Annals of Operations Research (2022) 310:223–255
https://doi.org/10.1007/s10479-021-03952-1

S.I.: MIM2019

An effective multi-objective whale swarm algorithm for energy-efficient scheduling of distributed welding flow shop

Guangchen Wang[1] · **Xinyu Li**[1] · **Liang Gao**[1] · **Peigen Li**[1]

Accepted: 15 January 2021 / Published online: 5 February 2021

Abstract
Distributed welding flow shop scheduling problem is an extension of distributed permutation flow shop scheduling problem, which possesses a set of identical factories of welding flow shop. On account of several machines can process one job simultaneously in welding shop, increasing the amount of machines can short the processing time of operation while waste more energy consumption at the same time. Thus, energy-efficient is of great significance to take total energy consumption into account in scheduling. A multi-objective mixed integer programming model for energy-efficient scheduling of distributed welding flow shop is presented based on three sub-problems with allocating jobs among factories, scheduling the jobs in each factory and determining the amount of machines upon each job. A multi-objective whale swarm algorithm is proposed to optimize the total energy consumption and makespan simultaneously. In the proposed algorithm, a new initialization method is designed to improve the quality of the initial solution. And various update operators, as well as local search, are designed according to the feature of the problem. To conduct the experiment, diversified indicators are applied to evaluate the proposed algorithm and other MOEAs performance. And the experiment results demonstrate the effectiveness of the proposed method. The proposed algorithm is applied in the real-life case with great performance compared with other MOEAs.

Keywords Distributed welding flow shop · Energy-efficient scheduling · Whale swarm algorithm · Multi-objective optimization

✉ Liang Gao
gaoliang@mail.hust.edu.cn

Guangchen Wang
wangguangchen620@163.com

Xinyu Li
lixinyu@hust.edu.cn

Peigen Li
pgli@mail.hust.edu.cn

[1] State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074, China

1 Introduction

In order to keep the closer step to the globalization trend, more managers transform the traditional single factory to distribued factories to meet the requirement of the market with high quality, low risk and quick response (Kahn et al. 2005; Wang et al. 2020). Under the distributed environment, Naderi and Ruiz (2010) named distributed permutation flow shop scheduling problem (DPFSP, denoted as DF/*prmu*/$C_{\max}$). In this study, DPFSP was proved as an NP-hard problem. As the specific case when $F=1$ is the well-known PFSP already proved NP-hard (Garey et al. 1976). Gao and Chen (2011) solved the DPFSP by using the genetic algorithm with local search (GA-LS) to get better results than Naderi. In the same year, Wang et al. (2013) put forward the estimation of distribution algorithm (EDA) for the DPFSP. According to the comparison, EDA is the most effective algorithm until Lin et al. (2013) proposed a modified iterated greedy (IG) algorithm. As introduced in the paper (Lin et al. 2013), IG got the best results in the Taillard benchmark (Taillard 1993) with more than half of the instance. Bahman Naderi and Ruiz (2014) proposed a hybrid scatter search (SS) with some advanced techniques and SS further improved the result in a comprehensive computational campaign including 10 existing algorithms, such as hybrid genetic algorithm (HGA) and tabu search (TS) proposed by Gao et al. (2013) and Chan et al. (2013). Fernandez-Viagas and Framinan (2014) studied the bounded-search IG (BIG) in the foundation of IG which has a better result than EDA (Wang et al. 2013) and the TS (Gao et al. 2013). In recent three years, the study of DPFSP has great development. Shao et al. (2017) studied the DPFSP with no-wait constraint and proposed effective algorithm IG with NEH initialization and local search methods. Different from this paper, Ruiz et al. (2018) extended the NEH initialization by adjusting the assignment of factory and sequence in the lowest makespan. For distributed assembly permutation flow shop scheduling problem, Pan et al. (2019) summarized constructive heuristics and meta-heuristics for this DPFSP extended problem. Different from the most criterion DPFSP with the objective of makespan, Fernandez-Viagas et al. (2018) studied DPFSP with the objective of total flow time. For solving this kind of problem, eighteen constructive heuristics and an iterative improvement algorithm are designed to obtain high-quality solutions. In order to solve DPFSP with total flowtime criterion, Pan et al. (2019) presented various heuristics at the foundation of LR, NEH heuristics and four metaheuristics at the foundation the effective frameworks of remarkable algorithms in comprehensive computational comparison. Wang et al. (2019) studied the energy-efficient DPFSP with controllable machine speed to optimize the makespan and energy consumption.

With the lack of the study related to DPFSP in welding shop, this paper study the DPFSP for welding shop which is the specific production shop in the manufacturing environment. It could be regarded as a special case of the assembly scheduling problem, such as girder welding shop related to the assembly of components. In general, traditional scheduling has the basic assumption (Gao et al. 2014; Grobler et al. 2010): each job must be operated by only one machine at a time. However, in most real-life welding shops, the engineers use more than one welding machine to improve production efficiency. To classify this kind of problem clearly, it could be regarded as multi-parallel processor tasks (MPPT) (Li et al. 2019a, b) problem. According to the feature of the problem, the amount of machine could be determined within the total number of such machines. Therefore, this assumption no longer meets the status of some real-life welding shops. And the traditional models and methods cannot formulate the

WSSP well (Marichelvam and Prabaharan 2015). Welding production wildly exists in real-life production and it requires a huge amount of energy consumption which intensify a large amount of CO_2 emission and global warming (Lu et al. 2018). In recent years, more works related to the welding shop scheduling problem (WSSP) have been constructed. As shown in the paper (Lu et al. 2016), WSSP is more sophisticated in real-life production because of its realistic constraints. Therefore, the traditional models and methods cannot formulate the WSSP well. Lu et al. (2016) presented a mathematical model related to WSSP with the objectives of makespan and machine load. Furthermore, the authors formulated a new mathematical model of dynamic WSSP with the objectives of makespan, instability and machine load (Lu et al. 2017a, b). The previous work of WSSP was only related to product effectiveness. But, there are few studies of WSSP with the consideration of environment indicators: carbon emission, energy consumption. Thus, considering the objective of energy consumption, Li et al. (2018) presented a modified multi-objective artificial bee colony algorithm for WSSP with consideration of productivity and energy cost simultaneously. At the foundation of the previous study of WSSP and DPFSP, this paper proposes a distributed welding flow shop scheduling problem (DWFSP) model considering welding manufacturing characteristics to minimize makespan and energy consumption simultaneously.

When the problem scale grows, the computation time exponential grows and it becomes a balance problem between the computation cost and quality of the solution. Facing this problem, metaheuristic algorithms (Pei et al. 2019)is a great way to solve this complex problem. In this study, we apply a multi-objective algorithm at the foundation of the newly designed whale swarm algorithm (WSA) (Zeng et al. 2019). This algorithm is developed with the niching method which has its advantage in avoiding falling into local optima. The niching method could update the individuals with evolution environment. On account of WSA is a newly proposed algorithm, it has been applied in continuous problems with the multimodal benchmark (Zeng et al. 2019) and we modify it as a multi-objective whale swarm algorithm (MOWSA) to solve discrete scheduling problems in this study.

With the algorithm design, the adapted multi-objective discrete optimizer with well-designed solution initialization and update operators is applied in MOWSA. Optimize the three sub-problems simultaneously, encoding representation is designed including three parts, factory assignment, permutation of jobs, and amount of machines which could represent three sub-problems separately. Moreover, to enhance the quality of the solution, critical factory based local search with makespan optimization and machine amount based local search with total energy consumption optimization respectively. And non-dominated sorting is applied to the combination of original populations and newly generated populations in the selection operator.

The structure of the paper is organized as follows: problem definition and the mathematical model is stated in Sect. 2. A newly proposed algorithm with constructive heuristic is proposed to solve the DWFSP in Sect. 3. And the comparison experiment with other multi-objective evolution algorithms (MOEAs) is developed in Sect. 4. A real-life case of study is described in Sect. 5. Conclusion and future research are listed in Sect. 6.

2 Problem formulation

2.1 Problem definition

The energy-efficient DWFSP can be defined as follow. A set of n jobs should be assigned to f identical factories. Each factory includes a flow shop with a set of m stages. Each factory in i stage has L_i welding machines. DWFSP has its constraints: only one job could be processed on each machine at a time and all jobs in its factory have to be processed in the same sequence. Meanwhile, there is at least one machine to operate on each job. Therefore, each job j assigned to factory k on i operation stages and selects its $M_{k,i}$ welding machines. When only one machine is used to process the operation, it could be processed with its normal processing time. Thus, the actual processing time is related (i.e., compressed) with allocating multiple welding machines. Considering a set of discrete and finite machine assignments with each job, the processing time is controllable and affected the machine amount in the assignment. There is no job preemption among all jobs. And the setup time-related stage and job took into account. Compared to the processing time and setup time, release times of each stage are relatively short which could be neglected. In this study, the assumptions and formulations are constructed at the foundation in Li's paper (Li et al. 2018). In the energy-efficient DWFSP, basic energy consumption, idle energy consumption, setup energy consumption, and welding energy consumption as shown in Fig. 1a are taken into account. There is a small example of DWFSP as shown in Fig. 1b, c. There is a real-life WSSP with five operations in Fig. 1b. From Fig. 1c, five jobs are assigned to three factories and each factory has five stages as in Fig. 1b. It is obvious that job2 is assigned to factory 1 and its machine amount of stage 2 is two which means 2 machines are operated on the job2 on stage 2 in factory 1 simultaneously.

2.2 Problem modeling

To facilitate the description of the problem, the notations in this paper are given below:
Parameters:

n : The number of jobs to process.
m : The number of stages in each factory
f : The number of factories
j : Index of job
g : Job position in a sequence
k : Index of factories
i, h : Index of stage
L_i: quantity of machines on the ith stage, L_i.is a constant
A: it is a very large positive number.
$np_{i,j}$: The normal processing time of job j on machine i
$st_{i,j}$: The setup time of the job j at machine i
P_{basic}: energy power in basic mode (kW)
P_{idle}: energy power when the machine is idle (kW)
P_{setup}: setup energy power (kW)
$P_{welding}$: welding energy power (kW)
K_w: welding duty cycle (%)

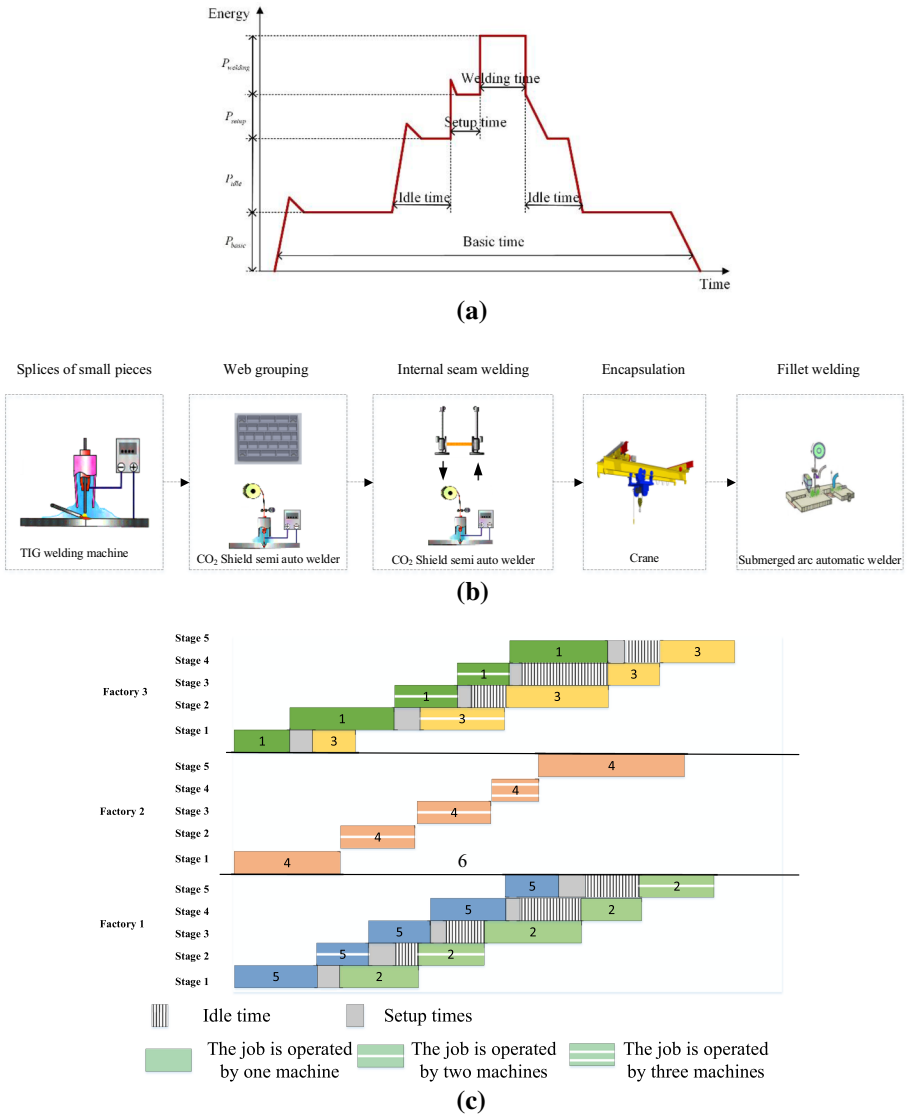


Fig. 1 **a** Welding shop energy consumption, **b** a flow chart for a welding shop scheduling plant, **c** an example of the problem

Variables:

$X_{j,g,k}$: The binary variable that takes value 1, if job j is assigned to factory k in g th position, and 0 otherwise
$p_{k,i,j}$: The actual processing time of job j on machine i in factory k
C_{max} : The makespan of the schedule
$C_{k,i,g}$: The completion time of g th job in sequence on i th machine in k th factory
$S_{k,i,g}$: The start time of the job g th job in sequence at machine i in factory k

$\mu_{k,i,j}$: quantity of machines capable of processing with job j at stage i in factory k

1. Basic energy consumption

It is due to auxiliary production work: supply system, control system, and welding protective gas. Basic energy consumption which is shown as formula (1) is generated along the whole process.

$$E_{basic} = P_{basic} \times t_{basic} \tag{1}$$

where $t_{basic} = makespan$ and t_{basic} is changed with different schedules. Thus, Basic energy consumption is influenced by the schedule change. Therefore, the consideration of basic energy consumption is necessary.

2. Idle energy consumption

It refers to the existence of no-load loss in the non-processing stage. In general, this portion is little, but it still cannot be ignored. In the WSSP, energy consumption is affected by the amount of used machines for the job in each process, and detail is given by formula (2):

$$E_{basic} = \mathrm{P}_{basic} \times \mathrm{t}_{basic} \tag{2}$$

The idle time is denoted as follows:

$$t_{idle} = \sum_{k=1}^{f} \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{g=1}^{n-1} \mu_{k,i,j} X_{j,g+1,k} \cdot \left(C_{k,i,g+1} - C_{k,i,g} - st_{i,j} - p_{k,i,j} \right) \tag{3}$$

referring to each processing part after removing job preparation and welding processing time. Since there is a process that multiple machines dealing with a single job, the idle energy consumption include the idle energy consumption that all the machines have.

3. Setup energy consumption

The setup mode often involves job unloading/loading and the pretreatment process of welding (Shrivastava et al. 2015). This part of energy consumption is mainly influenced by total setup time, which is formulated in formula (4) (Shrivastava et al. 2015):

$$E_{setup} = P_{setup} \times t_{setup} \tag{4}$$

where $t_{setup} = \sum_{g=1}^{n} \sum_{j=1}^{n} \sum_{i=1}^{m} \mu_{k,i,j} \cdot st_{i,j}$, because the setup time mainly relates to the job, machine factors and so on, the impact of a number of machines is considered.

4. Welding energy consumption

This part takes up the majority of total energy consumption related to loading energy consumption and idle energy consumption. And it is given in formula (4) (Si et al. 2010):

$$E_{welding} = \left[P_{idle} \cdot \left(1 - K_w \right) + P_{welding} \cdot K_w \right] \times t_{welding} \tag{5}$$

where $t_{welding} = \sum_{k=1}^{f} \sum_{j=1}^{n} \sum_{i=1}^{m} p_{k,i,j}$, it denotes as processing time of welding relating to the efficiency of welder.

Based on the above formulas (1)–(4), the Total Energy Consumption (TEC) is denoted as the following formula (5) (Shrivastava et al. 2015):

$$TEC = E_{basic} + E_{idle} + E_{setup} + E_{welding} \tag{6}$$

To get more accurate total energy consumption, four parts of energy consumption as basic energy consumption, idle time, setup time and processing time with different energy consumption are taken into account. Setup times such as clamping and fixing the position of the job are taken into consideration.

The mathematical model of a multi-objective DWFSP is as follow:

$$\min f_1 = \min\{\max_{k \in F}\{C_{k,m,n}\}\} \tag{7}$$

$$\min f_2 = TEC \tag{8}$$

$$\begin{aligned} &s.t. \\ &S_{k,1,1} = 0, \quad \forall k \in \{1, \ldots f\} \end{aligned} \tag{9}$$

$$\sum_{j=1}^{n} \sum_{k=1}^{f} X_{j,g,k} = 1, \quad \forall g \in \{1, \ldots, n\} \tag{10}$$

$$\sum_{g=1}^{n} \sum_{k=1}^{f} X_{j,g,k} = 1, \quad \forall j \in \{1, \ldots, n\} \tag{11}$$

$$1 \le \sum_{g=1}^{n} \sum_{j=1}^{n} \mu_{k,i,j} X_{j,g,k} \le L_i, \quad \forall i \in \{1, \ldots m\}, \quad \forall k \in \{1, \ldots f\} \tag{12}$$

$$S_{k,i,g+1} \ge S_{k,i,g} + \sum_{j=1}^{n} X_{j,g,k} \left(\frac{np_{i,j}}{\mu_{k,i,j}} + st_{i,j} \right), \quad \forall k \in \{1, \ldots f\}, \quad \forall i \in \{1, \ldots m\}, \quad g \in \{1, \ldots n-1\} \tag{13}$$

$$S_{k,i+1,g} \ge S_{k,i,g} + \sum_{j=1}^{n} X_{j,g,k} \frac{np_{i,j}}{\mu_{k,i,j}}, \quad \forall k \in \{1, \ldots f\}, \quad \forall i \in \{1, \ldots m-1\}, \quad \forall g \in \{1, \ldots n\} \tag{14}$$

$$C_{k,i,g} = S_{k,i,g} + \sum_{j=1}^{n} X_{j,g,k} \left(\frac{np_{i,j}}{\mu_{k,i,j}} + st_{i,j} \right), \quad \forall j, g \in \{1, \ldots n\} \tag{15}$$

$$X_{j,g,k} \in \{0, 1\}, \quad \forall k \in \{1, \ldots f\}, \quad \forall i \in \{1, \ldots m\}, \quad \forall j, g \in \{1, \ldots n\} \tag{16}$$

The objective as formula (7) is related to one of objectives to minimize the makespan. And formula (8) is to minimize the TEC. Constraint (9) is to determine the start time of the first job of each stage in each factory. Constraint (10–11) ensure every job could

be assigned to factories and every job only belongs to one factory one position of the sequence. Constraint (12) represents that each operation's machine amount is larger than one but within its maximal value. Constraint (13) ensures that one operation must start after its previous operation related to another job is completed on the same stage. Constraints (14) ensures that one operation can start after its previous operation of the same job is completed. Constraints (15) states the relationship between the operation's start time and the completion time. Constraints (16) is 0–1 variable constraints.

3 Proposed multi-objective whale swarm algorithm for energy-efficient DWFSP

The MOWSA is proposed at the foundation of the original WSA proposed by Zeng et al. (2019). Inspired by the whales' behavior of communicating with each other via ultrasound for hunting, WSA is designed for continuous optimization problems. For solving DWFSP a discrete optimization problem, the original WSA must be modified. Moreover, non-dominated sorting and crowding distance strategy are applied in MOWSA to obtain Pareto front with a non-dominated solution set instead of selecting individuals. In WSA's original procedure, a whale X will move under the guidance of its "better and nearest" whale Y according to the formula (17) as follows.

$$x_i^{t+1} = x_i^t + rand\left(0, \rho_0 e^{-\eta d_{x,y}}\right) \times (y_i^t - x_i^t) \tag{17}$$

As in Eq. (17), x_i^t represents i-th element's position of $\boldsymbol{X}$ at t iterations, and x_i^{t+1} denotes $\boldsymbol{X}$'s position at $t+1$ iterations respectively. Similarly, y_i^t represents the i-th element of $\boldsymbol{Y}$'s position at iteration t. ρ_0 this equation denotes the intensity of ultrasound source, according to the original WSA, it is set to 2 for almost all the cases. e is the natural constant. η denotes the attenuation coefficient. Moreover, $d_{x,y}$ represents the Euclidean distance between $\boldsymbol{X}$ and $\boldsymbol{Y}$. $rand\left(0, \rho_0 e^{-\eta d_{x,y}}\right)$ is a random value generated between 0 and $\rho_0 e^{-\eta d_{x,y}}$ uniformly.

In the whale population, most whales have their own "better and nearest" whale. At the foundation of Eq. (17), each individual is willing to move to its "better and nearest" whale and move randomly away from negative whale. In WSA, the population will contribute a lot to WSA's great outperformance of diversity. Furthermore, WSA has a strong ability to locate global optima with high accuracy.

According to the characteristic of energy-efficient DWFSP, MOWSA is proposed with the basic idea of finding the "Better and nearest" whale for genetic operators, such as crossover and mutation operators. To strengthen the quality of solutions, a critical factory based local search is proposed to enhance the production efficiency with makespan. Similarly, to enhance energy-efficiency, machine amount based local search is proposed to search the solution with low TEC. The details of MOWSA is shown in Fig. 2.

3.1 Solution representation

In this study, we proposed a machine amount adjusted encoding scheme for the energy-efficient MOWSA to solve this DWFSP, Therefore, a multi-chromosome structure is defined, which includes a permutation of n jobs, and factory assignment of each job and

Algorithm The details procedure of the MOWSA

Step 1. Initialize the population

Step 1.1: Generate initial population of size N. 1/5 of population are generated with MODNEH method and 4/5 individuals are generated with the random method.

Step 1.2: Evaluation of initial population.

Step 2. Evolution

Step 2.1: Select one whale denoted as a in parent population randomly. Calculate the distance from it to each other individuals.

Step 2.2: Compared these distance values and search its nearest one as a' which is better than whale a. If it could not be found, go to step 2.4.

Step 2.3: Update the whale a with whale a'.

Step 2.4: Select one whale denoted as b in parent population randomly.

Step 2.5: Update whale b and whale a' with genetic operators and generate the new two offspring whales.

Step 2.6: Update two offspring whales with critical factory based local search if new offspring whales dominate the original whales.

Step 2.7: Update two offspring whales with machine amount based local search if new offspring whales dominate the original whales.

Step 2.8: Select the new parent polutation from the combination of the offspring population and parent population by NDS and CD.

Step 3. Termination criteria.

Fig. 2 MOWSA algorithm

a machine vector of m stages corresponding each job's machine amount assignment, respectively. The solution representation for an individual x_i is shown in Fig. 3.

The individual's multi-dimensional vector as $x_i(\pi, a, N)$ represents ith solution where job $\pi_{i1} = 5$ is represented as the first position in the permutation of jobs and $a_{i5} = 1$ is represented as job 5 assigned to factory 1 and job 5 on stage 1 has the machine amount of $(N_{i15} = 2)$; similarly, job $\pi_{i2} = 2$ and its factory assignment is $a_{i2} = 1$, moreover job 2 on stage 3 has the machine amount $(N_{i32} = 1)$ and so on. Note that the different machine amount vectors are used in each machine.

Fig. 3 Solution representation

$$\begin{array}{r} \quad J_5 \;\; J_2 \;\; J_1 \;\; J_4 \;\; J_3 \\ \pi = [5 \;\; 2 \;\; 1 \;\; 4 \;\; 3] \end{array}$$

$$\begin{array}{r} \quad F_3 \;\; F_1 \;\; F_3 \;\; F_2 \;\; F_1 \\ a = [3 \;\; 1 \;\; 3 \;\; 2 \;\; 1] \end{array}$$

$$\begin{array}{r} \quad J_1 \;\; J_2 \;\; J_3 \;\; J_4 \;\; J_5 \\ \begin{array}{r} Stage1 \\ Stage2 \\ N = Stage3 \\ Stage4 \\ Stage5 \end{array} \begin{bmatrix} 2 & 1 & 2 & 1 & 2 \\ 1 & 2 & 1 & 3 & 2 \\ 3 & 1 & 2 & 1 & 3 \\ 2 & 2 & 1 & 3 & 1 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix} \end{array}$$

3.2 Initialization

The Multi-objective DNEH (MODNEH) method is designed in MOWSA initialization which is firstly proposed in solving energy-efficient DWFSP. This method is designed inspired by the Multi-objective NEH (MONEH) method which was proposed by Ding et al. (2016) for initialization and Distributed NEH (DNEH) proposed by Pan et al. (2019) for distributed permutation flow shop with total flow time.

The MODNEH method obtains the partially non-dominated solution by assigning a job to a factory and inserting it in this factory's job sequence until the whole sequence is generated. The first step is to sort of job in the non-ascending order of their total operation's processing times as an original job permutation which is similar to the DNEH. The main idea of MODNEH is as follow: Denote NS_j as the non-dominated set of sub-solution with j jobs ($j = 1, \ldots, n$) and their factory assignment. This non-dominated set could be transformed into the set of sub-solution with several factories and their sub-solution with jobs. NS_1 is the initialized with a given machine matrix and F jobs that are assigned in each F factories based on the non-ascending order of their total processing time. Next, the method performs $n-1$ iterations to generate a set of non-dominated solutions. In the k-th iteration, a job removed from the original job permutation should be inserted into the sequence in the partial solution set. To be specific, assigning each job from the job permutation by inserting it into all possible positions of all factories which could obtain the partial job permutation and job assignment until the last job assigned. Then the new non-dominated sequence is obtained by Pareto dominated selection rule. To avoid the explosion in non-dominated solutions, some of the crowded non-dominated partial solutions are removed based on the crowding distance. And the details of MODNEH is described in Fig. 4.

The method proposed by Deb et al. (2002) is used to calculate the crowding distance. The details of the CD method as follows in Fig. 5. And C and E are denoted as the objectives of makespan and TEC respectively.

Procedure MODNEH

Caculate the sum of processing time of each job
Obtain the initial job sequence $\lambda = (\lambda_1, \lambda_2, ..., \lambda_n)$ according to the sum of processing time by non-increasing order
$\pi := \{\pi_1, \pi_2, ..., \pi_f\}$, where $\pi_k = (\lambda_k)$, $k = 1, 2, ..., f$
Extract job j from λ
while $capacityof(\lambda) > 0$ **do**
for k:=1 to f **do**
Insert and calculate makespan after inserting job j into the position of π_k
$C_k :=$ Pareto dominated selection rule and crowding distance after j inserted
$\xi_k :=$ related position in Δ_k
end for
$k^* :=$ $\arg(\min_{k=1,2,...,F}(\Delta_k))$
Insert job j into the position ξ_{k^*} of π_k
end while
return π

Fig. 4 The procedure of MODNEH

On account of the wide searching field in the DWFSP and the great quality of the solution, the initial population about factory assignment and permutations of jobs of each factory are generated by the MODNEH method. This method could produce a better solution than it is generated randomly. To balance the quality and the diversity of the solution set, the 1/5 individuals of the population are produced by the MODNEH. The rest 4/5 individuals are generated by Random Method (Zhang et al. 2018) to guarantee the diversity of the population. Furthermore, the machine matrix of all individuals is generated randomly with the setting range of the problem.

Algorithm : ***crowding_distance(P)***

Input: Population parameters $C = (C_1, \cdots, C_{popsize}), E = (E_1, \cdots, E_{popsize})$
Output: the crowding distance $dist(i)$ for each solution $dist(i)$ $(i = 1, \cdots, popsize)$
Sort **C** in the increasing order. Denote the sorted sequence as $(C_{I(1)}, \cdots, C_{I(popsize)})$
Where $I = (I(1), \cdots, I(popsize))$ is the corresponding index vector.
Set $dist(I(1)) := +\infty, dist(I(popsize)) := +\infty$
For $i = 2$ to $popsize - 1$ do
$dist_C \leftarrow \left| C_{I(i+1)} - C_{I(i-1)} \right| / (\max C - \min C)$
$dist_E \leftarrow \left| E_{I(i+1)} - E_{I(i-1)} \right| / (\max E - \min E)$
$dist(I(i)) \leftarrow dist_C + dist_E$
End for

Fig. 5 Pseudocode of the crowding distance calculation method

3.3 Whale parents' update

One of the key elements of the proposed MOWSA is to search the better nearest solution to update the parent population in each iteration. To calculate the distance of two solutions, the distance could be regarded as the sum count of a binary numerical difference of each element according to the solution representation above. In this procedure, the first level as job permutation and the second level of factory assignment are taken into account. There are $2n$ elements in these two levels of each solution. The first step of calculating the distance of the two solutions is to judge whether there is any difference between the two elements in the same position with two compared solutions. If there is not any difference between two elements, the distance of two elements' position is denoted as 0, else denoted as 1. Then, sum all elements in the first and the second level of the solution as the distance of the solution. There is an instance of distance calculation of two solutions in Fig. 6. There are two solutions compared with each other. In permutation π_1 and π_2, we judge each position in two permutations whether there is the same value and get the distance of job permutation dis_{job}. It is obvious that the first position of job permutation is different between π_1 and π_2. Thus the distance of the first position is denoted as 1. Thus, the job distance of π_1 and π_2 is denoted as $dis_{job} = [1\,0\,0\,1\,1]$. The same procedure as in obtaining the distance of factory assignment dis_{fac} as $dis_{fac} = [1\,1\,0\,1\,1]$. And combine the two distance values as the distance of the solution which is denoted as $dis = 7$.

When two individuals are selected from the parent population randomly as individual a and individual b. Individual a is to search the nearest individual with individual a updating as individual a' meanwhile individual b remaining unchanged in the parent population. And, the individual a' and individual b are considered as the two individuals for crossover operator.

3.4 Crossover operator and mutation operator

Genetic operators such as crossover and mutation are widely applied in existed discrete scheduling problem. According to crossover will enhance the diversity of the population, these operators generally applied with definite possibility value. But it might produce an infeasible solution with problem constraint. To avoid the unfeasible solution in evolution

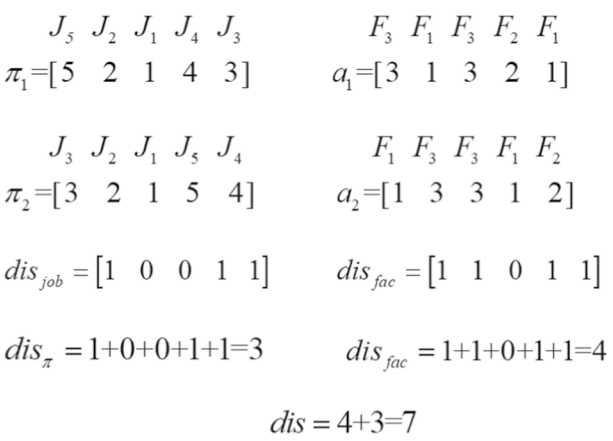


Fig. 6 An instance of distance calculation

procedure, a partially matched crossover (PMX) (Goldberg and Lingle 1985) is adopted in crossover operator with the permutation sequence. We give an example of PMX crossover as in Fig. 7. Firstly, a partial solution of the parent will be select in parent individual with two random point. Then, exchange the partial solution with two parents. It is obvious that two current offsprings obtained are infeasible and they need to be adjusted. From Fig. 7, job 5 and job 3 are repetitive in current offspring 1. Meanwhile, it the same situation with job 1 and job 4 in offspring 2. Therefore, it needs a specially designed way to delete the repetitive job and add the missing job in current offsprings. From the yellow partial part in two current offsprings, job 5 is matched with job 2 at the same position in current offsprings. Meanwhile, there is no repetitive job 2 in current offsprings. Therefore, job 2 cannot be the final matched job. Next, job 2 in current offspring 1 is matched with job 1 at the same position in current offspring 2. Therefore, it could satisfy the sequence rule when job 5 with the white box will be transform into job 1 in the end. It is the same rule when job 3 could be transformed into job 4. It is an effective rule when it occurs the repetitive job in permutation sequence in PFSP.

Moreover, factory assignment in each job and machine amount setting is independent of each other and these two levels have no such constraint as permutation. For the factory assignment, it is no sense that any factory has no job to be processed which is the constraint satisfied in initialization. In this study, two-point crossover (TPX) operation as shown in Fig. 7 is introduced in these two levels which could satisfy the constraint. When it operate with the factory assignment, it could obtain unchanged amount of jobs in each factory and each factory could not be empty in evolution procedure. In TPX operator, it has same procedure to select two points in parent individuals randomly. Next, exchange the partial sub-sequence between the two points in yellow box. For the machine amount matrix, it is the same procedure. Therefore, TPX will not change the the category or number of machine amount. And it will also ensure that each change will occur within factory constraint.

As for the mutation operator, it is a supplementary operator with a crossover operator during searching for potential solutions. As the same in the crossover operator, the feasibility of the solution should be taken into account. Seen from Fig. 7, a double-point exchange

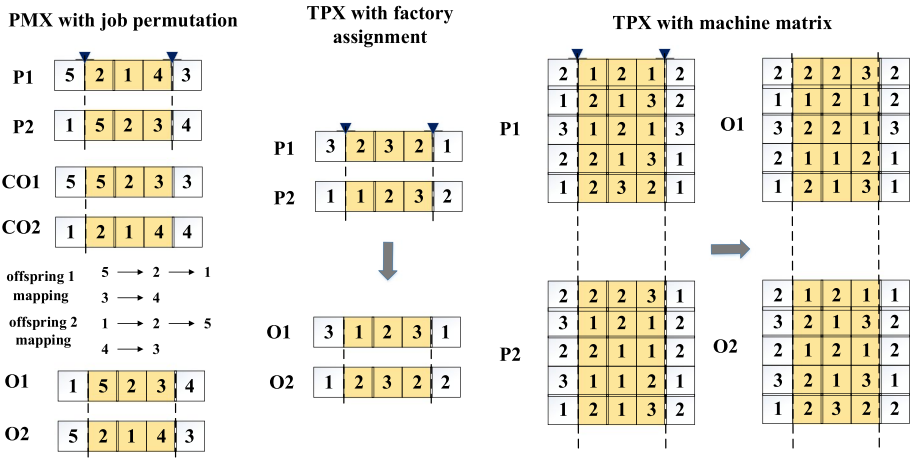


Fig. 7 An instance of crossover in three levels of solution representation

Fig. 8 An instance of Critical factory based local search ▶

mutation operation could maintain each element in the job sequence which could ensure the feasibility of the solution. However, as for assignment permutation, it needs more disturbance. In the mutation procedure, select the job randomly and change its corresponding factory. In case there exists an empty factory with no job, assign a job randomly to this empty factory until the occasion disappears. Different from the factory assignment, the mutation operator of the machine assignment matrix, select a job and stage and adjust its machine amount randomly within its setting range. It is mentionable that same with other metaheuristics all kinds of mutation operators proceed with a small possibility.

3.5 Critical factory based local search

As for the ordinary evolutionary method, local exploitation with an appropriate approach could enhance the effectiveness of the algorithms (Hansen et al. 2010; Li et al. 2019a, b; Nanda and Panda 2014). To balance the local exploitation and global exploitation, a critical factory based local search strategy is proposed to improve the MOWSA performance with the objective of the makespan. This strategy could be operated on offspring individuals at each generation with the perspective of productivity.

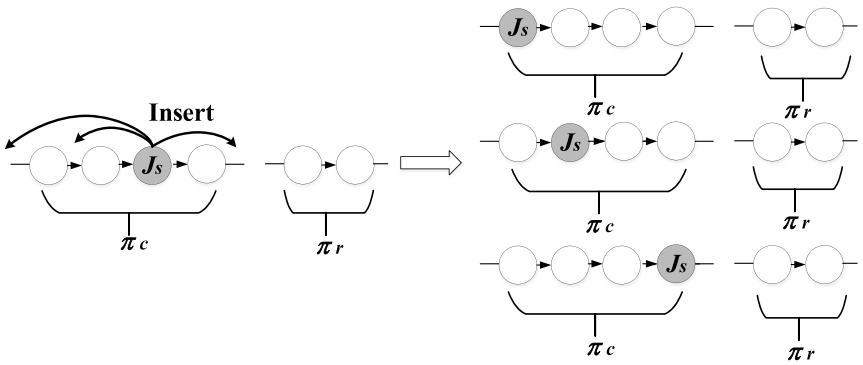
In this study, there are two objective: makespan and TEC. And makespan is denoted as the maximum completion time with all job. Thus, it is related to the factory with biggest completion time in DPFSP which is named critical factory (Zhang et al. 2018). The key point is to balance the completion time between each factory with the equal level. Therefore, we introduce the different strategies to readjust the critical factory in this procedure. According to the feature of the DWFSP, four types of local search strategies are designed as Fig. 8.

Inter-critical-factory job insertion operator (ICFJ insertion-operator): After obtaining the critical factory f_c and its job sequence, insert j_s to other possible positions in f_c. Thus this operator is operated within the first level of solution, without any change of the second level. And reassign it to the optimal insertion position to produce the least makespan as shown in Fig. 8a.

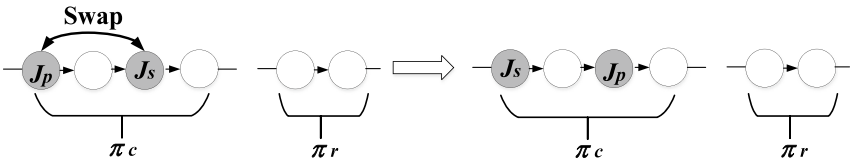
Inter-critical-factory job swap operator (ICFJ swap-operator): Within this job sequence in a critical factory, select a job denoted as j_s ($s \neq p$) and another job denoted as j_p which is not a neighboring job with the j_s in critical factory f_c randomly. Then, swap the position of j_s and j_p in the permutation without any change of the second and third level of solution as shown in Fig. 8b. It is a specific case when selecting a neighboring job with the j_p is the same operator as the ICFJ insertion-operator.

Exter-critical-factory job insertion operator (ECFJ insertion-operator): This operator is related to a critical factory and non-critical factory and it is operated on both of the permutation and the factory assignment. Firstly, select a job as j_s within a critical factory and non-critical factory f_n randomly. Then, insert the j_s to the possible position in factory f_c and assign j_s to the optimal position in factory f_n to obtain the least makespan as shown in Fig. 8c.

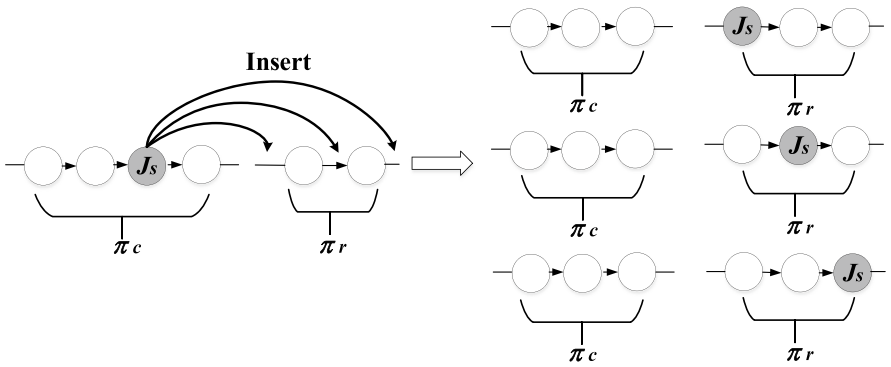
Exter-critical-factory factory swap operator (ECFF swap-operator): Similar to ECFJ insertion-operator, this operator is operated between critical factory f_c and non-critical factory f_n. Thus, this operatory is operated on the solution's first level and second level. Swap the factory number of j_s in factory f_c and j_p in factory f_n to search the least makespan as



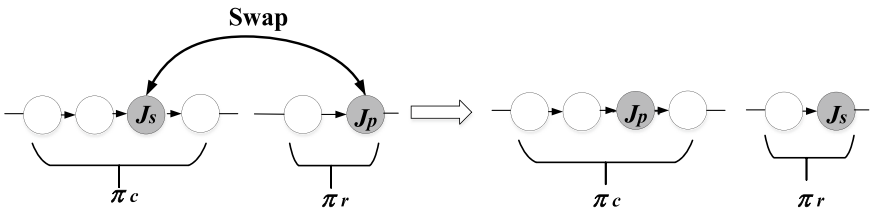
(a) ICFJ insertion-operator



(b) ICFJ swap-operator



(c) ECFJ insertion-operator



(d) ECFF swap-operator

shown in Fig. 8d. The amount of the job in each factory stays unchanged but the factory assignment and the position of j_s and j_p.

3.6 Machine amount based local search

In this study, TEC as an objective should be optimized. We design Machine amount based local search to adjust the machine matrix separately. Different from the makespan, TEC is effected by all factories. Thus, adjust with the limited range could optimize the TEC. In critical route in the critical factory, the more machines are, the faster the operation is, the much less makespan is. But in another case, no matter how fast the operation is finished, the machine of the next operation is likely to be idle. Therefore, this unnecessary idle time will cause unnecessary energy consumption. The way to reduce the unnecessary idle is to prolong the processing time which could fill the idle phase to cut down the extra energy waste without affecting the makespan. The main key is to judge whether it could reduce the amount of machine or whether it could increase the amount of machine to minimize the makespan but it may cause energy consumption simultaneously. The mechanism does not require to change the job permutation and factory assignment but cut down the TEC whilst keeping the same makespan. This procedure is related to two sub-procedures which are related to a critical factory and all factories where idle time exists. The detailed process of the energy-efficient based local search as shown in Fig. 9.

This heuristic search procedure is designed to optimize energy consumption and strengthen the diversity of the solutions. Before the update procedure, we should affirm the critical factory and if each operation could add one machine amount within its range. On account of this operator will change which is the critical factory, the procedure operated on the critical factory should ensure the operator operated on the factory which is a critical factory. Therefore, if the critical factory is changed, the procedure will be ended and return the updated solution.

To optimize the current solution furtherly, the heuristic search procedure is to cut down the unnecessary idle time in the second sub-procedure. Firstly, we should ensure the operation that is to be updated should satisfy some constraints: such as the amount of machine before the operation should be more than one and the waiting time before the next operation is longer than the extended processing time. In this way, the procedure could be operated within the allowed expansion and would not affect the following operation in the permutation range. Thus, makespan has not any change when compared to it with the original machine amount matrix. Meanwhile, the TEC is optimized through this strategy.

3.7 Non-dominated sorting algorithm

The solutions in population obtained by the procedure above will be sorted by the NDS algorithm with CD (Deb et al. 2002). On account of NSGA-II is recognized as one of the most sophisticated MOEAs in previous research, we applied NDS with CD to

Energy-efficient based local search (π, a, r)

f= critical factory
  for g=1 to n_f do
    get each job sequence in the factory k
      for i=1 to m do
        if $R_{f,i,\mathrm{g}} > R_{f,i}^{\max}$ &&rand(0,1)<0.5// $R_{f,i}^{\max}$ is the maximal value of machine i in factory f
          $R'_{f,i,\mathrm{g}}=R_{f,i,\mathrm{g}}+1$
            if Critical factory is changed
              Return (π, a, r')
          end if
      end for
    end for
  end if
end for
for k=1 to f do // f is the total amount of the factory in the run mode
  for g=1 to n_k do // n_k is the total amount of the job in factory k
    get each job sequence in the factory k
      for i=1 to m do // m is the total stage in each factory
        if $idle_{k,i,g+1} > p_{k,i,j}(\frac{1}{R_{k,i,\mathrm{g}}\text{-}1} - \frac{1}{R_{k,i,\mathrm{g}}})$ // $R_{k,i,\mathrm{g}}$ is the number of the machine of job g on the machine i in factory k, $idle_{k,i,g}$ is the idle time of job g+1 on the machine i in factory k
          $R'_{k,i,\mathrm{g}}=R_{k,i,\mathrm{g}}-1$.
        end if
      end for
    end for
Return (π, a, r')

Fig. 9 The pseudocode of energy-efficient based local search

sort the population combined with the offspring population and parent population in MOWSA.

In unconstrained multi-objective optimization, the NDS applies the crowded-comparison operator (CO,≻) as the main point could direct the selection procedure at the various stages of the algorithm. In crowded-comparison operator, each individual *a*

possesses a nomination rank (a_{rank}) and a crowding distance rank ($a_{distance}$). It defines a CO as follows:

$$\text{if}\left(a_{rank} < b_{rank}\right)\text{then} \quad a \succ b$$
$$\text{or}\left(\left(a_{rank} = b_{rank}\right)\text{and}\left(a_{distance} > b_{distance}\right)\right)$$

Thus, an individual with a lower (better) rank is preferred when they have different non-domination ranks. When both individuals have the same rank, an individual with the lesser-crowded region is preferred. After obtaining offspring population Qt, it will be combined with the parent population Pt to apply NDS with CD technique and form the parent population with size N for the next generation.

In this algorithm, its first step is to confirm the following The property parameters of each solution: (1) n_p denotes the number of solutions which dominate the solution p, and (2) S_p, a set of solutions that the solution p dominates. It should be noted that each solution in the first non-dominated front would set its n_p as zero. Then, for each solution p with $n_p = 0$, visit member q of its set S_p one by one and reduce its domination count by one. If any member q's domination count is 0, it would be put in a separate list Q where all members are assigned in the second non-dominated front. This process continues until all Pareto fronts are obtained.

The pseudocode of the NDS is described in Fig. 10.

4 Experimental results and discussion

4.1 Performance metric

1. Generational Distance (GD) (Deb et al. 2002): the metric indicator denotes the distance between obtained pareto front (PF) and optimal pareto front (PF*), and it is formulated as:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{18}$$

where di represents the Euclidean distance between the i-th point of PF and the nearest point of the PF*, and n denotes the amount of point in PF found heretofore. This metric represents a good convergence to PF*. Thus, a lower GD value is desirable.

2. Inverse Generational Distance (IGD) (Lu et al. 2017a, b): IGD is the variation of GD, and it is a more comprehensive indicator, reflecting convergence, diversity, uniformly and cardinality simultaneously. It is a measured value defined as the distance between each point in PF* and PF. It is the different measurement with GD and it is shown as:

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^{*2}}}{n} \tag{19}$$

Similar to the in GD, d_i* is the distance between each point consisting of PF* and the nearest points of PF. But n in this equation is the amount of point found in PF*. Therefore, the value of IGD is lower, the diversity, uniformly and the convergence of the

Algorithm: *Non-dominanted-sort* (P)

Input: Population solutions

output: Pareto sorting solutions set

Step1.//Creat the Pareto level 1

For each solution $k \in P$

  $S_k = \varnothing$ //Memory the solutions which are dominated by k

  $n_k = 0$ //The number of solutions which dominate k

  For each solution $u \in P$

    If $u \prec k$

      $S_k = S_k \cup \{u\}$

    End if

    If $k \prec u$

      $n_k = n_k + 1$

    End if

  End for

  If $n_k = 0$//Store Pareto level 1

    $\Gamma_1 = \Gamma_1 \cup \{k\}$

  End if

End for

Step 2.//Fetch the Pareto level 2 from Pareto level 1, and then fetch the Pareto from level 2,and so on.

Step 2.1 $i = 1$

Step 2.2 If $\Gamma_i \neq \varnothing$, then $\Gamma_{i+1} = \varnothing$

  For each solution $k \in \Gamma_i$

    For each solution $u \in S_k$

      $n_u = n_u - 1$

      If $n_u = 0$ $\Gamma_{i+1} = \Gamma_{i+1} \cup \{u\}$

    End for

  End for

  End while $i = i + 1$

  Go back to step2.2

Fig. 10 Pseudocode of the Pareto non-dominated sorting algorithm

solution set is desirable. But in most common situations, *PF** may be unknown. Thus, all solutions in *PF** will be obtained by different MOEAs.

3. Coverage of Two Sets (Qingfu Zhang 2007):

$$C(A,B) = |\{b \in B; \exists h \in A : h \pm b\}|/|T| \tag{20}$$

where $|T|$ represents the number of the solution in set B and $C(A,B)$ equals 1 if any solution of A dominate all solutions of B.

It is assumed that A is obtained by the proposed algorithm, it is obvious that $C(A,B)$ in larger value and $C(B,A)$ in smaller value is preferred. But it must be noted that $C(A,B)$ equal to $1 - C(B,A)$. With the formulated as Eq. (20), $C(A,B)$ and $C(B,A)$ could be

calculated to compare two algorithms related to A and B. Thus, $C(A,B) > C(B,A)$ could imply that algorithm related set A is more effective than the algorithm related set B.

There are various metrics, such as Coverage and DIR (Cai et al. 2018) related to the diversity of obtained solution, HV (Zitzler and Thiele 1999) reflected on convergence and spread. Their calculations are based on reference vector or reference points. With the limited space in this paper, GD, IGD and C are selected to apply in this study. According to the review of indicators of MOP (Li and Yao 2019), the properties could be classified as four types: convergence, spread, uniformity and cardinality. GD is a metric which could reflect the convergence sensitively, and C could reflect the dominance relations between two solution set which related to convergence and cardinality. As a comprehensive metric, IGD could represent the convergence, spread, uniformity and cardinality of solution set simultaneously. Thus, these three indicators could cover all aspects of the properties of solution set.

4.2 Test problems and parameter settings

In this section, an experiment is conducted to evaluate the performance of the MOWSA with these well-known MOEAs including NSGA-II (Deb et al. 2002), SPEA2 (Zitzler and Lothar 2001), MOEA/D (Qingfu Zhang 2007) and MOBSO (Fu et al. 2019). In the current study, there is not any exact study in energy-efficient DWFSP or its benchmarks about it. To demonstrate the effectiveness of the proposed algorithm, we randomly generate 20 instances as shown in Table 1, where $n = \{20,40,60,80,100\}$, $f = \{2,3\}$, $m = \{2,5\}$. And the stop criterion is set as the same CPU time ($T = 40 * 100$ ms). For each combination of n, F, and m, we generate an instance. The standard with processing times is set from the uniform distribution U(30, 50). The basic power, idle power, and welding of power are the same as those in Li et al. (2018). Noted that the normal processing time is the processing time with one machine.

In this section, there are paremeters setting about population size (PS), maximum CPU times (ms), crossover operator probability (α) and mutation operator probability (β). Therefore, the Taguchi method (DOE) (Peng et al. 2019) is conducted to demonstrate the effect of these paremeters with moderate size instance with 60 jobs are

Table 1 Data set distribution

Input variables	Value
Number of jobs (n)	20, 40, 60, 80, 100
Number of factories (f)	2, 3
Number of machines(m)	2, 5
Normal processing time ($np_{i,j}$:, min)	~DU (30,50)
Setup time with adjacent jobs($st_{i,j}$, min)	~DU (1,10)
Machine amount of stage	~DU (2,5)
Basic energy power (P_{basic}, kW)	5
Power during idle (P_{idle}, kW)	0.36
Power during setup (P_{setup}, kW)	10
Power of welding ($P_{welding}$, kW)	28
Welding duty cycle (K_w, %)	80

Table 2 Levels of parameters for MOWSA

Parameter	Parameter level			
	1	2	3	4
PS	20	50	100	200
A	10	20	30	40
α	0.80	0.85	0.90	0.95
β	0.10	0.15	0.20	0.25

assigned to 2/3 factories with 2/5 stages. The details of the parameter setting is formulated as Table 2. According to the comprehensive value of IGD in Table 3 and the Fig. 11, orthogonal array could reflect the influence of the parameters above. According to Fig. 11, the parameter values are set as follows: $PS = 100$, $A = 40$, $\alpha = 10.90$, $\beta = 10.20$.

The pilot experiment is designed to obtain the assessment with different parameter configurations for the proposed algorithms. Thus, to get a relatively fair competitive comparison, all instances apply the uniform population size and CPU times. With the limited space, the population size (*PS*) is set as 100 and topping criterion *A* is set as 40 for all algorithms under pilot experiments. Besides, crossover and mutation operators have the same details in each algorithm. And their probabilities are taken as 0.9 and 0.2, respectively. Each problem is run for 30 replications independently. All algorithms are coded with platform jMetal (Durillo and Nebro 2011) in Java language, and all experimental tests are performed on a computer with Intel Core i7, 2.70 GHz, 8 GB RAM, and Windows 10 operating system.

Table 3 Orthogonal array and average values

Nos	*PS*	*A*	α	β	IGD
1	20	10	0.80	0.10	7.30
2	20	20	0.85	0.15	6.41
3	20	30	0.90	0.20	4.56
4	20	40	0.95	0.25	4.07
5	50	10	0.85	0.20	5.90
6	50	20	0.80	0.25	6.63
7	50	30	0.95	0.10	4.28
8	50	40	0.90	0.15	2.75
9	80	10	0.90	0.25	2.52
10	80	20	0.95	0.20	1.44
11	80	30	0.80	0.15	1.04
12	80	40	0.85	0.10	0.72
13	100	10	0.95	0.15	4.93
14	100	20	0.90	0.10	3.87
15	100	30	0.85	0.25	2.62
16	100	40	0.80	0.20	1.84

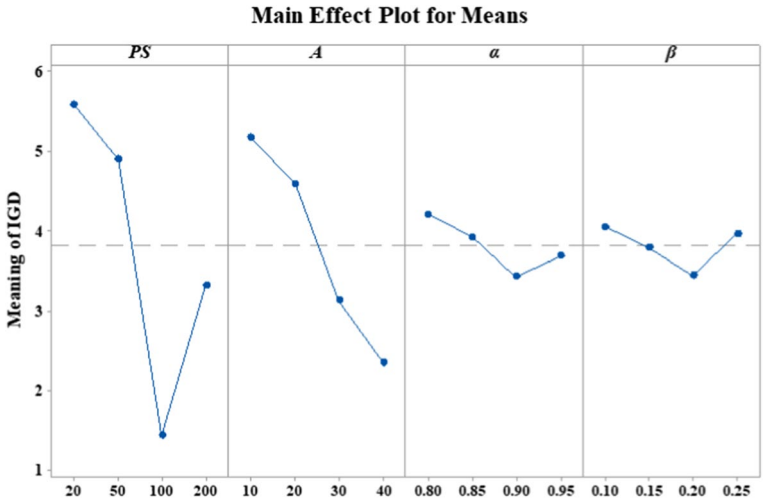


Fig. 11 The trend of the factor level

4.3 Comparison of MOWSA with the other multi-objective optimization algorithms

To construct the fair comparison with other MOEAs, all parameters are set as same in Sect. 4.2 as well as the algorithm with genetic operators. The computational experiment results are shown in Tables 4, 5 and 6 and boxplots with average values are shown in Figs. 12, 13 and 14.

Seen from Table 4 and Fig. 12, it is obvious that MOWSA has lower GD and IGD than NSGA-II, SPEA2, MOEA/D and MOBSO for most instances. And it is could be seen that all values of GD of the MOWSA are smaller than those by NSGA-II, SPEA2, MOEA/D and MOBSO except "80_3_2" and "100_3_2". It implies that the obtained solutions by the MOWSA have better convergence than the other MOEAs in most instances. As for the comprehensive metric IGD in Table 5 and boxplot in Fig. 13, the outperformance of the MOWSA is overwhelming on all test instances except "80_2_2" and "100_2_2", which could be analyzed that great convergence and uniform spread are both obtained with the proposed algorithm. On account of archive set of SPEA2, it will cost much CPU time to maintain and update external archive sets. Thus, the values of GD and IGD of SPEA2 are worse than other MOEAs.

According to Table 6 and Fig. 14 about the metric of Coverage, it can be seen that the MOWSA is better than other MOEAs. From the table, the Coverage metric values in bold font are overwhelmingly distributed in MOWSA which implies that the non-dominated solutions obtained by the MOWSA are better.

Comparing with the other MOEAs, C (MOWSA, NSGAII) is much larger than C (NSGAII, MOWSA) on each instance. And the same situation compared with MOEA/D. Meanwhile, it is obvious that C (MOWSA, SPEA2) is approximately equal to 1 meanwhile C (SPEA2, MOWSA) is approximately equal to 0 on all test instances, which means all non-dominated solutions obtained by SPEA2 are dominated by those obtained

Table 4 GD of MOWSA and other MOEAs

Problem	GD				
	MOWSA	NSGA-II	SPEA2	MOEA/D	MOBSO
20_2_2	**7.45E−04/2.9E−04**	1.21E−03/7.2E−04	6.45E−02/2.9E−02	2.43E−03/9.1E−04	2.21E−03/8.3E−04
20_2_5	**3.95E−03/1.6E−03**	9.56E−03/3.2E−03	1.49E−01/7.2E−02	1.21E−02/3.1E−03	2.38E−02/3.5E−03
20_3_2	**8.87E−04/4.3E−04**	1.42E−03/7.6E−04	6.44E−02/1.9E−02	2.68E−03/1.4E−03	3.61E−03/1.1E−03
20_3_5	**4.72E−03/1.8E−03**	9.35E−03/3.8E−03	1.38E−01/6.1E−02	1.45E−02/4.0E−03	2.86E−02/3.9E−03
40_2_2	**1.36E−03/5.1E−04**	3.39E−03/1.0E−03	6.86E−02/2.2E−02	4.56E−03/1.5E−03	6.88E−03/1.3E−03
40_2_5	**5.52E−03/1.8E−03**	1.40E−02/3.4E−03	1.39E−01/8.0E−02	1.85E−02/3.6E−03	2.30E−02/4.0E−03
40_3_2	**1.53E−03/6.9E−04**	4.45E−03/1.2E−03	7.76E−02/2.1E−02	5.36E−03/1.4E−03	8.89E−03/2.2E−03
40_3_5	**7.39E−03/3.4E−03**	1.87E−02/3.5E−03	1.82E−01/6.2E−02	2.25E−02/4.5E−03	3.04E−02/5.1E−03
60_2_2	**1.75E−03/5.5E−04**	2.31E−03/8.0E−04	6.65E−02/2.3E−02	2.85E−03/9.1E−04	4.96E−03/1.1E−03
60_2_5	**5.11E−03/1.7E−03**	6.19E−03/2.9E−03	8.62E−02/3.3E−02	6.44E−03/2.6E−03	1.89E−02/3.2E−03
60_3_2	**1.86E−03/6.0E−04**	2.22E−03/6.5E−04	5.29E−02/1.7E−02	2.47E−03/9.1E−04	5.71E−03/1.1E−03
60_3_5	**4.97E−03/2.4E−03**	5.98E−03/2.0E−03	7.20E−02/2.8E−02	6.61E−03/2.8E−03	1.66E−02/2.3E−03
80_2_2	**1.77E−03/5.3E−04**	2.37E−03/8.9E−04	5.79E−02/1.8E−02	2.15E−03/9.5E−04	5.60E−03/1.0E−03
80_2_5	**3.80E−03/1.8E−03**	2.39E−02/4.0E−03	1.91E−01/1.0E−01	2.65E−02/4.6E−03	1.16E−02/2.3E−03
80_3_2	2.16E−03/5.8E−04	2.20E−03/8.4E−04	4.91E−02/1.0E−02	**1.70E−03/7.7E−04**	5.65E−03/1.2E−03
80_3_5	**4.29E−03/1.8E−03**	2.43E−02/5.0E−03	1.49E−01/4.6E−02	2.33E−02/4.3E−03	1.27E−02/2.5E−03
100_2_2	**2.95E−03/1.1E−03**	3.29E−03/1.2E−03	6.66E−02/1.8E−02	3.23E−03/1.4E−03	7.86E−03/1.4E−03
100_2_5	**5.83E−03/2.6E−03**	2.08E−02/4.3E−03	1.53E−01/6.3E−02	2.17E−02/4.2E−03	1.77E−02/3.2E−03
100_3_2	**3.82E−03/9.2E−04**	4.89E−03/1.4E−03	7.04E−02/2.0E−02	**3.42E−03/1.5E−03**	9.46E−03/1.3E−03
100_3_5	**5.94E−03/2.6E−03**	2.29E−02/4.7E−03	1.33E−01/4.0E−02	1.89E−02/4.2E−03	1.58E−02/3.4E−03

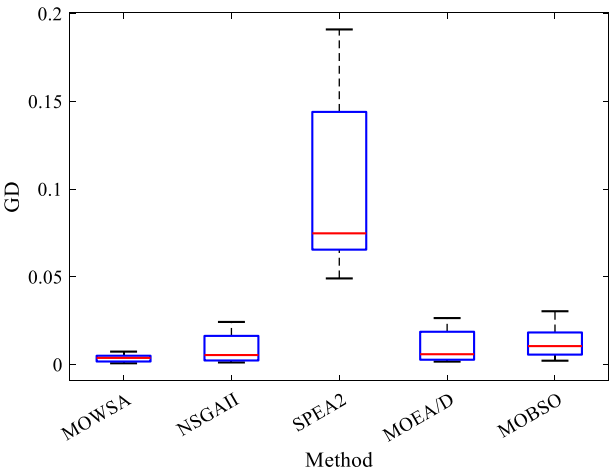


Fig. 12 Boxplot of GD on MOWSA and other MOEAs

by MOWSA. The same situation when compared with MOBSO. Furthermore, Fig. 14 also demonstrates the above conclusion about *C*-metric. According to the comparison result above, MOWSA has great performance with the convergence and spread uniformly and widely.

Table 5 IGD of MOWSA and other MOEAs

Problem	IGD				
	MOWSA	NSGA−II	SPEA2	MOEA/D	MOBSO
20_2_2	**7.30E−04/2.9E−04**	1.55E−03/3.9E−04	1.61E−02/5.8E−03	1.85E−03/3.8E−04	4.30E−03/1.2E−03
20_2_5	**6.29E−03/1.4E−03**	8.97E−03/2.8E−03	4.07E−02/5.8E−03	1.31E−02/2.9E−03	1.36E−02/2.1E−03
20_3_2	**7.83E−04/2.9E−04**	1.07E−03/2.9E−04	1.34E−02/4.2E−03	1.59E−03/4.7E−04	3.95E−03/8.4E−04
20_3_5	**5.35E−03/1.4E−03**	8.86E−03/3.4E−03	4.23E−02/6.7E−03	1.43E−02/3.7E−03	1.58E−02/2.1E−03
40_2_2	**2.34E−03/7.5E−04**	3.01E−03/5.7E−04	2.28E−02/5.9E−03	3.67E−03/7.4E−04	5.56E−03/1.1E−03
40_2_5	**8.64E−03/2.4E−03**	1.42E−02/3.7E−03	6.01E−02/7.1E−03	2.14E−02/5.3E−03	1.60E−02/1.3E−03
40_3_2	**1.76E−03/6.6E−04**	3.40E−03/7.0E−04	2.78E−02/7.8E−03	4.23E−03/9.9E−04	6.55E−03/9.8E−04
40_3_5	**7.29E−03/2.8E−03**	1.57E−02/3.0E−03	6.10E−02/1.3E−02	2.38E−02/5.2E−03	1.73E−02/2.4E−03
60_2_2	**2.27E−03/5.8E−04**	2.68E−03/6.1E−04	3.03E−02/8.5E−03	3.32E−03/7.3E−04	6.06E−03/1.2E−03
60_2_5	**4.65E−03/1.4E−03**	5.48E−03/1.8E−03	3.61E−02/5.6E−03	7.03E−03/1.5E−03	9.70E−03/1.6E−03
60_3_2	**2.60E−03/6.8E−04**	3.26E−03/6.8E−04	3.38E−02/8.9E−03	4.01E−03/1.2E−03	7.12E−03/1.3E−03
60_3_5	**4.59E−03/1.3E−03**	6.25E−03/1.3E−03	3.60E−02/6.5E−03	7.45E−03/2.0E−03	8.94E−03/1.2E−03
80_2_2	2.43E−03/6.9E−04	**2.20E−03/5.4E−04**	2.85E−02/5.7E−03	2.35E−03/5.9E−04	6.37E−03/1.7E−03
80_2_5	**4.78E−03/1.7E−03**	1.97E−02/3.3E−03	5.83E−02/7.1E−03	2.56E−02/4.7E−03	8.08E−03/1.5E−03
80_3_2	**1.52E−03/3.9E−04**	2.04E−03/5.3E−04	2.81E−02/6.8E−03	2.61E−03/1.3E−03	4.13E−03/7.8E−04
80_3_5	**4.29E−03/1.5E−03**	1.84E−02/3.5E−03	5.82E−02/1.0E−02	2.02E−02/3.4E−03	7.88E−03/1.6E−03
100_2_2	4.56E−03/8.8E−04	**4.16E−03/8.4E−04**	3.55E−02/5.9E−03	4.80E−03/1.1E−03	7.62E−03/1.4E−03
100_2_5	**7.09E−03/2.2E−03**	1.68E−02/3.3E−03	6.38E−02/7.5E−03	2.30E−02/4.7E−03	1.23E−02/2.0E−03
100_3_2	**3.78E−03/7.9E−04**	5.41E−03/1.6E−03	4.63E−02/8.2E−03	5.28E−03/2.0E−03	8.25E−03/1.2E−03
100_3_5	**5.34E−03/1.9E−03**	1.54E−02/2.7E−03	6.34E−02/1.2E−02	1.69E−02/3.4E−03	9.27E−03/2.0E−03

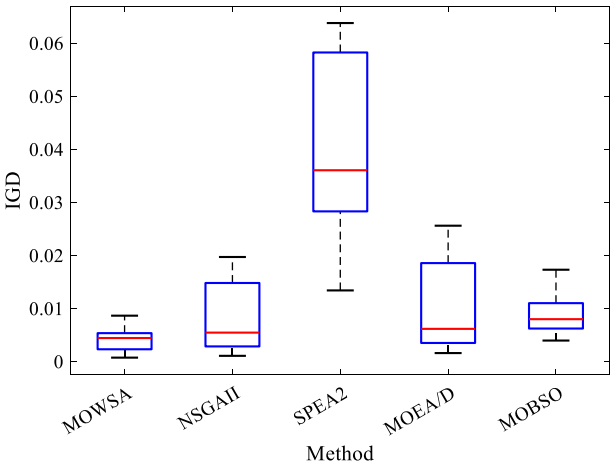


Fig. 13 Boxplot of IGD on MOWSA and other MOEAs

5 Real-life case of study

The proposed method has also been applied to solve a real-life case of the girder welding shop of a crane company in China. There are 2 identical factories to assign 10 jobs with 5 stages.

Table 6 *C*-metric of MOWSA and other MOEAs

Problem	MOWSA versus NSGAII		MOWSA versus SPEA2		MOWSA versus MOEA/D		MOWSA versus MOBSO	
	C (MOWSA, NSGAII)	*C* (NSGAII, MOWSA)	*C* (MOWSA, MOEA/D)	*C* (MOWSA, MOEA/D)	*C* (MOWSA, SPEA2)	*C* (MOWSA, SPEA2)	*C* (MOWSA, MOBSO)	*C* (MOBSO, MOWSA)
20_2_2	**0.60/0.25**	0.08/0.06	**0.99/0.06**	0.00/0.00	**0.90/0.10**	0.02/0.03	**0.87/0.12**	0.01/0.02
20_2_5	**0.86/0.21**	0.07/0.15	**1.00/0.00**	0.00/0.00	**0.86/0.06**	0.01/0.02	**0.98/0.05**	0.00/0.00
20_3_2	**0.63/0.27**	0.09/0.10	**0.97/0.18**	0.00/0.00	**0.87/0.10**	0.03/0.04	**0.93/0.08**	0.00/0.00
20_3_5	**0.73/0.32**	0.15/0.26	**1.00/0.00**	0.00/0.00	**0.99/0.01**	0.00/0.00	**0.97/0.05**	0.00/0.00
40_2_2	**0.78/0.16**	0.06/0.06	**0.99/0.08**	0.00/0.00	**0.91/0.08**	0.03/0.04	**0.98/0.02**	0.00/0.00
40_2_5	**0.92/0.06**	0.02/0.04	**0.97/0.11**	0.00/0.00	**0.91/0.06**	0.01/0.02	**0.98/0.04**	0.00/0.00
40_3_2	**0.88/0.09**	0.03/0.03	**0.96/0.14**	0.00/0.00	**0.93/0.07**	0.03/0.04	**0.99/0.02**	0.00/0.00
40_3_5	**0.93/0.16**	0.03/0.13	**1.00/0.00**	0.00/0.00	**0.98/0.06**	0.01/0.03	**0.97/0.05**	0.00/0.00
60_2_2	**0.64/0.23**	0.18/0.13	**0.97/0.18**	0.00/0.00	**0.78/0.22**	0.15/0.18	**0.95/0.05**	0.00/0.00
60_2_5	**0.51/0.40**	0.35/0.34	**1.00/0.00**	0.00/0.00	**0.63/0.39**	0.26/0.32	**0.97/0.04**	0.00/0.00
60_3_2	**0.57/0.27**	0.23/0.23	**0.98/0.09**	0.00/0.00	**0.67/0.32**	0.19/0.24	**0.97/0.03**	0.00/0.00
60_3_5	**0.55/0.41**	0.31/0.34	**1.00/0.00**	0.00/0.00	**0.51/0.40**	0.33/0.31	**0.95/0.07**	0.02/0.04
80_2_2	**0.68/0.21**	0.17/0.17	**1.00/0.00**	0.00/0.00	**0.66/0.28**	0.18/0.20	**0.97/0.04**	0.00/0.00
80_2_5	**0.99/0.01**	0.00/0.00	**1.00/0.00**	0.00/0.00	**1.00/0.00**	0.00/0.00	**0.89/0.24**	0.07/0.20
80_3_2	**0.47/0.31**	0.32/0.24	**0.89/0.29**	0.00/0.00	**0.46/0.36**	0.33/0.29	**0.96/0.05**	0.00/0.00
80_3_5	**1.00/0.00**	0.00/0.00	**1.00/0.00**	0.00/0.00	**1.00/0.00**	0.00/0.00	**0.89/0.15**	0.04/0.08
100_2_2	**0.48/0.38**	0.33/0.34	**1.00/0.00**	0.00/0.00	**0.53/0.44**	0.36/0.40	**0.96/0.03**	0.01/0.02
100_2_5	**0.99/0.02**	0.00/0.01	**1.00/0.00**	0.00/0.00	**0.99/0.02**	0.00/0.00	**0.93/0.14**	0.03/0.11
100_3_2	**0.61/0.34**	0.22/0.26	**0.98/0.11**	0.00/0.00	**0.99/0.03**	0.00/0.00	**0.98/0.04**	0.00/0.00
100_3_5	**0.98/0.08**	0.08/0.04	**1.00/0.00**	0.00/0.00	**0.41/0.38**	0.40/0.37	**0.83/0.26**	0.10/0.23

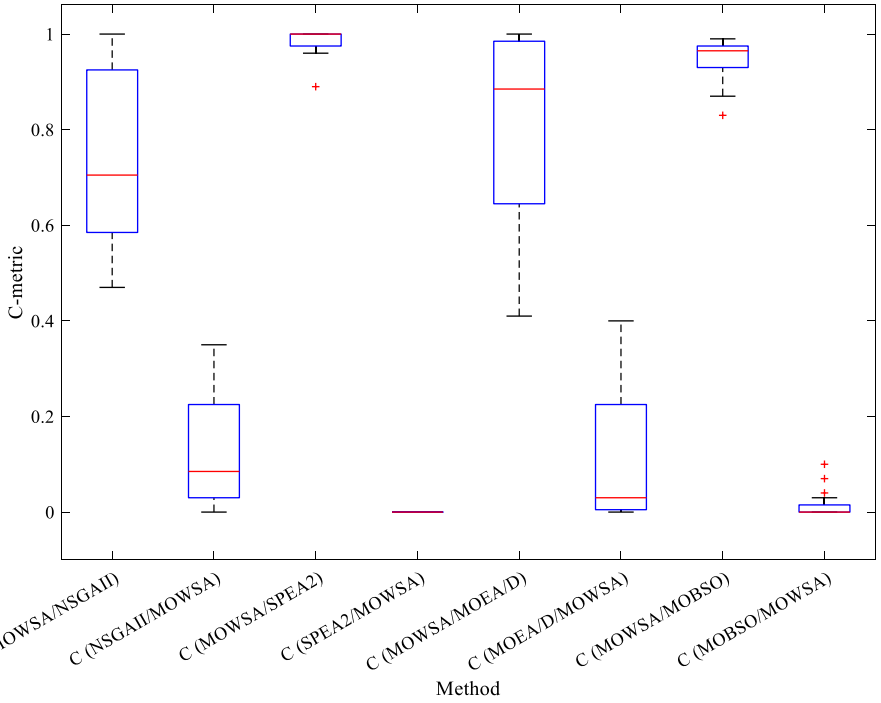


Fig. 14 Boxplot of *C*-metric on MOWSA and other MOEAs

Crane is a kind of lift machine and it is widely used in moving heavy-weight. There is a crane structure of crane in Fig. 15. And the main part is the largest span girder. And the girder's welding procedure is shown in Fig. 16. And it has five main procedures in the girder welding shop.

There are two identical welding shops, and in each stage, there has a fixed amount of machines and other corollary equipment. 10 jobs are assigned to two factories and all jobs should pass through all operations with the same sequence. In the girder's welding procedure, there are five stages in Fig. 16 and Table 7, splices of small pieces, splices of large parts, internal seam weld, cover slab encapsulation and outside seam welding. And the

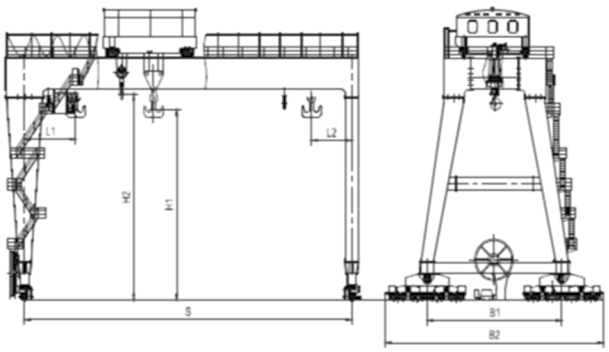


Fig. 15 The structure chart of the crane

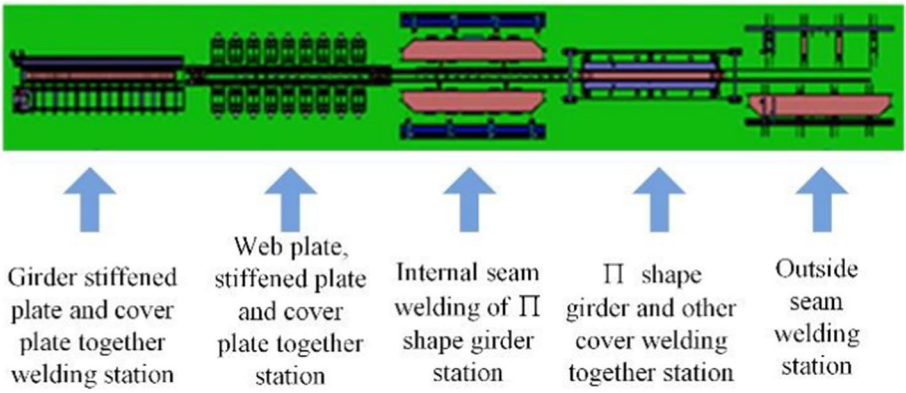


Fig. 16 The girder welding shop layout

Table 7 The available machine in the welding shop

Machine type	Machine	Applicable procedure
TIG welder	1	Splices of small pieces
Semi-auto CO_2 arc shield welder	2	Splices of large parts
Semi-auto CO_2 arc shield welder	2	Internal seam weld
Crane	1	Cover slab encapsulation
Submerged arc automatic welder	2	Assembling and outside seam welding

Table 8 weekly production task in the welding shop

Job type	Span (m)	Assigned job no	Processing/setup time(min)				
			Slicing	Web grouping	Internal seam weld	Encapsulation	Fillet welding
5tA5A6	22.5	1	18/11	32/22	33/10	12/10	25/10
5tA5A6	25.5	2	28/12	38/20	45/15	22/15	28/9
5tA5A6	28.5	3	25/15	50/37	40/14	15/23	45/20
10tA5A6	22.5	4	17/16	23/19	35/12	12/15	28/16
10tA5A6	25.5	5	23/12	36/29	40/13	15/17	34/13
10tA5A6	28.5	6	30/21	45/35	43/15	13/24	37/16
20t16tA5	22.5	7	25/19	23/25	39/15	17/21	32/17
20t16tA5	25.5	8	46/29	59/38	55/24	27/34	58/22
20t16tA5	28.5	9	27/21	38/32	42/18	19/26	39/18
20t16tA6	25.5	10	34/22	49/34	49/16	22/27	42/19

fixed amount of machines are listed in Table 7. In real manufacturing situation, they usually make production plans weekly and the production information is listed in Table 8.

To verify the effectiveness of the proposed MOWSA in solving this problem, the comparison with other MOEAs is conducted in this real-life case. The obtained Pareto

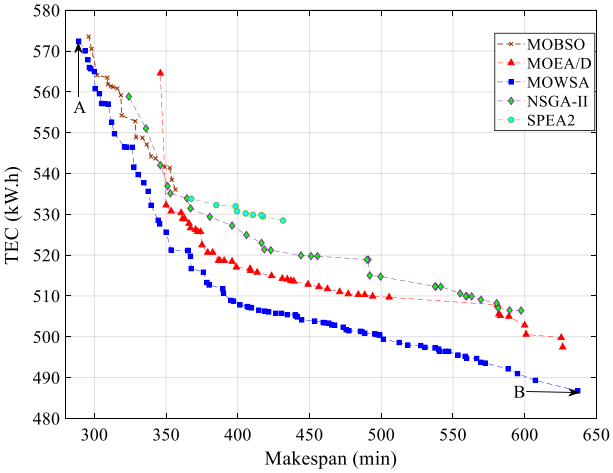


Fig. 17 The Pareto Front of the real-life case

Fig. 18 Solution of point A

$$\begin{array}{rcccccccccc} & J_3 & J_8 & J_{10} & J_7 & J_2 & J_9 & J_4 & J_5 & J_6 & J_1 \\ \pi = [& 3 & 8 & 10 & 7 & 2 & 9 & 4 & 5 & 6 & 1] \end{array}$$

$$\begin{array}{rcccccccccc} & J_1 & J_2 & J_3 & J_4 & J_5 & J_6 & J_7 & J_8 & J_9 & J_{10} \\ a = [& 1 & 2 & 2 & 2 & 1 & 2 & 1 & 1 & 1 & 2] \end{array}$$

$$N = \begin{array}{r} \\ Stage1 \\ Stage2 \\ Stage3 \\ Stage4 \\ Stage5 \end{array} \begin{array}{c} \begin{array}{cccccccccc} J_1 & J_2 & J_3 & J_4 & J_5 & J_6 & J_7 & J_8 & J_9 & J_{10} \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 1 & 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 1 \end{bmatrix} \end{array}$$

Fronts of the real-life case are displayed as shown in Fig. 17. It could be seen that the proposed MOWSA has a better performance in the case. And point A(290.5, 578.5) is the solution shown as Fig. 18 with optimal makespan obtained by MOWSA, and point B is the solution shown as Fig. 19 with optimal TEC obtained by MOWSA. The Gantt chart of point A is shown as Figs. 20 and 21 and the Gantt chart of point B(607.5,489.8) is shown as Figs. 22 and 23. It should be noted that the box with different colors represents the actual processing time and the white bars represent the number of additional machines. The white box denotes the waiting time between the adjacent jobs. Regarding point A with optimal makespan, it has two approximate completion times with 289 in factory 1 and 288.5 in factory 2. And most machines have been utilized in the whole producing procedure. Different from point A, point B has a great difference

Fig. 19 Solution of point B

$$\begin{array}{c} J_8\ J_{10}\ J_2\ J_9\ J_3\ J_4\ J_7\ J_6\ J_5\ J_1 \\ \pi=[8\ \ 10\ \ 2\ \ 9\ \ 3\ \ 4\ \ 7\ \ 6\ \ 5\ \ 1] \end{array}$$

$$\begin{array}{c} J_1\ J_2\ J_3\ J_4\ J_5\ J_6\ J_7\ J_8\ J_9\ J_{10} \\ a=[1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 2\ ] \end{array}$$

$$N = \begin{array}{c} \\ Stage1 \\ Stage2 \\ Stage3 \\ Stage4 \\ Stage5 \end{array} \begin{array}{c} \begin{array}{cccccccccc} J_1 & J_2 & J_3 & J_4 & J_5 & J_6 & J_7 & J_8 & J_9 & J_{10} \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 2 \end{bmatrix} \end{array}$$

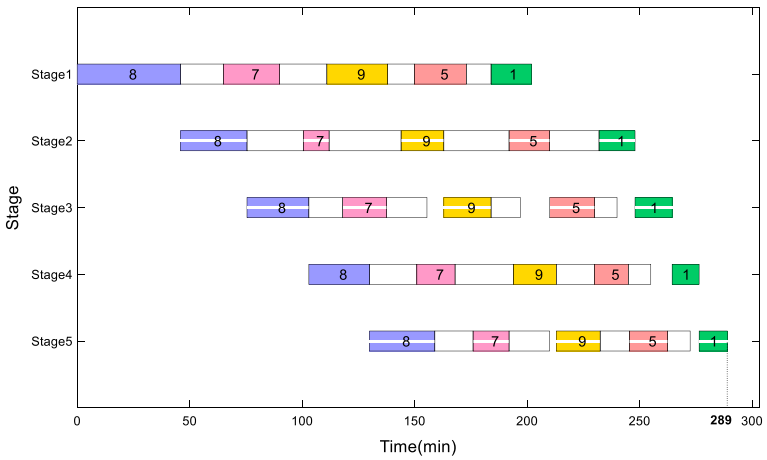


Fig. 20 The Gantt chart of point A in factory 1

with makespan value with 637 and 126 obtained by two factories. But few operations use more than one machine and it will result in low TEC.

6 Conclusions and future work

This work addresses an energy-efficient DWFSP with setup time which is a specific case in the manufacturing environment. This problem involves the adjusted amount of machine of operation which could influence the objectives of TEC and makespan.

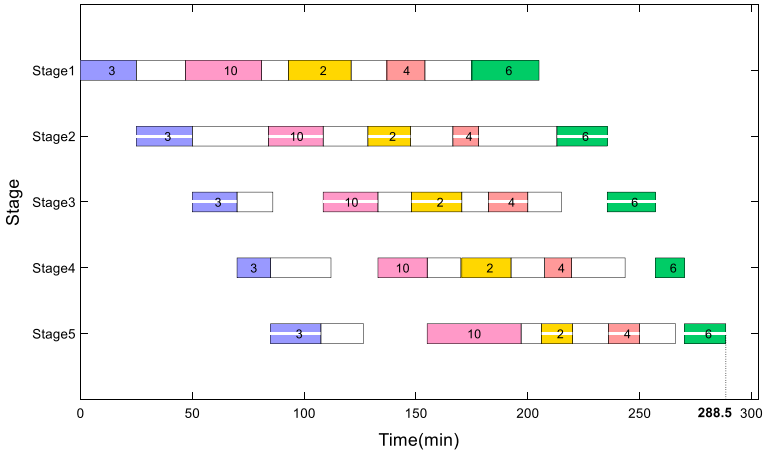


Fig. 21 The Gantt chart of point A in factory 2

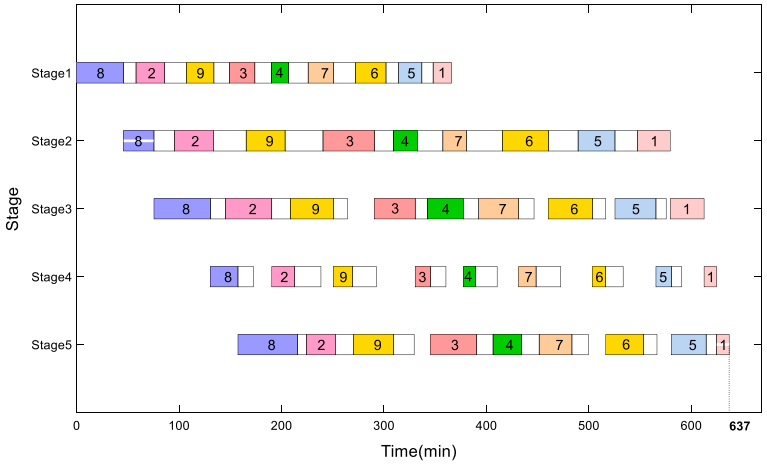


Fig. 22 The Gantt chart of point B in factory 1

In this study, we propose a new multi-objective mathematical model and modified MOWSA to optimize the TEC and makespan simultaneously. To enhance the quality of the solution, various local search operators are designed according to the feature of the problem. Furthermore, we launch the experiment to compare the proposed MOWSA with well-known MOEAs: NSGA-II, SPEA2, MOEA/D and MOBSO. With the experiment instances, the proposed MOWSA has enormous effectiveness in terms of the GD, IGD, and *C*-metrics. At last, the proposed algorithm is applied to address the real-life DWFSP with a great performance compared with other MOEAs.

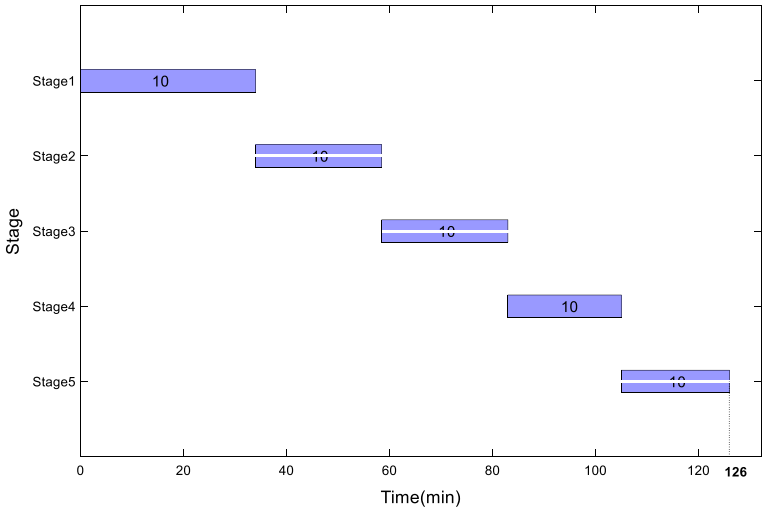


Fig. 23 The Gantt chart of point B in factory 2

Concerning future work, we plan to consider more objectives such as noise pollution and tardiness in distributed scheduling problem. And the transportation time could be taken in to account into distributed scheduling problem.

Acknowledgments This work was supported by National Natural Science Foundation for Distinguished Young Scholars of China (Grant No. 51825502), National Natural Science Foundation of China (Grant No. 51775216), Natural Science Foundation of Hubei Province (Grant No. 2018CFA078) and Program for HUST Academic Frontier Youth Team (Grant No. 2017QYTD04).

References

Cai, X., Sun, H., & Fan, Z. (2018). A diversity indicator based on reference vectors for many-objective optimization. *Information Sciences, 430–431,* 467–486.

Chan, F. T. S., Prakash, A., Ma, H. L., & Wong, C. S. (2013). A hybrid Tabu sample-sort simulated annealing approach for solving distributed scheduling problem. *International Journal of Production Research, 51*(9), 2602–2619.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197.

Ding, J.-Y., Song, S., & Wu, C. (2016). Carbon-efficient scheduling of flow shops by multi-objective optimization. *European Journal of Operational Research, 248*(3), 758–771.

Fernandez-Viagas, V., & Framinan, J. (2014). A bounded-search iterated greedy algorithm for the distributed permutation flowshop scheduling problem. *International Journal of Production Research, 53,* 1111–1123.

Fernandez-Viagas, V., Perez-Gonzalez, P., & Framinan, J. M. (2018). The distributed permutation flow shop to minimise the total flowtime. *Computers and Industrial Engineering, 118,* 464–477.

Fu, Y., Tian, G., Fathollahi-Fard, A. M., Ahmadi, A., & Zhang, C. (2019). Stochastic multi-objective modelling and optimization of an energy-conscious distributed permutation flow shop scheduling problem with the total tardiness constraint. *Journal of Cleaner Production, 226,* 515–525.

Gao, H., Kwong, S., Fan, B., & Wang, R. (2014). A hybrid particle-swarm tabu search algorithm for solving job shop scheduling problems. *IEEE Transactions on Industrial Informatics, 10*(4), 2044–2054.

Gao, J., & Chen, R. (2011). A hybrid genetic algorithm for the distributed permutation flowshop scheduling problem. *International Journal of Computational Intelligence Systems, 4*(4), 497–508.

Gao, J., Chen, R., & Deng, W. (2013). An efficient tabu search algorithm for the distributed permutation flowshop scheduling problem. *International Journal of Production Research, 51*(3), 641–651.

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of operations research, 2*(1), 117–129.

Goldberg, D. E. & Lingle, R. (1985). Alleles, loci, and the traveling salesman problem. In: Proceedings of an international conference on genetic algorithms and their applications (Vol. 154, pp. 154–159): Lawrence Erlbaum, Hillsdale, NJ.

Grobler, J., Engelbrecht, A. P., Kok, S., & Yadavalli, S. (2010). Metaheuristics for the multi-objective FJSP with sequence-dependent set-up times, auxiliary resources and machine down time. *Annals of Operations Research, 180*(1), 165–196.

Hansen, P., Mladenovic, N., & Perez, J. A. M. (2010). Variable neighbourhood search: methods and applications. *Annals of Operations Research, 175*(1), 367–407.

Kahn, K. B., Castellion, G., & Griffin, A. (2005). *The PDMA handbook of new product development*. Hoboken, NJ: Wiley.

Li, M., & Yao, X. (2019). Quality evaluation of solution sets in multiobjective optimisation: a survey. *ACM Computing Surveys, 52*(2), 26.

Li, X., Gao, L., Pan, Q., Wan, L., & Chao, K. (2019a). An effective hybrid genetic algorithm and variable neighborhood search for integrated process planning and scheduling in a packaging machine workshop. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 49*(10), 1933–1944.

Li, X., Lu, C., Gao, L., Xiao, S., & Wen, L. (2018). An effective multiobjective algorithm for energy-efficient scheduling in a real-life welding shop. *IEEE Transactions on Industrial Informatics, 14*(12), 5400–5409.

Li, X., Xiao, S., Wang, C., & Yi, J. (2019b). Mathematical modeling and a discrete artificial bee colony algorithm for the welding shop scheduling problem. *Memetic Computing, 11,* 1–19.

Lin, S., Ying, K., & Huang, C. (2013). Minimising makespan in distributed permutation flowshops using a modified iterated greedy algorithm. *International Journal of Production Research, 51*(16), 5029–5038.

Lu, C., Gao, L., Li, X., Pan, Q., & Wang, Q. (2017a). Energy-efficient permutation flow shop scheduling problem using a hybrid multi-objective backtracking search algorithm. *Journal of Cleaner Production, 144,* 228–238.

Lu, C., Gao, L., Li, X., & Xiao, S. (2017b). A hybrid multi-objective grey wolf optimizer for dynamic scheduling in a real-world welding industry. *Engineering Applications of Artificial Intelligence, 57,* 61–79.

Lu, C., Gao, L., Li, X., Zheng, J., & Gong, W. (2018). A multi-objective approach to welding shop scheduling for makespan, noise pollution and energy consumption. *Journal of Cleaner Production, 196,* 773–787.

Lu, C., Xiao, S., Li, X., & Gao, L. (2016). An effective multi-objective discrete grey wolf optimizer for a real-world scheduling problem in welding production. *Advances in Engineering Software, 99,* 161–176.

Marichelvam, M. K., & Prabaharan, T. (2015). Solving realistic industrial scheduling problems using a multi-objective improved hybrid particle swarm optimisation algorithm. *International Journal of Operational Research, 23*(1), 94–129.

Naderi, B., & Ruiz, R. (2010). The distributed permutation flowshop scheduling problem. *Computers and Operations Research, 37*(4), 754–768.

Naderi, B., & Ruiz, R. (2014). A scatter search algorithm for the distributed permutation flowshop scheduling problem. *European Journal of Operational Research, 239*(2), 323–334.

Nanda, S. J., & Panda, G. (2014). A survey on nature inspired metaheuristic algorithms for partitional clustering. *Swarm and Evolutionary Computation, 16,* 1–18.

Pan, Q., Gao, L., Wang, L., Liang, J., & Li, X. (2019). Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem. *Expert Systems with Applications, 124,* 309–324.

Pei, J., Cheng, B. Y., Liu, X. B., Pardalos, P. M., & Kong, M. (2019). Single-machine and parallel-machine serial-batching scheduling problems with position-based learning effect and linear setup time. *Annals of Operations Research, 272*(1–2), 217–241.

Peng, K., Pan, Q.-K., Gao, L., Li, X., Das, S., & Zhang, B. (2019). A multi-start variable neighbourhood descent algorithm for hybrid flowshop rescheduling. *Swarm and Evolutionary Computation, 45,* 92–112.

Qingfu Zhang, H. L. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation, 11*(6), 712–731.

Ruiz, R., Pan, Q.-K. & Naderi, B. (2018). Iterated Greedy methods for the distributed permutation flowshop scheduling problem. Omega 18: 213-222.

Shao, W., Pi, D., & Shao, Z. (2017). Optimization of makespan for the distributed no-wait flow shop scheduling problem with iterated greedy algorithms. *Knowledge-Based Systems, 137,* 163–181.

Shrivastava, A., Krones, M., & Pfefferkorn, F. E. (2015). Comparison of energy consumption and environmental impact of friction stir welding and gas metal arc welding for aluminum. *CIRP Journal of Manufacturing Science and Technology, 9,* 159–168.

Si, L., Pan, Y., & Yang, Q. (2010). The current situation and development of arc welding energy. *Electric Welder, 40*(6), 108–132.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research, 64*(2), 425–434.

Wang, G., Gao, L., Li, X., Li, P., & Tasgetiren, M. F. (2020). Energy-efficient distributed permutation flow shop scheduling problem using a multi-objective whale swarm algorithm. *Swarm and Evolutionary Computation, 57,* 100716.

Wang, G., Li, X., Gao, L., & Li, P. (2019). A multi-objective whale swarm algorithm for energy-efficient distributed permutation flow shop scheduling problem with sequence dependent setup times. *IFAC-PapersOnLine, 52*(13), 235–240.

Wang, S., Wang, L., Liu, M., & Xu, Y. (2013). An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem. *International Journal of Production Economics, 145*(1), 387–396.

Zeng, B., Li, X., Gao, L., Zhang, Y., & Dong, H. (2019). Whale swarm algorithm with the mechanism of identifying and escaping from extreme points for multimodal function optimization. *Neural Computing and Applications, 32,* 1–21.

Zhang, G., Xing, K., & Cao, F. (2018). Discrete differential evolution algorithm for distributed blocking flowshop scheduling with makespan criterion. *Engineering Applications of Artificial Intelligence, 76,* 96–107.

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation, 3*(4), 257–271.

Zitzler E. M. L., & Lothar, T. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm. TIK-report, 103.