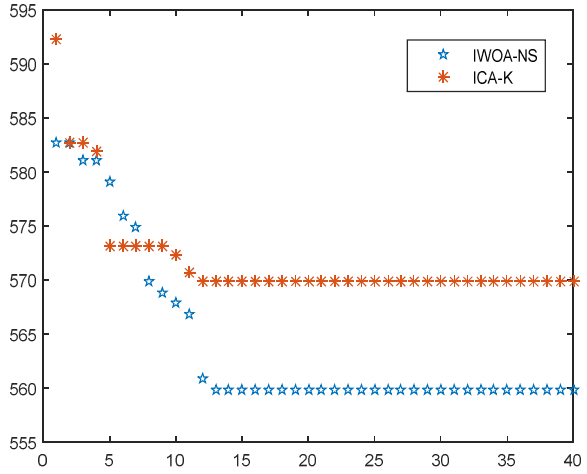

Fig.11 (a). 20-5-8 convergence

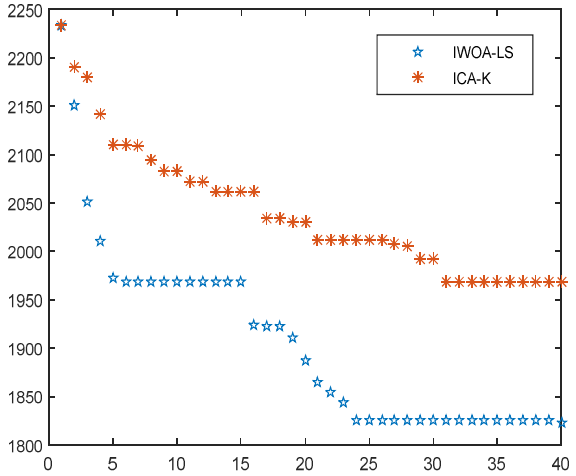

Fig.11 (b). 100-2-8 convergence

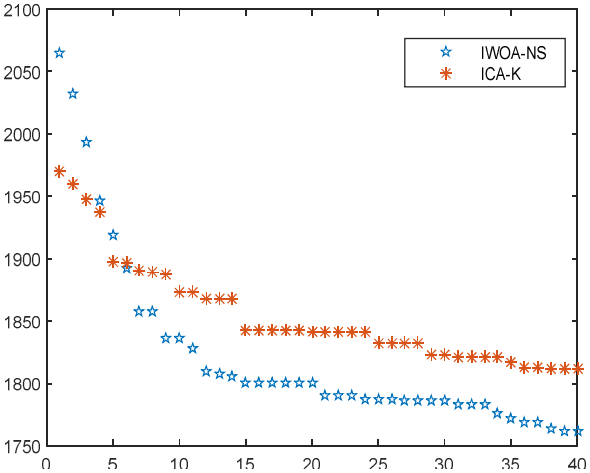

Fig.11(c). 100-5-5 convergence

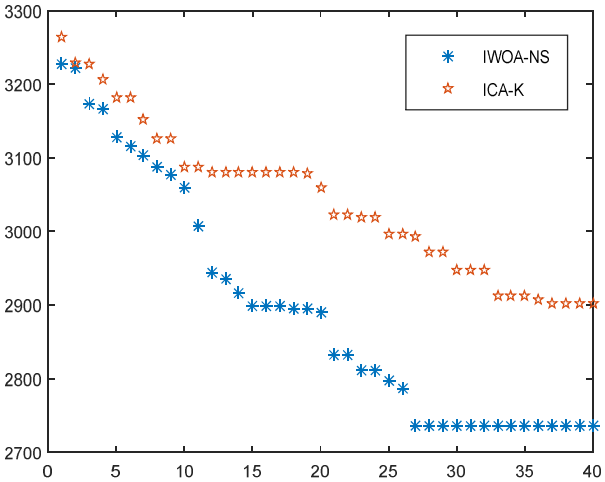

Fig.11 (d). 200-5-2 convergence

Table 6 Comparisons of Performance of algorithms at different scales

Scales	Best	Min fitness			dev		
		$IWOA_{NS}$	ABC-Y	ICA-K	$IWOA_{NS}$	ABC-Y	ICA-K
20	493.51	493.51	492.71	496.05	0.15	**0.00**	0.48
50	1021.56	1021.56	1083.84	1106.80	**0.00**	5.22	6.87
80	1568.78	1568.78	1748.02	1794.54	**0.00**	9.11	11.62
100	1926.94	1926.94	2192.32	2248.08	**0.00**	10.65	13.07
200	3856.91	3856.91	4758.81	4812.30	**0.00**	18.02	19.06
avg	1773.38	1773.54	2055.14	2091.554	0.03	8.6	10.22

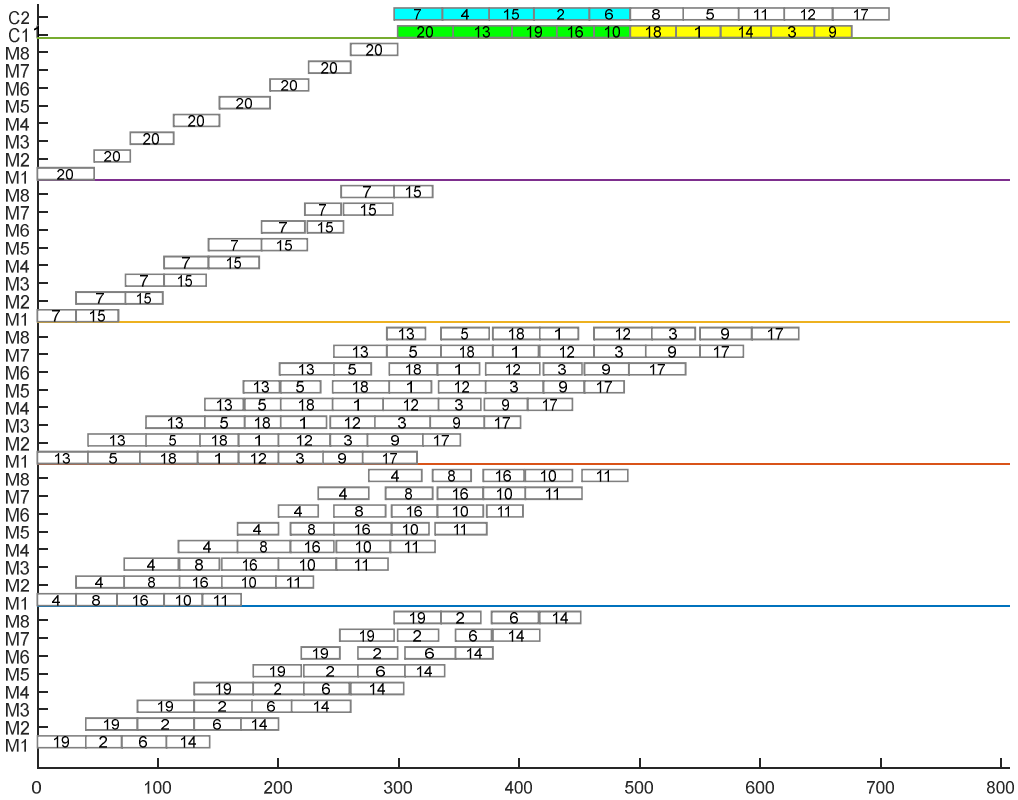

Fig.12. Gantt chart for the best solution of the instance 20-8-5-2

Table 8 Comparisons of $IWOA_{NS}$ and other algorithms

Ins	Best	Min fitness			dev		
		$IWOA_{NS}$	ABC-Y	ICA-K	$IWOA_{NS}$	ABC-	ICA-K
20-2-2	432.54	432.54	432.55	434.95	**0.00**	**0.00**	0.55
20-2-5	507.62	507.62	504.43	514.84	0.63	**0.00**	1.40
20-2-8	613.81	613.81	612.19	619.41	0.26	**0.00**	0.90
20-5-2	369.47	369.47	369.47	369.48	**0.00**	**0.00**	**0.00**
20-5-5	467.78	467.78	467.79	467.78	**0.00**	**0.00**	**0.00**
20-5-8	569.82	569.82	569.85	569.83	**0.00**	0.01	**0.00**
50-2-2	1014.74	1014.74	1168.34	1205.14	**0.00**	13.15	15.80
50-2-5	1043.07	1043.07	1160.68	1209.50	**0.00**	10.13	13.76
50-2-8	1172.63	1172.63	1275.05	1327.08	**0.00**	8.03	11.64
50-5-2	849.88	849.88	849.89	849.89	**0.00**	**0.00**	**0.00**
50-5-5	987.63	987.63	987.65	987.66	**0.00**	**0.00**	**0.00**
50-5-8	1061.40	1061.40	1061.44	1061.50	**0.00**	**0.00**	0.01
80-2-2	1674.59	1674.59	2021.82	2017.82	**0.00**	17.17	17.01
80-2-5	1651.03	1651.03	2043.86	2098.29	**0.00**	19.22	21.32
80-2-8	1755.69	1755.69	2014.91	2083.69	**0.00**	12.87	15.74
80-5-2	1334.64	1334.64	1394.65	1442.66	**0.00**	4.30	7.49
80-5-5	1465.17	1465.17	1474.81	1522.02	**0.00**	0.65	3.74
80-5-8	1531.57	1531.57	1538.04	1602.78	**0.00**	0.42	4.44
100-2-2	2097.55	2097.55	2533.61	2594.44	**0.00**	17.21	19.15
100-2-5	2091.65	2091.65	2598.88	2652.45	**0.00**	19.52	21.14
100-2-8	2180.19	2180.19	2680.16	2703.52	**0.00**	18.65	19.36
100-5-2	1631.66	1631.66	1711.68	1751.67	**0.00**	4.67	6.85
100-5-5	1750.76	1750.76	1792.45	1864.44	**0.00**	2.33	6.10
100-5-8	1809.80	1809.80	1837.14	1921.98	**0.00**	1.49	5.84
200-2-2	4439.25	4439.25	5747.23	5662.46	**0.00**	22.76	21.60
200-2-5	4251.25	4251.25	5612.87	5738.51	**0.00**	24.26	25.92
200-2-8	4272.08	4272.08	5496.89	5551.39	**0.00**	22.28	23.04
200-5-2	3385.87	3385.87	3980.29	4043.49	**0.00**	14.93	16.26
200-5-5	3331.87	3331.87	3894.35	3902.32	**0.00**	14.44	14.62
200-5-8	3461.11	3461.11	3821.21	3975.61	**0.00**	9.42	12.94
avg	1773.38	1773.54	2055.14	2091.55	0.03	8.60	0.55

6 Conclusion

In this study, a distributed flowshop scheduling problem with batch delivery constraint is solved. The problem can be applied to the field of garment processing. The processed products are delivered to customers in batches according to customer needs. One-stop service not only shortens the time of garment processing, but also improves the efficiency. To solve this problem, a whale swarm optimization algorithm is employed. Moreover, the algorithm can be used in a wide range of fields, such as distributed scheduling, flexible job shop, distribution network, power systems, etc. In this research, a wale optimization algorithm (IWOA) combined with neighborhood structure is utilized to solve the problem. Then, the local search strategy is applied in the proposed algorithm to enhance search ability. Furthermore, the SA and clustering method are embedded, to improve the performance of the algorithm. Finally, comparisons algorithms with ABC-Y and ICA-K, $IWOA_{NS}$ has the best performance.

In the future, next works are mainly developed as follows: (1) considering distribute flow shop problem with fuzzy constraint; (2) studying more accurate energy consumption in batch delivery; (3) combining the proposed algorithm with other kinds of problems, such as the parallel machine scheduling problem; and (4) proposing better optimization algorithms or more strategies to solve the current problem.

Acknowledgements This research is partially supported by major basic research projects in Shandong (ZR2018ZB0419), and a Grant of Key Laboratory of Intelligent Optimization and Control with Big Data.

Compliance with ethical standards

Conflict of interest the authors declare that they have no conflict of interest.

Human and animal rights this article does not contain any studies with human participants or animals performed by any of the authors.

References

[1] Hatami S, Ruiz R, Andrs-Romano C. Heuristics and metaheuristics for the distributed assembly permutation flowshop scheduling problem with s uence dependent setup times. International Journal of Production Economics, 2015, 169: 76-88.

[2] Ji M, Yang Y, Duan W, et al. Scheduling of no-wait stochastic distributed assembly flowshop by hybrid PSO. IEEE Congress on Evolutionary Computation (CEC). IEEE, 2016: 2649-2654.

[3] Rifai A P, Nguyen H T, Dawal S Z M. Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling. Applied Soft Computing, 2016, 40: 42-57.

[4] Ying K C, Lin S W, Cheng C Y, et al. Iterated reference greedy algorithm for solving distributed no-idle permutation flowshop scheduling problems. Computers Industrial Engineering, 2017, 110: 413-423.

[5] Deng J, Wang L. A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem. Swarm and evolutionary computation, 2017, 32:121-131.

[6] Gonzalez-Neira E M, Ferone D, Hatami S, et al. A biased-randomized simheuristic for the distributed assembly permutation flowshop problem with stochastic processing times. Simulation Modelling Practice and Theory, 2017, 79: 23-36.

[7] Bargaoui, Hafewa, Olfa Belkahla Driss, and Khaled Ghdira. A novel chemical reaction optimization for the distributed permutation flowshop scheduling problem with makespan criterion. Computers Industrial Engineering, 2017, 111: 239-250.

[8] Zhang G, Xing K. Memetic social spider optimization algorithm for scheduling two-stage assembly flowshop in a distributed environment. Computers Industrial Engineering, 2018, 125: 423-433.

[9] Sheikha Shaya, Komakib G.M., Kayvanfar Vahid. Multi objective two-stage assembly flow shop with release time. Computers Industrial Engineering, 2018, 124: 276-292.

[10] Zhang G, Xing K, Cao F. Discrete differential evolution algorithm for distributed blocking flowshop scheduling with makespan criterion. Engineering Applications of Artificial Intelligence, 2018, 76:96-107.

[11] Li J Q, Duan P, Cao J, et al. A hybrid Pareto-based tabu search for the distributed flexible job shop scheduling problem with E/T criteria. IEEE Access, 2018, 6: 58883-58897.

[12] Shao W, Pi D, Shao Z. A Pareto-Based Estimation of Distribution Algorithm for Solving Multiobjective Distributed No-Wait Flow-Shop Scheduling Problem With S uenceDependent Setup Time. IEEE Transactions on Automation Science and Engineering, 2019.

[13] Seidgar H, Fazlollahtabar H, Zandieh M, et al. Scheduling two-stage assembly flow shop with random machines breakdowns: integrated new self-adapted differential evolutionary and simulation approach. soft computing, 2019: 1-25.

[14] Marandi F, Ghomi S M T F. Network configuration multi-factory scheduling with batch delivery: A learning-oriented simulated annealing approach. Computers Industrial Engineering, 2019, 132: 293-310.

[15] Shen L, Gupta J N D, Buscher U. Flow shop batching and scheduling with s uencedependent setup times. Journal of Scheduling, 2014, 17(4):353-370.

[16] Akbalik A, Rapine C. Lot sizing problem with multi-mode replenishment and batch delivery. Omega, 2018, 81: 123-133.

[17] Wang S, Wu R, Chu F, et al. Variable neighborhood search-based methods for integrated hybrid flow shop scheduling with distribution. soft computing, 2019: 1-20.

[18] Yin Y, Wang Y, Cheng T C E, et al. Two-agent single-machine scheduling to minimize the batch delivery cost. Computers Industrial Engineering, 2016, 92: 16-30.

[19] Qi X, Yuan J. A further study on two-agent scheduling on an unbounded serial-batch machine with batch delivery cost. Computers Industrial Engineering, 2017, 111: 458-462.
[20] Basir S A, Mazdeh M M, Namakshenas M. Bi-level genetic algorithms for a two-stage assembly flow-shop scheduling problem with batch delivery system. Computers Industrial Engineering, 2018, 126: 217-231.
[21] Noroozi A, Mazdeh M M, Heydari M, et al. Coordinating order acceptance and integrated production-distribution scheduling with batch delivery considering Third Party Logistics distribution. Journal of manufacturing systems, 2018, 46: 29-45.
[22] Jiang T, Zhang C, Sun Q M. Green job shop scheduling problem with discrete whale optimization algorithm. IEEE Access, 2019, 7: 43153-43166.
[23] Kong L, Li H, Luo H, et al. Sustainable performance of just-in-time (JIT) management in time-dependent batch delivery scheduling of precast construction. Journal of cleaner production, 2018, 193: 684-701.
[24] Kazemi H, Mazdeh M M, Rostami M. The two stage assembly flow-shop scheduling problem with batching and delivery. Engineering Applications of Artificial Intelligence, 2017, 63: 98-107.
[25] Agnetis A, Aloulou M A, Fu L L. Production and interplant batch delivery scheduling: Dominance and cooperation. International Journal of Production Economics, 2016, 182:38-49.
[26] Wang K, Luo H, Liu F, et al. Permutation flow shop scheduling with batch delivery to multiple customers in supply chains. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2017, 48(10): 1826-1837.
[27] Basir, Saeedeh Ahmadi, Mohammad Mahdavi Mazdeh, and Mohammad Namakshenas. Bi-level genetic algorithms for a two-stage assembly flow-shop scheduling problem with batch delivery system. Computers Industrial Engineering, 2018, 126:217-231.
[28] Karaboga, D., Gorkemli, B., Ozturk, C., Karaboga, N.. A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artificial Intelligence Review, 2014, 42, 21-57.
[29] Peng K, Pan Q K, Gao L, et al. An improved artificial bee colony algorithm for realworld hybrid flowshop rescheduling in steelmaking-refining-continuous casting process. Computers Industrial Engineering, 2018, 122: 235-250.
[30] Yurtkuran A, Yagmahan B, Emel E. A novel artificial bee colony algorithm for the workforce scheduling and balancing problem in sub-assembly lines with limited buffers. Applied Soft Computing, 2018, 73:767-782.
[31] Liao C J, Tjandradjaja E, Chung T P. An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem. Applied Soft Computing, 2012, 12(6): 1755-1764.
[32] Gao J, Chen R, Deng W. An efficient tabu search algorithm for the distributed permutation flowshop scheduling problem. International Journal of Production Research, 2013, 51(3): 641-651.
[33] Chen T L, Cheng C Y, Chou Y H. Multi-objective genetic algorithm for energy-efficient hybrid flow shop scheduling with lot streaming. Annals of Operations Research, 2018: 1-24.
[34] Ark O A. Population-based Tabu search with evolutionary strategies for permutation flow shop scheduling problems under effects of position-dependent learning and linear deterioration. Soft Computing, 2020.
[35] Mirjalili, S., and Lewis, A. The whale optimization algorithm. Advances in engineering software, 2016, 95:51-67.
[36] Prakash, D. B., and Lakshminarayana, C. Optimal siting of capacitors in radial distribution network using whale optimization algorithm. Alexandria Engineering Journal, 2017, 56(4): 499-509.
[37] Prakash, D. B., and Lakshminarayana, C. Multiple DG placements in radial distributionsystem for multi objectives using Whale Optimization Algorithm. Alexandria engineering journal, 2018, 57(4): 2797-2806.
[38] Hasanien, H. M. Performance improvement of photovoltaic power systems using an optimal control strategy based on whale optimization algorithm. Electric Power Systems Research, 2018, 157:168-176.
[39] Sun Y, Wang X, Chen Y, et al. A modified whale optimization algorithm for large-scale global optimization problems. Expert Systems with Applications, 2018, 114: 563-577.
[40] Nasiri J, Khiyabani F M. A whale optimization algorithm (WOA) approach for clustering. Cogent Mathematics Statistics, 2018, 5(1): 1483565.
[41] Fu M, Zhonghua H, Zhijun G, et al. Whale optimization algorithm for flexible flow shop scheduling with setup times. 2017 9th International Conference on Modelling, Identification and Control (ICMIC). IEEE, 2017: 157-162.
[42] Mafarja M M, Mirjalili S. Hybrid whale optimization algorithm with simulated annealing for feature selection. Neurocomputing, 2017, 260: 302-312.
[43] Abdel-Basset M, Manogaran G, El-Shahat D, et al. A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem. Future Generation Computer Systems, 2018, 85: 129-145.
[44] Jiang T, Zhang C, Zhu H, et al. Energy-efficient scheduling for a job shop using an improved whale optimization algorithm. Mathematics, 2018, 6(11): 220.
[45] Luan F, Cai Z, Wu S, et al. Improved Whale Algorithm for Solving the Flexible Job Shop Scheduling Problem. Mathematics, 2019, 7(5): 384.
[46] Kirkpatrick S, Gelatt C D, Vecchi M P. Optimization by simulated annealing. Science, 1983, 220(4598):671-680.
[47] Seidgar H, Kiani M, Abedi M, et al. An efficient imperialist competitive algorithm for scheduling in the two-stage assembly flow shop problem. International Journal of Production Research, 2014, 52(4): 1240-1256.