sciendo

JAISCR, 2024, Vol. 14, No. 2, pp. 101 – 120
10.2478/jaiscr-2024-0006

A SHUFFLED FROG LEAPING ALGORITHM WITH Q-LEARNING FOR DISTRIBUTED HYBRID FLOW SHOP SCHEDULING PROBLEM WITH ENERGY-SAVING

Jingcao Cai[1,2,*] and Lei Wang[1,*]

[1]*School of Mechanical Engineering, Anhui Polytechnic University, Wuhu 241000, China*

[2]*AnHui Key Laboratory of Detection Technology and Energy Saving Devices, AnHui Polytechnic University, Wuhu 241000, China*

[*]*E-mail:caijingcao@foxmail.com, wangdalei2000@126.com*

Submitted: 11th December 2023; Accepted: 27th February 2024

Abstract

Energy saving has always been a concern in production scheduling, especially in distributed hybrid flow shop scheduling problems. This study proposes a shuffled frog leaping algorithm with Q-learning (QSFLA) to solve distributed hybrid flow shop scheduling problems with energy-saving(DEHFSP) for minimizing the maximum completion time and total energy consumption simultaneously. The mathematical model is provided, and the lower bounds of two optimization objectives are given and proved. A Q-learning process is embedded in the memeplex search of QSFLA. The state of the population is calculated based on the lower bound. Sixteen search strategy combinations are designed according to the four kinds of global search and four kinds of neighborhood structure. One combination is selected to be used in the memeplex search according to the population state. An energy-saving operator is presented to reduce total energy consumption without increasing the processing time. One hundred forty instances with different scales are tested, and the computational results show that QSFLA is a very competitive algorithm for solving DEHFSP.

Keywords: energy-saving; distributed scheduling; hybrid flow shop; shuffled frog-leaping algorithm; reinforcement learning

1 Introduction

Energy has always been a concern of human society. Using fossil fuels will cause air pollution, and the energy shortage will restrict industrial production and even the country's economic development. Studying energy-saving scheduling in manufacturing is necessary to protect the environment better and reduce energy waste.

The hybrid flow shop scheduling problem(HFSP) is a typical shop scheduling problem in the manufacturing process. It widely exists in many real-life manufacturing industries such as steel industries, paper industries, textile industries, semiconductor manufacturing, electronics industries, chemical industry, automobile manufacturing, etc. HFSP with energy-saving has received extensive attention from researchers. Lei et al. [13] addressed HFSP with the minimization of total en-

ergy consumption and total tardiness. Lu et al. [19] considered noise pollution, production efficiency, and energy consumption. Zhang et al. [32] addressed HFSP by minimizing the maximum completion time and total energy consumption. Qin et al. [24] solved HFSP with blocking to minimize the maximum completion time and energy efficiency. Unlike the maximum completion time index related to production efficiency, energy consumption is seldom optimized as a sole optimization objective, as small energy consumption and pollution are allowed, and the enterprise needs to survive. Therefore, in the study of HFSP, energy-saving indicators are often optimized along with production efficiency and other indicators.

With economic globalization and manufacturing globalization, a distributed manufacturing model has gradually formed to reduce manufacturing costs, energy consumption, and management risks, thereby improving enterprises' market competitiveness. The common distributed scheduling problems include distributed parallel scheduling problems, distributed job shop scheduling problems, distributed flexible job shop scheduling problems, distributed flow shop scheduling problems, distributed hybrid flow shop scheduling problems (DHFSP), distributed assembly scheduling problems, etc. The distributed shop scheduling problem with energy-saving had been given enough attention. Pan et al. [22] solved distributed parallel scheduling problems with energy-efficient. Jiang et al. [11] addressed distributed job shop scheduling with energy-efficient. Du et al. [8] solved distributed flexible job shop scheduling with crane transportation to minimize the maximum completion time and energy consumption. Wang et al. [26] studied energy-efficient scheduling of distributed welding flow shop. Cai et al. [1] addressed DHFSP with fuzzy processing time.

DHFSP is an extension of HFSP, and the current research mainly focuses on minimizing the maximum completion time related to production efficiency. Ying et al. [31] presented a mixed integer linear programming formulation and self-tuning iterated greedy (SIG) algorithm that incorporates an adaptive cocktail decoding mechanism to solve DHFSP with multiprocessor tasks. Cai et al. [2] proposed a dynamic shuffled frog-leaping algorithm to address the same problem as [31] and provide and prove a lower bound. Hao et al. [10] and Li et al. [15] developed the hybrid brainstorm optimization algorithm. Li et al. [17, 16, 18] designed the artificial bee colony algorithm. Wang et al. [28] proposed a bi-population cooperative memetic algorithm. Cai et al. [5] presented a novel shuffled frog-leaping algorithm with reinforcement learning for DHFSP with assembly. Meng et al. [20] formulated three novel mixed-integer linear programming models and a constraint programming model for DHFSP with sequence-dependent setup times. Only Qin et al. [23] considered optimizing the minimization of the sum of earliness, tardiness, and delivery costs in the single-objective optimization study of DHFSP and proposed an adaptive human-learning-based genetic algorithm. Thus, given the requirements of actual production scheduling, researchers are collectively dedicated to studying methods that enhance production capacity.

As regards the multi-objective DHFSP, existing studies usually take the maximum completion time as one of the optimization objectives. Lei et al. [14] proposed a shuffled frog leaping algorithm with memeplex grouping to minimize the maximum completion time and the number of tardy jobs. Cai et al. [4] presented a shuffled frog-leaping algorithm with memeplex quality to simultaneously minimize total tardiness and the maximum completion time. Cai et al. [3] developed a collaborative variable search to optimize fuzzy maximum completion time and total agreement index simultaneously. Wang et al. [30] and Cai et al. [1] proposed two different shuffled frog leaping algorithms to optimize fuzzy maximum completion time, total agreement index, and fuzzy total energy consumption simultaneously. Jiang et al. [12] presented a novel multi-objective evolutionary algorithm based on decomposition to minimize maximum completion time and total energy consumption. [25] provided a multi-objective evolutionary algorithm based on multiple neighborhoods' local search for minimizing maximum completion time, total weighted earliness and tardiness, and total workload. Wang et al. [29] developed a cooperative memetic algorithm with a reinforcement learning-based policy agent for minimizing the maximum completion time and energy consumption. Zheng et al. [33] proposed a cooperative coevolution algorithm with problem-specific strategies to optimize the fuzzy total tardiness and robustness simultane-

ously without considering the optimization object of the maximum completion time.

Regarded as DHFSP with energy saving, Wang et al. [30] and Cai et al. [1] studied energy-saving DHFSP with three objectives but did not consider the problem of a deterministic environment. Jiang et al. [12] addressed the multi-objective energy-saving DHFSP in the deterministic environment. However, the processing speed of machines is fixed in this problem, and it is not considered to reduce energy consumption by reducing the processing speed of machines. Although Wang et al. [29] addressed the multi-objective energy-saving DHFSP and reduced energy consumption by reducing the processing speed of machines, the energy consumption during the period from machine opening to job processing is not considered. As stated above, most existing studies on DHFSP, whether a single objective optimization problem or multi-objective optimization problem, focused on improving production efficiency. Moreover, energy consumption is generally not optimized as an independent objective but often along with the production efficiency objective as another optimization objective. Since energy saving and consumption reduction need to be based on the survival of enterprises, it is meaningful to consider both energy saving and production efficiency. Given the necessity of studying energy consumption scheduling and filling the research gap, this paper studies the distributed hybrid flow shop scheduling problem with energy saving(DEHFSP).

DHFSP is the NP-hard problem, and as the extension of DHFSP, DEHFSP is also NP-hard. Due to the complexity, DHFSP is almost impossible to be solved by exact algorithms and obtain exact solutions in an acceptable time [20]. The intelligent optimization algorithm is the primary method to solve DHFSP, especially multi-objective DHFSP, such as shuffled frog leaping algorithm(SFLA)[1], variable neighborhood search algorithm[3], evolutionary algorithm based on decomposition[12], evolutionary algorithm based on multiple neighborhoods local search[25], cooperative memetic algorithm[29], and cooperative coevolution algorithm[33]. Because of the shortage of exact algorithms to solve large-scale distributed shop scheduling problems and the successful application of intelligent optimization algorithms, especially SFLA, in DHFSP, we use SFLA to solve DEHFSP.

SFLA is an intelligent optimization algorithm that mimics the process of frogs searching for food [9]. SFLA has a good search framework, and its optimization ability is strong with fewer parameters, and it has been successfully applied to distributed shop scheduling[1, 21]. SFLA, like other intelligent optimization algorithms, has been proven to have a good search framework when it was proposed. However, the search strategy, or the operator, must be designed according to the characteristics of the problem in order to avoid the deterioration of the algorithm's performance. The operator, such as crossover operator, mutation operator, global search operator, or local search operator, to generate new solutions to explore the solution space is critically essential for intelligent optimization algorithms. Nevertheless, in recent research, the selection and use of these critical operators are still based on experiment or experience. Consequently, it is necessary to adopt an effective new method to make the selection of effective operators more intelligent in many complicated operations.

Q-learning is a reinforcement learning that can dynamically select an action in a state and can be used to choose operators intelligently. The combination of Q-learning and the intelligent optimization algorithm to improve the algorithm's performance has attracted some attention. Chen et al. [6] used the Q-learning algorithm to intelligently adjust the key parameters of the genetic algorithm to address the flexible job shop scheduling problem. It was an exciting exploration and worked well. Wang et al. [27] implemented a Q-learning algorithm to dynamically select search operators for solving distributed three-stage assembly scheduling problems. Cai et al. [5] embedded Q-learning algorithm in QSFLA to select a search strategy dynamically for memeplex search to solve a distributed assembly hybrid flow shop scheduling problem. However, in current research, the fusion of Q-learning and the intelligent optimization algorithm is only used to solve single-objective shop scheduling problems. For the multi-objective optimization problem, the evaluation of the population and the design of action need to explore. Therefore, combining Q-learning with intelligent optimization algorithms to solve multi-objective optimization problems is highly essential.

In this study, DEHFSP is considered. The main contributions are summarised as follows. (1) DEHFSP with energy saving is described, and its mathematical model is provided. (2) The lower bounds of the maximum completion and total energy consumption of DEHFSP are given and proved. (3) A variety of neighborhood search strategies and global search strategies are designed, and an energy-saving operator is presented for some unique solutions. (4) The Q-learning algorithm is embedded in SFLA to dynamically chose a search strategy in memeplex search. (5) Several experiments are conducted, and the computational results show that Q-learning is effective and efficient and QSFLA is a very competitive method for DEHFSP.

The remainder of the paper is organized as follows. The problem description is formulated in Section 2. Section 3 gives and proves the lower bound. The basic SFLA and Q-learning is introduced in Section 4. Section 5 provides the detailed designs of the proposed algorithm. Section 6 gives the computational experiments and analyses the results. Finally, we end the paper with some conclusions and future work in Section 7.

2 Problem description

Distributed energy-saving hybrid flow shop scheduling problem is described as follows. There are n jobs $J_1, J_2, \ldots, J_n$ need to be processed in F homogeneous factories. Each factory can be regarded as a hybrid flow shop. In the f-th factory, there are S stages, each of which contains m_{fs} identical parallel machines. M_{fsl} is the l-th machine at the s-th stage in the f-th factory. v_{is} is the processing speed of J_i on M_{fsl}. p_{is} represents the processing time of J_i on M_{fsl} when $v_{is} = 1$. The actual processing time of J_i on M_{fsl} is p_{is}/v_{is}.

A machine has two modes, including processing mode and standby mode, and both of theses modes incur energy consumption. E_{is} is defined as the energy consumption per unit time of J_i processed on M_{fsl} at v_{is}. $E_{is} = \xi_{is} \cdot v_{is}^2$, where ξ_{is} is the coefficient of energy consumption. SE is the energy consumption per unit time of M_{fsl} in standby mode. It's evident that the faster the processing speed, the higher the energy consumption of the machine. Table 1 gives notations and descriptions.

Table 1. Notations and descriptions

Notation	Description
i, f, s, l	Indexes
n	The number of jobs
J_i	The i-th job
F	The number of factories
S	The number of stages
m_{fs}	The number of parallel machines at stage s of factory f
M_{fsl}	The l-th machine at stage s of factory f
V	A one-dimensional vector which represents the set of selectable process speed of machines
v_{is}	The processing speed of J_i in stage s and $v_{is} \in V$
p_{is}	The processing time of J_i on M_{fsl} when $v_{is} = 1$
E_{is}	The energy consumption per unit time of J_i processed on M_{fsl} at v_{is}
SE	The energy consumption per unit time of M_{fsl} in standby mode
C_i	The completion time of J_i
C_{max}	The maximum completion time of all jobs
TEC	The total energy consumption
Φ_{fsl}	The set of all jobs processed on M_{fsl}
U	A very large positive number
X_{if}	If J_i is processed in the f-th factory, $X_{if} = 1$; otherwise $X_{if} = 0$
Y_{ifsl}	If J_i is processed in the M_{fsl}, $Y_{fsl} = 1$; otherwise $Y_{fsl} = 0$
$Z_{ii'fs}$	If J_i is processed before $J_{i'}$ at the l-th stage in the f-th factory, $Z_{ii'fs} = 1$; otherwise $Z_{ii'fs} = 0$
st_{is}	The start time of process of J_i at the l-th stage
et_{is}	The end time of process of J_i at the l-th stage

DEHFSP aims to allocate jobs to machines in each stage in different factories. Then decide the processing sequence of jobs in each machine and select an appropriate speed for each job on each machine to minimize maximum completion time and total energy consumption simultaneously. The mathematical model of DEHFSP is as follows:

$$min\ C_{\max} = \max_{i\in\{1,2,\ldots,n\}} \{C_i\} \tag{1}$$

$$min\ TEC = \sum_{f=1}^{F}\sum_{s=1}^{S}\sum_{l=1}^{m_{fs}}\Big(\sum_{J_i\in\Phi_{fsl}} p_{is}/v_{is}\cdot(E_{is}-SE)+SE\cdot \max_{J_i\in\Phi_{fsl}} C_i\Big) \tag{2}$$

$$E_{is} = \xi_{is} \times v_{is}^2 \tag{3}$$

$$\sum_{f=1}^{F} X_{if} = 1, \forall i \tag{4}$$

$$\sum_{l=1}^{m_{fs}} Y_{ifsl} = X_{if}, \forall i, f, s \tag{5}$$

$$st_{i1} \geq 0, \forall i \tag{6}$$

$$st_{i(s+1)} \geq et_{is}, \forall i, s \tag{7}$$

$$et_{is} = st_{is} + \sum_{f=1}^{F}\sum_{s=1}^{S}\sum_{l=1}^{m_{fs}}(p_{is}/v_{is}\cdot X_{if}\times Y_{ifsl}), \forall i \quad (8)$$

$$Z_{ii'fs} + Z_{i'ifs} \le 1, \forall f,s,i,i' \quad (9)$$

$$Z_{ii'fs} + Z_{i'ifs} \ge Y_{ifsl} + Y_{i'fsl} - 1, \forall f,s,i' > i \quad (10)$$

$$st_{i'l} \ge et_{il} - U\times(3 - Y_{ifsl} - Y_{i'fsl} - Z_{ii'fs}),$$
$$\forall i \ne i', f,s,l \in \{1,2,...,m_{fs}\} \quad (11)$$

$$X_{if} \in \{0,1\}, \forall i,f \quad (12)$$

$$Y_{ifsl} \in \{0,1\}, \forall i,f,s,l \in \{1,2,...,m_{fs}\} \quad (13)$$

$$Z_{ii'fs} \in \{0,1\}, \forall i,i',f,s \quad (14)$$

where equation (1) is to minimize maximum completion time; equation (2) is to minimize total energy consumption; equation (3) provides the method to calculate energy consumption; the constraint (4) shows that each job only can be processed in one factory; constraint (5) demonstrates that each job only can be assigned to one machine at each stage in each factory; constraint (6) indicates that each job can be processed after zero time; constraint (7) shows that the start time of process at stage $s+1$ is not earlier than the end time of process at stage s; constraint (8) demonstrates that the process cannot be interrupted; constraints (9)-(11) indicate that each machine can only process one job at one time; constraints (12)-(14) provide the binary decision variables.

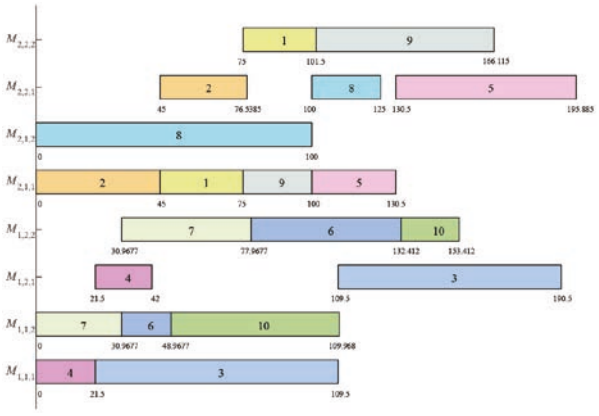


Figure 1. Gantt Chart of the case

An example is provided to illustrate the problem better. In a distributed hybrid flow shop, ten jobs need to be processed in two factories, each factory has two process stages, and each stage contains two parallel machines. The detailed processing information is shown in Table 2. Gantt Chart of this case is shown in Fig.1.

As shown in Fig.1, the completing time of J_5 is 195.885, and J_5 is the last finished job; as a result, the maximum complete time C_{max}=195.885 based on equation (1). The unit energy consumption of each machine at different time points is shown in Figure 2, and the total energy consumption TEC = 5702.83 can be calculated by equations (2)-(3).

3 Lower bound

The lower bound is an effective method to evaluate the quality of scheduling, and it is also the basis of evaluating population state in multi-objective optimization. For DEHFSP, there are two optimization objectives, C_{max} and TEC, and their lower bound are given and proved below.

Theorem 1. Lower bound on C_{max} of DEHFSP is defined by

$$LB_{c_{\max}} = \max_{s\in\{1,2,...,S\}}\left\{\frac{1}{F}\left(\Psi_s^1 + \frac{1}{m_{fs}}\cdot\sum_{i=1}^{n}\frac{p_{is}}{\max\{V\}} + \Psi_s^2\right)\right\} \quad (15)$$

where $\psi_{is}^1 = \sum_{s'=1}^{s-1} p_{is'}/v_{is'}$, $\psi_{is}^2 = \sum_{s'=s+1}^{S} p_{is'}/v_{is'}$, sort all ψ_{is}^r in the ascending order and suppose that $\psi_{1s}^r \le \psi_{2s}^r \le \cdots \le \psi_{ns}^r$, then $\Psi_s^r = \sum_{f=1}^{F}\psi_{fs}^r, r = 1,2.$

Proof. Cai et al. [2] provide the lower bound on C_{max} of DHFSP with multiprocessor tasks and the distinction of DEHFSP and DHFSP with multiprocessor tasks is that a job can only be processed by one machine in DEHFSP. The lower bound on C_{max} of DEHFSP can be obtained when the number of processing machines required for jobs is adjusted to 1.

Theorem 2. Lower bound on TEC of DEHFSP is defined by

$$LB_{TEC} = \min_{v_{is}\in V}\{\sum_{s=1}^{S}\sum_{i=1}^{n} p_{is}\cdot\xi_{is}\cdot v_{is}$$
$$+\frac{1}{F}\sum_{s=1}^{S}\min_{i\in\{1,2,...,n\}}\{\sum_{s'=1}^{s}\{p_{is'}/v_{is'}\cdot m_{fs}\cdot SE\}\}\} \quad (16)$$

Proof. According to Equation (2),

$$TEC = \sum_{s=1}^{S}\sum_{i=1}^{n} p_{is}/v_{is}\cdot E_{is} + \Delta_1 + \Delta_2, \quad (17)$$

Table 2. The processing information of a case

n	F	S	m_{f1}	m_{f2}	ξ_{is}	SE	V
10	2	2	2	2	3	1	$\{1.00, 1.30, 1.55, 1.80, 2.00\}$

$p_{1,1}$	$p_{2,1}$	$p_{3,1}$	$p_{4,1}$	$p_{5,1}$	$p_{6,1}$	$p_{7,1}$	$p_{8,1}$	$p_{9,1}$	$p_{10,1}$	$p_{1,2}$	$p_{2,2}$	$p_{3,2}$	$p_{4,2}$	$p_{5,2}$	$p_{6,2}$	$p_{7,2}$	$p_{8,2}$	$p_{9,2}$	$p_{10,2}$
39	81	88	43	61	36	48	100	45	61	53	41	81	41	85	98	47	45	84	42
$v_{1,1}$	$v_{2,1}$	$v_{3,1}$	$v_{4,1}$	$v_{5,1}$	$v_{6,1}$	$v_{7,1}$	$v_{8,1}$	$v_{9,1}$	$v_{10,1}$	$v_{1,2}$	$v_{2,2}$	$v_{3,2}$	$v_{4,2}$	$v_{5,2}$	$v_{6,2}$	$v_{7,2}$	$v_{8,2}$	$v_{9,2}$	$v_{10,2}$
1.30	1.80	1.00	2.00	2.00	2.00	1.55	1.00	1.80	1.00	2.00	1.30	1.00	2.00	1.30	1.80	1.00	1.80	1.30	2.00

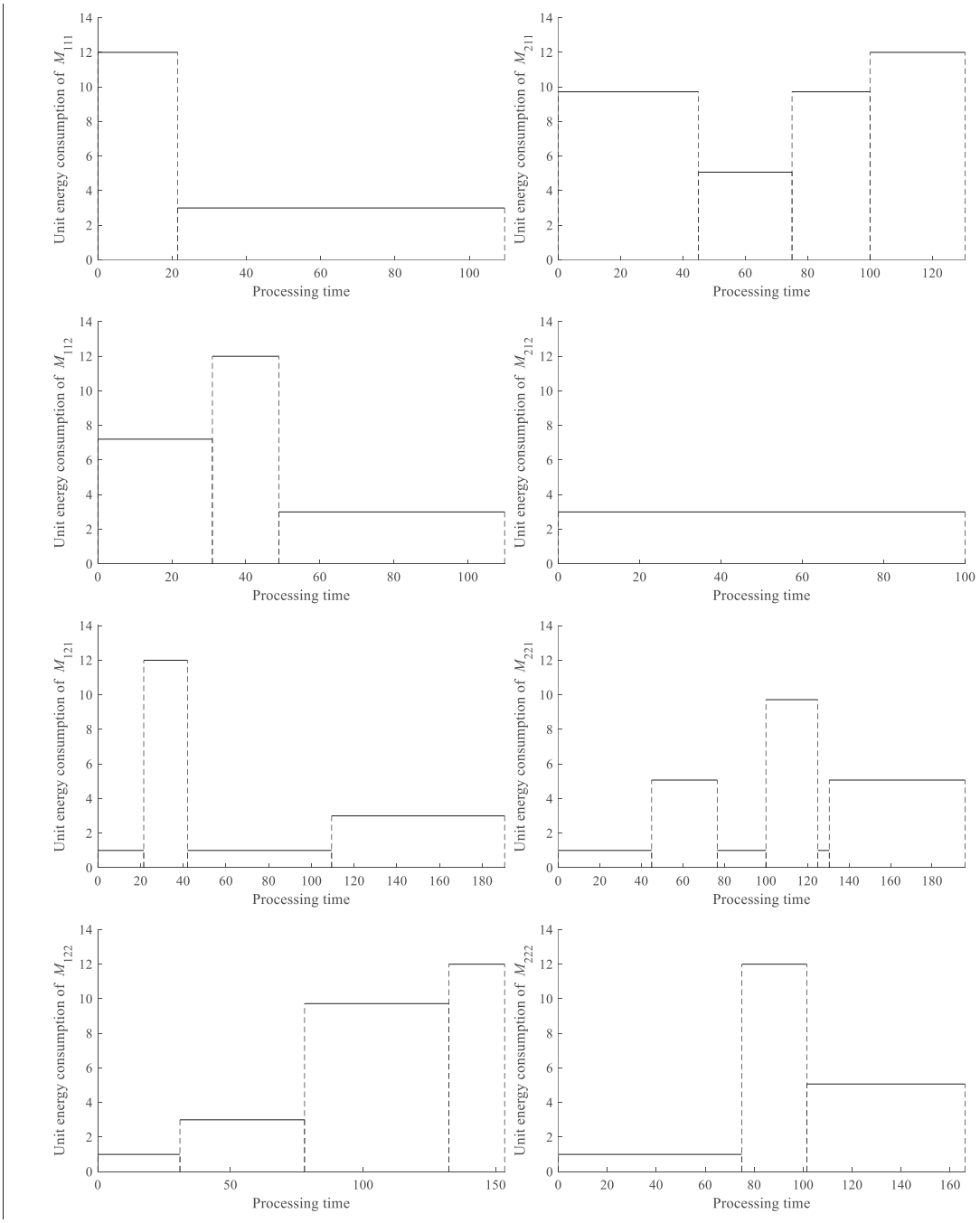


Figure 2. Energy consumption of each machine

where Δ_1 is the energy consumption of all machines from the start of an operation to the start of processing jobs in standby mode, and Δ_2 represents the other consumption of all machines in standby mode.

It is obviously,

$$\sum_{s=1}^{S}\sum_{i=1}^{n} p_{is}/v_{is}\cdot E_{is}+\Delta_1 \le TEC. \tag{18}$$

According to constraints (7-8),

$$\begin{aligned}\Delta_1 &= \sum_{f=1}^{F}\sum_{s=1}^{S}\sum_{l=1}^{m_{fs}}\min_{Y_{ifsl}=1}\{st_{is}\}\cdot SE\\ &= \sum_{f=1}^{F}\sum_{s=1}^{S}(\min_{Y_{ifs1}=1}\{st_{is}\}+\min_{Y_{ifs2}=1}\{st_{is}\}+\cdots\\ &+\min_{Y_{ifsm_{fs}}=1}\{st_{is}\})\cdot SE\\ &\ge \sum_{f=1}^{F}\sum_{s=1}^{S}\min_{i\in\{1,2,\ldots,n\}}\{st_{is}\cdot X_{if}\cdot m_{fs}\}\cdot SE\end{aligned} \tag{19}$$

According to constraints (7-8),

$$st_{is}\ge\sum_{s'=1}^{s-1} p_{is'}/v_{is'}. \tag{20}$$

According to equations (19-20),

$$\begin{aligned}\Delta_1 &\ge \sum_{f=1}^{F}\sum_{s=1}^{S}\min_{i\in\{1,2,\ldots,n\}}\{\sum_{s'=1}^{s}\{p_{is'}/v_{is'}\cdot X_{if}\cdot m_{fs}\cdot SE\}\}\\ &\ge \frac{1}{F}\sum_{s=1}^{S}\min_{i\in\{1,2,\ldots,n\}}\{\sum_{s'=1}^{s}\{p_{is'}/v_{is'}\cdot m_{fs}\cdot SE\}\}\end{aligned} \tag{21}$$

According to equations (18) and (21),

$$\begin{aligned}&\sum_{s=1}^{S}\sum_{i=1}^{n} p_{is}/v_{is}\cdot E_{is}\\ &+\frac{1}{F}\sum_{s=1}^{S}\min_{i\in\{1,2,\ldots,n\}}\{\sum_{s'=1}^{s}\{p_{is'}/v_{is'}\cdot m_{fs}\cdot SE\}\}\le TEC\end{aligned} \tag{22}$$

Since $v_{is}\in V$,

$$\begin{aligned}&\min_{v_{is}\in V}\{\sum_{s=1}^{S}\sum_{i=1}^{n} p_{is}\cdot\xi_{is}\cdot v_{is}\\ &+\frac{1}{F}\sum_{s=1}^{S}\min_{i\in\{1,2,\ldots,n\}}\{\sum_{s'=1}^{s}\{p_{is'}/v_{is'}\cdot m_{fs}\cdot SE\}\}\le TEC\end{aligned} \tag{23}$$

Thus, LB_{TEC} is proved.

In particular, it is not the lower the processing speed of jobs, the smaller the total energy consumption. The first impression is that the processing energy of jobs is reduced, then the total energy consumption is also reduced. However, in the real processing environment, from the first job processing, all machines are turned on until all jobs are finished. When the processing speed of the jobs decreases, the processing energy consumption will decrease, while the idle energy consumption Δ_1 will increase. Therefore, if the total energy consumption is to be minimized, the processing speed of all jobs needs to be coordinated, rather than minimized for all jobs.

4 Introduction to SFLA and Q-learning

4.1 Introduction to SFLA

The main steps of SFLA contain population initialization, population division, memeplex search, and population shuffling. The population P includes N solutions, each of which is defined as the position of a virtual frog. All solutions are initialized, and the initial population is obtained. Then all solutions are sorted in the descending order of fitness and put into Ps memeplexes in population division. In memeplex search, μ searches are executed for each memeplex, and the pseudo-code of memeplex search is shown in **Algorithm 1**. After all memeplexes execute the search, they are shuffled, and a new population is formed. Repeat population division, memeplex search, and population shuffling until the terminal condition is satisfied. The pseudo-

code of SFLA is shown in **Algorithm 2**.

Algorithm 1 Memeplex search of SFLA

for $i = 1$ to Ps **do**
  for $j = 1$ to μ **do**
    $x_{new} = x_w + rand \cdot (x_b - x_w)$;
    if x_{new} is better than x_w **then**
      $x_w = x_{new}$;
      continue;
    end if
    $x_{new} = x_w + rand(x_g - x_w)$;
    if x_{new} is better than x_w **then**
      $x_w = x_{new}$;
      continue;
    end if
    randomly generate a new solution x_{new};
    $x_w = x_{new}$;
  end for
end for

Algorithm 2 SFLA

Population initialization;
while terminal condition is not satisfied **do**
  Population division;
  Memeplex search;
  population shuffling;
end while
Output optimization result;

Algorithm 3 Q-learning

Initialize Q-table;
while terminal condition is not satisfied **do**
  Evaluate the state s_t of environment;
  Select an action a_t according to Q-table by $\varepsilon - greed$;
  Update Q-table by

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot (r_{t+1} + \gamma \max_a (s_{t+1}, a) - Q(s_t, a_t)); \quad (24)$$

end while
Output Q-table;

4.2 Introduction to Q-learning

Q-learning is a model-free reinforcement learning algorithm which is a machine learning method. Q-learning includes four parts, $\mathcal{S}$, $\mathcal{A}$, $\mathcal{R}$ and $\mathcal{P}$, where S is the state space of environment, $\mathcal{A}$ is the action space, $\mathcal{R}$ is the reward function, and $\mathcal{P}$ is the state transition probability.

In Q-learning, action is chosen according to the state of the environment and Q-table. After the selected action is executed, Q-table is updated based on the reward of the current action and the new state of the environment. The process of Q-learning is shown in **Algorithm 3**.

5 A shuffled frog leaping algorithm with Q-learning

5.1 Coding and decoding

DEHFSP includes four sub-problem: factory assignment, machine assignment, sequence assignment, and speed assignment. Cai and his colleagues used a method that determines machine assignment at the decoding stage to reduce sub-problems. This method has been testified to be an effective one, and we adopt the same method to solve DEHFSP.

An encoding of DEHFSP is represented as a factory string $[\theta_1, \theta_2, ..., \theta_n]$, a sequence string $[\pi_1, \pi_2, ..., \pi_n]$ and a speed string $[\omega_{1,1}, \omega_{2,1}, ..., \omega_{n,1}, ..., \omega_{1,S}, \omega_{2,S}, ..., \omega_{n,S}]$, where $\theta_i \in \{1, 2, ..., F\}$, $\pi_i \in \{1, 2, ..., n\}$ and $\pi_i \neq \pi_j$ $(\forall i \neq j)$, $\omega_{i,j} \in \{1, 2..., |V|\}$ and V is the set of machine processing speeds and $V_{\omega_{i,j}}$ is the processing speed of J_j in stage i.

The decoding procedure is described as follows. Assign all jobs to factories according to the factory string, where J_i is assigned to factory θ_i. In each factory, the permutation of all jobs is decided by sequence string, and for two jobs J_{π_i} and J_{π_j}, if $i < j$, assign J_{π_i} to machines preferentially. When selecting a machine for a job, first determine the set of machines that can be selected, then choose a machine that can minimize the completion time of this job. The processing speed of jobs is determined by the speed string, and J_i is processed on M_{fsl} with the speed of $V_{\omega_{is}}$. If the completion time of this job in some machines is the same, select the machine with the smaller number. For example, if machine M_{111}

and machine M_{112} meet the requirements, machine M_{111} is selected. What needs illustration is that jobs are assigned to machines at the same priority at all stages.

To further explain the decoding process, a possible solution for a case in **Section 2** is given, which consists of a factory string [2,2,1,1,2,1,1,2,2,1], a sequence string [2,8,4,7,1,3,9,6,10,5] and a speed string [2,4,1,5,5,5,3,1,4,1;5,2,1,5,2,4,1,4,2,5]. According to the factory string, J_3,J_4,J_6,J_7 and J_{10} are assigned to factory 1, and their sequence is J_4,J_7,J_3,J_6 and J_{10} determined by the sequence string. J_4 is first arranged to a machine, and M_{111} and M_{112} can be chosen. No matter which machine is selected, the completion time of J_4 can be minimized, and M_{111} is selected because of its smaller number. Then J_4 and J_3 are assigned to M_{111} and J_7,J_6,J_{10} are assigned to M_{112}. In M_{111}, the processing speed of J_4 and J_3 is $\omega_{1,4} = 5$ and $\omega_{1,3} = 1$ respectively, so the processing time of J_4 and J_3 are $p_{41}/V_{\omega_{1,4}} = 21.5$ and $p_{31}/V_{\omega_{1,3}} = 88$. In the same way, the scheduling scheme can be obtained, and the Gantt chart is shown in Figure 1, then $C_{max} = 195.885$. In M_{111}, the energy consumption of M_{111} equals to $p_{41}/V_{\omega_{1,4}} \times \xi_{41} \times V^2_{\omega_{1,4}} + p_{31}/V_{\omega_{1,3}} \times \xi_{31} \times V^2_{\omega_{1,3}} = 43/2 \times 3 \times 2^2 + 88/1 \times 3 \times 1^2 = 258 + 264 = 522$. Figure 2 provides the energy consumption of each machine. After calculating the energy consumption of all machines, the total energy consumption can be available and $TEC = 5702.83$.

5.2 Global search

Global search is an indispensable and effective method to generate new solutions in swarm intelligence optimization algorithms. In SFLA, the representation of global search is that one solution learns from and moves to another solution to produce a new solution. **Algorithm 1** gives the global search equation of SFLA in continuous space. Since the solution space of DEHFSP is discrete, it is necessary to design a global search for solutions for DEHFSP.

A global search $GS(x,y)$ for DEHFSP is designed, in which x,y represent two different solutions, x is the optimized one, and y is the learned one. x learns from y to get a new solution x', and it is expected that x can inherits the part of advantages of y. The global search is described as follows: (1) randomly determine a set Π of jobs; (2) for the factory string, the value of elements corresponding to jobs of Π in x is changed to that in y; (3) for the sequence string, rearrange the order of jobs in x belong to Π to match the order of this jobs in y; (4) for the speed string, the value in x corresponding to jobs in Π is adjusted to be the same as that in y. Figure 3 provides the process of global search. It can be found from Figure 3 that the new solution x' varies greatly compared with x.

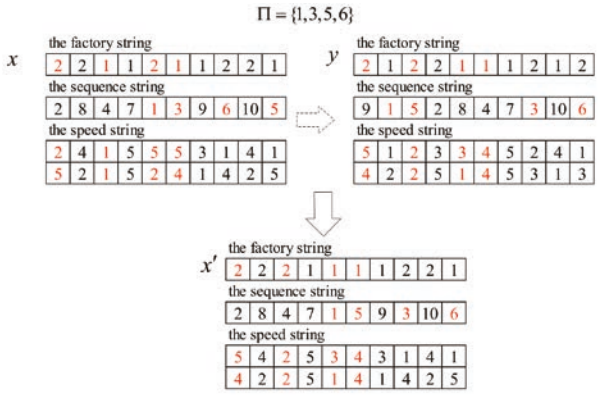


Figure 3. The process of global search

5.3 Local search

Local search is the operator that changes the optimized solution slightly and includes insert and swap commonly. Inserting or swapping jobs between different factories or within the same factory can produce a new solution whose objectives differ from the original solution.

An insert operator $\mathcal{N}_1$ and a swap operator $\mathcal{N}_2$ are designed to change the factory string and the sequence string. In $\mathcal{N}_1$, randomly choose jobs J_i and J_j, $i < j$, and suppose $\pi_{pos_1} = i, \pi_{pos_2} = j$, then insert $\pi_{pos_2} = j$ to position of π_{pos_1} in the sequence string and let $\theta_j = \theta_i$. In $\mathcal{N}_2$, randomly choose jobs J_i and J_j, $i < j$, and suppose $\pi_{pos_1} = i, \pi_{pos_2} = j$, then swap the value π_{pos_1} and π_{pos_2} in the sequence string and swap the value of θ_j and θ_i in the factory string.

In $\mathcal{N}_1$ and $\mathcal{N}_2$, the speed string does not change. When the processing speed of jobs increases, the energy consumption will increase, and the completion time will decrease; on the contrary, when the processing speed of jobs decreases, the energy consumption will decrease, and the completion time will increase. Based on the above characteristics

of DEHFSP, $\mathcal{N}_3$ and $\mathcal{N}_4$ are designed to adjust the speed string.

$\mathcal{N}_3$ is shown below. Randomly determine a set Π of jobs and a randomly number $k, k \in \{1,2,...,S\}$. For each $J_i \in \Pi$, $\omega_{i,k} = \max\{\omega_{i,k} - 1, 1\}$. $\mathcal{N}_4$ is similar with $\mathcal{N}_3$, expect that $\omega_{i,k} = \min\{\omega_{i,k} + 1, |V|\}$.

Figure 4 depicts the process of generating new solutions in $\mathcal{N}_1$, $\mathcal{N}_2$, $\mathcal{N}_3$ and $\mathcal{N}_4$.

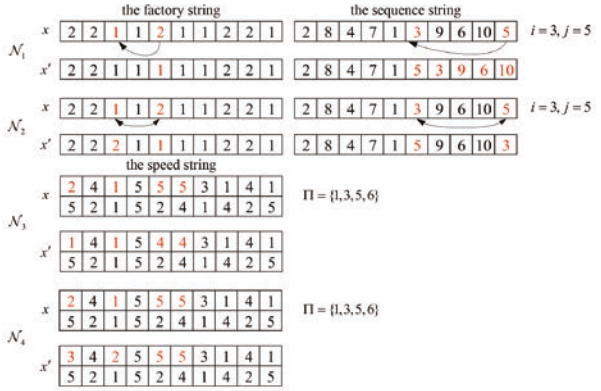


Figure 4. The process of local search

5.4 Energy saving operator

Energy conservation is significant for reducing carbon emissions and environmental protection; however, it is meaningless to consider energy saving only and ignore the production efficiency index such as C_{max}, which is related to the production efficiency of enterprises. Based on this, an energy-saving operator is designed which can reduce TEC without changing C_{max}.

The energy-saving operator is described below. For each job in each stage, the processing speed of the job is reduced to the minimum value if meets the following two conditions: (1) the start time of the next job, which is processed on the machine where the current job is located, is not changed; (2) the start time of the current job in next stage is not changed.

In order to better explain the energy-saving operator, an example is given, and the solution is given in **Section 5.1**. As can be seen from the Gantt chart corresponding to this solution, J_4 processed on M_{121} satisfies conditions of energy saving operator, and the processing speed of J_4 in stage 2 is adjusted from 2.00 to 1.00. The Gantt chart before and after adjustment is shown in Figure 5. From Figure 5, it can be found that C_{max} has not increased, and TEC has been reduced from 5702.83 to 5559.33.

5.5 Q-learning process

Q-learning algorithm includes state set s_t, action set a_t, reward function, and action selection strategy. The environment state is determined by the evaluation of the population, and actions are represented by different search strategies; a new reward function is designed, and the action selection strategy uses the $\varepsilon - greedy$ strategy.

Population P is a set of solutions, which include N solutions, and the evaluation of population is the key to using Q-learning. The evaluation value ev_i of the generation t of P is defined as

$$ev_t = \sum_{i=1}^{N} \left(\frac{C_{\max}^{t,i} - LB_{C_{\max}}}{\overline{C}_{\max} - LB_{C_{\max}}} + \frac{TEC^{t,i} - LB_{TEC}}{\overline{TEC} - LB_{TEC}}\right)/N \tag{25}$$

where $C_{\max}^{t,i}$ and $TEC^{t,i}$ represent the maximum completion time and total energy consumption of the ith solution of population P in the tth generation respectively and $\overline{C}_{\max} = \max_{i=\{1,2...,N\}} \{C_{\max}^{1,i}\}$, $\overline{TEC} = \max_{i=\{1,2...,N\}} \{TEC^{1,i}\}$.

ev_t is basically in the interval $[ev^1, ev^2]$ and let $ev^1 = 0, ev^2 = 2$. The state set $S = \{1,2,...,10\}$. Divide the interval into 10 equal parts, and the state value $s_t = k$ if $ev_t \in [0.2 \times (k-1), 0.2 \times k)$, $0 \le k \le 10$ and $a_t = 10$ if $ev_t \ge 2$.

Table 3. The corresponding relationship between action and search method

Action	search method	Action	search method
1	GS_1+$\mathcal{N}_1$	9	GS_3+$\mathcal{N}_1$
2	GS_1+$\mathcal{N}_2$	10	GS_3+$\mathcal{N}_2$
3	GS_1+$\mathcal{N}_3$	11	GS_3+$\mathcal{N}_3$
4	GS_1+$\mathcal{N}_4$	12	GS_3+$\mathcal{N}_4$
5	GS_2+$\mathcal{N}_1$	13	GS_4+$\mathcal{N}_1$
6	GS_2+$\mathcal{N}_2$	14	GS_4+$\mathcal{N}_2$
7	GS_2+$\mathcal{N}_3$	15	GS_4+$\mathcal{N}_3$
8	GS_2+$\mathcal{N}_4$	16	GS_4+$\mathcal{N}_4$

The variants of GS are built, which are GS_1, GS_2 and GS_3. GS_1, GS_2, and GS_3 are only changing one string, and only the factory string is changed in GS_1, only the sequence string is altered in GS_2, and only the speed string is modified in GS_3. The action set A is constructed by using different global searches and local searches. There are 4 global searches and 4 lo-

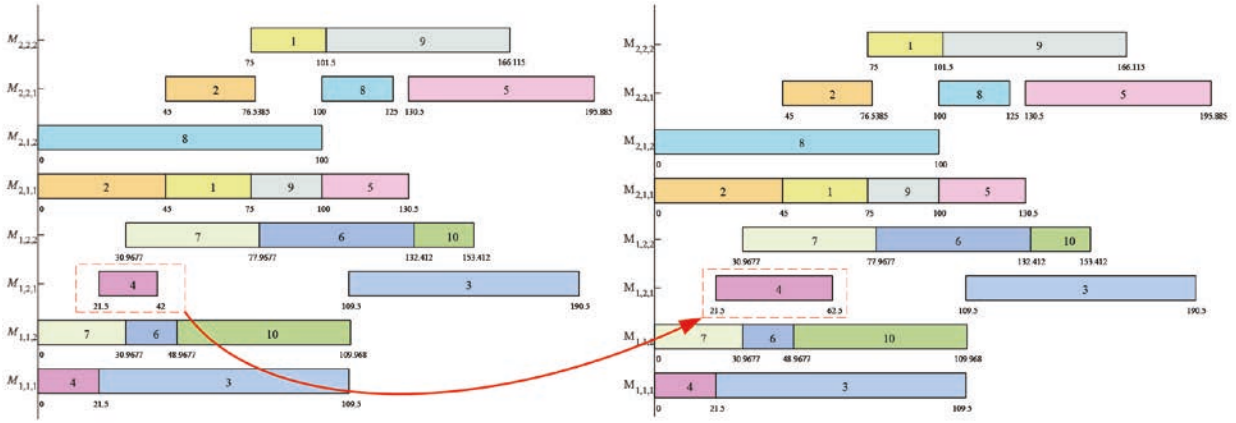


Figure 5. The Gantt chart before and after adjustment

cal searches, so there are 16 actions corresponding to 16 combinations. The corresponding relationship is shown in Table 3.

Action 1 is described as follows: GS_1 is implemented for x_w and x_b to generate x_{new}, and if x_w dominated x_{new}, GS_1 is executed for x_w and x_g to produce x_{new}, and if x_w dominated x_{new}, $\mathcal{N}_1$ is applied to obtain x_{new}, then $x_w = x_{new}$. The steps of Action 2-16 are similar to Action 1, but use different global search and local searches.

Since the s_t is smaller, the population is closer to the Pareto front in the solution space, let $r_{t+1} = s_{t+1} - s_t$. Obviously, the positive gain is obtained when using action a_t in state a_t to improve the state of the population, and vice versa, the penalty is obtained.

$\varepsilon - greed$ is used directly to select an action, and Equation (24) is used to update Q-table. Generally, all elements of the initial Q-table are 0.

5.6 Algorithm description

QSFLA is an algorithm that combines SFLA and Q-learning, and Q-learning is used in the process of memeplexes search to choose suitable global and local search. The main steps of QSFLA include population initialization and Q-table initialization, population division, memeplexes search with Q-learning, Q-table update, energy saving operator, population shuffling, and output results. Figure 6 shows the flow chart of QSFLA.

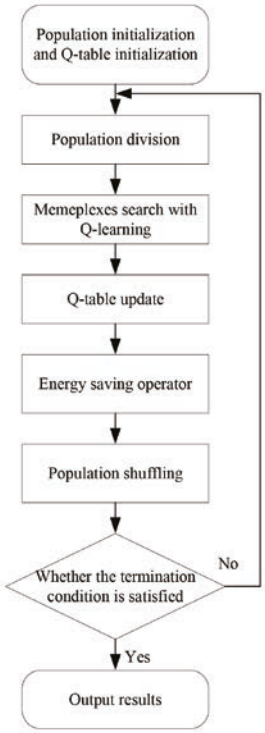


Figure 6. The flow chart of QSFLA

The initial population is randomly generated. The initial Q-table is initialized to 0. In population division, all solutions in the population are ordered according to the non-dominated sort rule in NSGA-II. Then these solutions are sequentially assigned to each memeplex according to the rule to **Section 4.1**. In memeplexes search, the state of the population is first determined, and the combination of global search and local search corresponding to the action is selected according to $\varepsilon - greed$. Each of the memeplexes performs μ searches. After the

memeplexes search, all solutions from all memeplexes first execute an energy saving operator and then are shuffled to form a new population. The result is output when the termination condition is satisfied.

6 Computational experiments

All experiments are programmed using Visual Studio 2022 C++ and run on a computer with 16.0G RAM 12th Gen Intel(R) Core(TM) i7-12700H 2.70GHz.

6.1 Test instances

To test the performance of the proposed algorithm, 140 instances are provided, which consist of the different numbers of jobs, factories, and stages, and these instances can download from https://gitee.com/caijingcao/modhfsp114. $n \times F \times S$ represents the instance where there are n jobs, F factories and S stages.

6.2 Comparative algorithms

In order to verify whether Q-learning plays an active role in QSFLA, 16 variants of QSFLA were designed, each of which only uses one action. QSFLA$_v$ indicates the vth variant, which only uses action v. For example, the combination of $GS_1 + \mathcal{N}_1$ is used in QSFLA$_1$. Another major strategy of QSFLA is energy saving operator. QSFLA^{E-} is given to test the effectiveness of energy saving operator. In addition, NSGA-II is selected as a comparative algorithm, which is a classical multi-objective optimization algorithm and has been successfully used to solve a series of the multi-objective optimization problem. Therefore, a total of 18 comparative algorithms are selected to verify the performance of QSFLA from different perspectives.

6.3 Performance evaluation metrics

The performance evaluation of multi-objective algorithms mainly includes convergence, uniformity, and spread.

Metric *GD* is used to calculate the convergence index, which can evaluate the distance between the solution set and Pareto optimal frontier. Metric *GD* is defined as

$$GD(\Omega_A, \Omega^*) = \frac{\sqrt{\sum_{y \in \Omega_A} \min_{x \in \Omega^*} dis(x,y)^2}}{|\Omega_A|} \tag{26}$$

where Ω_A is the solution set obtained by algorithm A, Ω^* denotes the conference solution set, $dis(x,y)^2$ represents the Euclidean distance between solution x and y in the objective space.

Metric *Spacing* is used for evaluating the uniformity index, which is calculated as

$$Spacing(\Omega_A) = \sqrt{\frac{1}{|\Omega_A|} \sum_{i=1}^{|\Omega_A|} (\overline{d} - d_i)^2} \tag{27}$$

where d_i represents the minimum Euclidean distance between x_i with other solution in Ω_A and $\overline{d}$ is the mean value of all d_i.

Metric Δ_{Ω_A} is estimated for spread index, which is defined as

$$\Delta(\Omega_A) = \frac{d_f + d_l + \sum_{i=1}^{|\Omega_A|} \left| d_i - \overline{d} \right|}{d_f + d_l + |\Omega_A| \cdot \overline{d}} \tag{28}$$

where d_f and d_l represent the minimum Euclidean distance between the two boundary solutions of Ω^* and Ω_A, respectively.

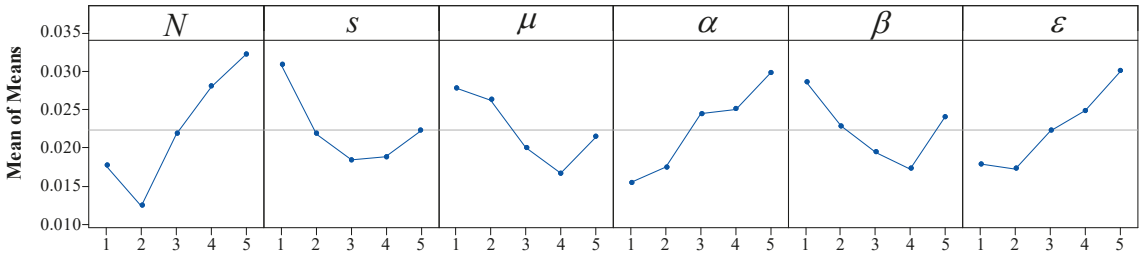
Table 4. Parameters and their levels

Parameters	Factor level				
	1	2	3	4	5
N	30	60	90	120	150
s	2	3	5	6	10
μ	30	40	50	60	70
α	0.1	0.2	0.3	0.4	0.5
γ	0.6	0.7	0.8	0.9	1.0
ε	0.1	0.2	0.3	0.4	0.5

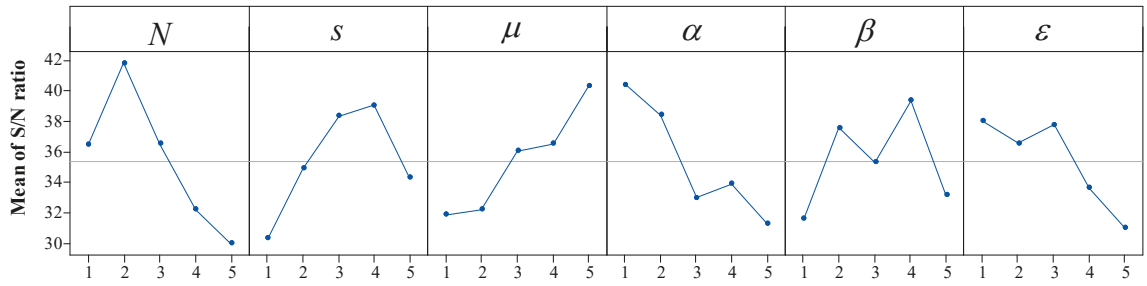
6.4 Parameter settings

QSFLA has seven main parameters: N, s, μ, α, γ, ε and stopping condition. Although the longer the algorithm runs, the more likely it is to get better results, it is found through experiments that QSFLA and its comparison algorithm can converge or hardly improve significantly after running for $0.1 \times S \times n$ seconds, so we choose $0.1 \times S \times n$ seconds as the stopping condition for all algorithms to make the comparison fair which is similar to other studies.

Taguchi method is used to decide the settings of other parameters[5]. Several instances with a different number of jobs, factories, or stages are selected



a) Main effects plot for means



b) Main effects plot for S/N ratios

Figure 7. Main effects plot for means and S/N ratios

for parameter experiments. The same settings can be obtained by using these instances, and we show the result of instance $80 \times 4 \times 4$.

Table 4 exhibits the level of parameters. Table 5 provides the orthogonal array $L_{25}(5^6)$. Each combination runs ten times independently, for instance $80 \times 4 \times 4$, and then the obtained non-dominated solutions are recorded. After all the combinations are tested, the final non-dominated solution set is obtained. The results of *GD* and S/N ratio are shown in Figure 6.3 and 6.3, in which S/N ratio is defined as $-10 \times \log_{10}(DI_R{}^2)$.

As shown in Figure 6.3 and 6.3, QSFLA with $N = 60$, $s = 5$, $\mu = 60$, $\alpha = 0.1$, $\gamma = 0.9$ and $\varepsilon = 0.2$ performs better than QSFLA with other parameter combinations, so the above settings are adopted.

All parameters of QSFLA^{E-} and QSFLA$_v$, $v \in \{1,2,...,16\}$, adopt the same parameter combination with QSFLA in order to explain the role of reinforcement learning and energy saving operator more intuitively. All parameters of NSGA-II except the stopping condition are directly obtained from [7].

Table 5. The orthogonal $L_{25}(5^6)$

Experiment number	Factor level						GD
	N	s	μ	α	β	ε	
1	1	1	1	1	1	1	0.026
2	1	2	2	2	2	2	0.011
3	1	3	3	3	3	3	0.010
4	1	4	4	4	4	4	0.008
5	1	5	5	5	5	5	0.033
6	2	1	2	3	4	5	0.029
7	2	2	3	4	5	1	0.009
8	2	3	4	5	1	2	0.011
9	2	4	5	1	2	3	0.001
10	2	5	1	2	3	4	0.012
11	3	1	3	5	2	4	0.038
12	3	2	4	1	3	5	0.013
13	3	3	5	2	4	1	0.002
14	3	4	1	3	5	2	0.022
15	3	5	2	4	1	3	0.034
16	4	1	4	2	5	3	0.027
17	4	2	5	3	1	4	0.037
18	4	3	1	4	2	5	0.040
19	4	4	2	5	3	1	0.028
20	4	5	3	1	4	2	0.008
21	5	1	5	4	3	2	0.034
22	5	2	1	5	4	3	0.039
23	5	3	2	1	5	4	0.029
24	5	4	3	2	1	5	0.035
25	5	5	4	3	2	1	0.024

Table 6. Computational results on average *GD* with different number of factories

	F=2	F=3	F=4	F=5	F=6
QSFLA	0.000	0.000	0.000	0.000	0.000
QSFLA1	0.011	0.013	0.014	0.013	0.013
QSFLA2	0.009	0.010	0.011	0.011	0.011
QSFLA3	0.010	0.010	0.008	0.009	0.010
QSFLA4	0.008	0.009	0.011	0.010	0.010
QSFLA5	0.031	0.023	0.022	0.021	0.022
QSFLA6	0.031	0.021	0.017	0.016	0.017
QSFLA7	0.025	0.023	0.020	0.020	0.021
QSFLA8	0.022	0.019	0.018	0.018	0.019
QSFLA9	0.027	0.022	0.021	0.021	0.023
QSFLA10	0.027	0.025	0.021	0.022	0.025
QSFLA11	0.023	0.025	0.022	0.023	0.025
QSFLA12	0.023	0.024	0.023	0.023	0.024
QSFLA13	0.006	0.005	0.007	0.006	0.007
QSFLA14	0.006	0.006	0.006	0.007	0.008
QSFLA15	0.010	0.011	0.011	0.011	0.011
QSFLA16	0.009	0.010	0.013	0.012	0.013

Table 7. Computational results on average *GD* with different number of jobs

	N=20	N=40	N=60	N=80	N=100	N=120	N=140
QSFLA	0.000	0.000	0.000	0.000	0.000	0.000	0.000
QSFLA1	0.008	0.010	0.011	0.013	0.014	0.016	0.018
QSFLA2	0.007	0.007	0.008	0.011	0.014	0.013	0.015
QSFLA3	0.007	0.009	0.008	0.011	0.010	0.010	0.010
QSFLA4	0.007	0.009	0.009	0.010	0.011	0.011	0.011
QSFLA5	0.020	0.018	0.020	0.027	0.026	0.024	0.031
QSFLA6	0.016	0.014	0.014	0.025	0.025	0.021	0.026
QSFLA7	0.015	0.019	0.022	0.026	0.024	0.024	0.024
QSFLA8	0.018	0.018	0.018	0.022	0.021	0.018	0.020
QSFLA9	0.021	0.018	0.017	0.028	0.024	0.024	0.028
QSFLA10	0.021	0.017	0.019	0.026	0.027	0.027	0.032
QSFLA11	0.016	0.020	0.023	0.028	0.026	0.026	0.027
QSFLA12	0.022	0.019	0.024	0.027	0.025	0.023	0.024
QSFLA13	0.005	0.005	0.006	0.007	0.007	0.007	0.006
QSFLA14	0.005	0.006	0.006	0.007	0.008	0.007	0.006
QSFLA15	0.012	0.011	0.010	0.011	0.011	0.011	0.010
QSFLA16	0.012	0.013	0.012	0.011	0.011	0.010	0.011

Table 8. Computational results on average *GD* with different number of stages

	S=2	S=4	S=6	S=8
QSFLA	0.000	0.000	0.000	0.000
QSFLA1	0.011	0.013	0.014	0.013
QSFLA2	0.009	0.010	0.012	0.012
QSFLA3	0.008	0.010	0.010	0.009
QSFLA4	0.009	0.010	0.010	0.010
QSFLA5	0.025	0.024	0.025	0.022
QSFLA6	0.022	0.020	0.021	0.019
QSFLA7	0.019	0.023	0.023	0.022
QSFLA8	0.018	0.021	0.019	0.019
QSFLA9	0.024	0.024	0.022	0.021
QSFLA10	0.025	0.026	0.024	0.022
QSFLA11	0.025	0.025	0.024	0.021
QSFLA12	0.030	0.023	0.021	0.019
QSFLA13	0.004	0.006	0.007	0.007
QSFLA14	0.004	0.006	0.007	0.009
QSFLA15	0.011	0.011	0.011	0.010
QSFLA16	0.012	0.012	0.011	0.011

Table 9. Computational results on average *Spacing* with different number of factories

	F=2	F=3	F=4	F=5	F=6
QSFLA	0.010	0.013	0.012	0.014	0.015
QSFLA1	0.014	0.015	0.018	0.018	0.015
QSFLA2	0.014	0.015	0.013	0.013	0.014
QSFLA3	0.010	0.008	0.009	0.010	0.013
QSFLA4	0.011	0.011	0.010	0.012	0.013
QSFLA5	0.011	0.009	0.008	0.011	0.008
QSFLA6	0.010	0.010	0.007	0.008	0.010
QSFLA7	0.008	0.008	0.009	0.012	0.013
QSFLA8	0.012	0.008	0.009	0.011	0.013
QSFLA9	0.010	0.009	0.010	0.009	0.011
QSFLA10	0.006	0.010	0.011	0.013	0.012
QSFLA11	0.012	0.015	0.013	0.012	0.016
QSFLA12	0.007	0.013	0.010	0.010	0.011
QSFLA13	0.012	0.011	0.013	0.012	0.012
QSFLA14	0.011	0.014	0.013	0.016	0.013
QSFLA15	0.009	0.015	0.013	0.012	0.013
QSFLA16	0.013	0.010	0.020	0.014	0.013

Table 10. Computational results on average *Spacing* with different number of jobs

	N=20	N=40	N=60	N=80	N=100	N=120	N=140
QSFLA	0.017	0.014	0.012	0.013	0.013	0.010	0.010
QSFLA1	0.021	0.022	0.013	0.016	0.014	0.012	0.015
QSFLA2	0.018	0.013	0.012	0.014	0.013	0.012	0.014
QSFLA3	0.017	0.013	0.009	0.011	0.007	0.006	0.007
QSFLA4	0.015	0.015	0.011	0.011	0.010	0.008	0.010
QSFLA5	0.017	0.011	0.008	0.009	0.007	0.009	0.006
QSFLA6	0.017	0.014	0.009	0.010	0.005	0.005	0.005
QSFLA7	0.016	0.010	0.010	0.010	0.008	0.008	0.006
QSFLA8	0.019	0.012	0.012	0.010	0.008	0.006	0.007
QSFLA9	0.016	0.009	0.010	0.012	0.008	0.008	0.005
QSFLA10	0.016	0.011	0.014	0.010	0.006	0.010	0.006
QSFLA11	0.015	0.013	0.013	0.015	0.012	0.010	0.016
QSFLA12	0.018	0.011	0.009	0.013	0.010	0.007	0.005
QSFLA13	0.013	0.012	0.010	0.014	0.011	0.013	0.012
QSFLA14	0.014	0.015	0.017	0.014	0.018	0.009	0.008
QSFLA15	0.023	0.013	0.012	0.009	0.014	0.008	0.006
QSFLA16	0.016	0.021	0.017	0.015	0.010	0.009	0.011

Table 11. Computational results on average *Spacing* with different number of stages

	S=2	S=4	S=6	S=8
QSFLA	0.010	0.014	0.016	0.011
QSFLA1	0.019	0.019	0.014	0.012
QSFLA2	0.017	0.015	0.011	0.011
QSFLA3	0.011	0.012	0.009	0.008
QSFLA4	0.013	0.014	0.009	0.010
QSFLA5	0.011	0.008	0.010	0.008
QSFLA6	0.012	0.010	0.008	0.008
QSFLA7	0.011	0.008	0.010	0.010
QSFLA8	0.012	0.010	0.009	0.011
QSFLA9	0.014	0.009	0.008	0.009
QSFLA10	0.014	0.012	0.009	0.007
QSFLA11	0.014	0.012	0.014	0.014
QSFLA12	0.014	0.010	0.008	0.009
QSFLA13	0.012	0.014	0.012	0.010
QSFLA14	0.018	0.011	0.011	0.014
QSFLA15	0.018	0.010	0.010	0.011
QSFLA16	0.019	0.013	0.012	0.012

Table 12. Computational results on average Δ with different number of factories

	F=2	F=3	F=4	F=5	F=6
QSFLA	0.777	0.769	0.716	0.728	0.743
QSFLA1	0.964	0.969	0.964	0.955	0.939
QSFLA2	0.969	0.983	0.962	0.961	0.952
QSFLA3	0.888	0.882	0.849	0.865	0.859
QSFLA4	0.903	0.903	0.879	0.884	0.888
QSFLA5	0.975	0.972	0.962	0.960	0.946
QSFLA6	0.988	0.973	0.970	0.969	0.966
QSFLA7	0.912	0.906	0.905	0.902	0.899
QSFLA8	0.934	0.918	0.924	0.917	0.915
QSFLA9	0.980	0.977	0.966	0.964	0.956
QSFLA10	0.978	0.979	0.979	0.965	0.966
QSFLA11	0.952	0.945	0.922	0.922	0.914
QSFLA12	0.946	0.957	0.948	0.939	0.932
QSFLA13	0.962	0.957	0.940	0.925	0.880
QSFLA14	0.965	0.979	0.955	0.950	0.939
QSFLA15	0.892	0.901	0.871	0.849	0.859
QSFLA16	0.912	0.896	0.892	0.883	0.869

Table 13. Computational results on average Δ with different number of jobs

	N=20	N=40	N=60	N=80	N=100	N=120	N=140
QSFLA	0.678	0.731	0.739	0.768	0.780	0.747	0.784
QSFLA1	0.863	0.948	0.956	0.985	0.984	0.976	0.995
QSFLA2	0.899	0.946	0.961	0.982	0.985	0.988	0.996
QSFLA3	0.812	0.848	0.867	0.884	0.883	0.887	0.900
QSFLA4	0.798	0.866	0.884	0.903	0.919	0.926	0.943
QSFLA5	0.888	0.952	0.966	0.976	0.985	0.989	0.986
QSFLA6	0.916	0.975	0.978	0.981	0.979	0.995	0.990
QSFLA7	0.849	0.884	0.898	0.915	0.920	0.931	0.936
QSFLA8	0.874	0.907	0.917	0.933	0.940	0.935	0.945
QSFLA9	0.905	0.952	0.973	0.978	0.990	0.992	0.989
QSFLA10	0.909	0.966	0.981	0.987	0.988	0.992	0.990
QSFLA11	0.843	0.894	0.926	0.938	0.965	0.958	0.991
QSFLA12	0.896	0.927	0.941	0.964	0.963	0.952	0.966
QSFLA13	0.836	0.897	0.935	0.962	0.978	0.980	0.939
QSFLA14	0.857	0.937	0.960	0.986	0.994	0.982	0.987
QSFLA15	0.839	0.851	0.870	0.867	0.910	0.892	0.892
QSFLA16	0.807	0.873	0.891	0.906	0.909	0.914	0.936

Table 14. Computational results on average Δ with different number of stages

	S=2	S=4	S=6	S=8
QSFLA	0.747	0.762	0.767	0.711
QSFLA1	0.934	0.970	0.961	0.967
QSFLA2	0.946	0.974	0.965	0.975
QSFLA3	0.845	0.880	0.878	0.872
QSFLA4	0.846	0.899	0.909	0.911
QSFLA5	0.954	0.966	0.964	0.968
QSFLA6	0.975	0.981	0.967	0.970
QSFLA7	0.894	0.903	0.907	0.914
QSFLA8	0.923	0.924	0.917	0.923
QSFLA9	0.973	0.965	0.965	0.971
QSFLA10	0.971	0.980	0.970	0.971
QSFLA11	0.926	0.922	0.930	0.945
QSFLA12	0.956	0.947	0.936	0.938
QSFLA13	0.907	0.943	0.924	0.956
QSFLA14	0.952	0.949	0.966	0.963
QSFLA15	0.883	0.861	0.875	0.879
QSFLA16	0.879	0.879	0.899	0.906

Table 15. Computational results of QSFLA and comparative algorithms on *GD*

Instance	QSFLA	QSFLAE	NSGA-II	Instance	QSFLA	QSFLAE	NSGA-II
1	0.000	0.006	0.008	71	0.000	0.008	0.009
2	0.000	0.007	0.006	72	0.000	0.010	0.020
3	0.000	0.006	0.008	73	0.000	0.010	0.014
4	0.000	0.011	0.010	74	0.000	0.009	0.009
5	0.000	0.006	0.004	75	0.000	0.011	0.020
6	0.000	0.007	0.005	76	0.000	0.007	0.010
7	0.000	0.007	0.008	77	0.000	0.009	0.008
8	0.000	0.007	0.002	78	0.000	0.010	0.019
9	0.000	0.007	0.008	79	0.000	0.011	0.012
10	0.000	0.008	0.006	80	0.000	0.009	0.019
11	0.000	0.006	0.007	81	0.000	0.001	0.004
12	0.000	0.006	0.014	82	0.000	0.008	0.009
13	0.000	0.007	0.005	83	0.000	0.011	0.008
14	0.000	0.009	0.008	84	0.000	0.007	0.020
15	0.000	0.004	0.007	85	0.000	0.007	0.006
16	0.000	0.006	0.011	86	0.000	0.006	0.010
17	0.000	0.011	0.005	87	0.000	0.017	0.014
18	0.000	0.005	0.009	88	0.000	0.006	0.004
19	0.000	0.006	0.006	89	0.000	0.005	0.008
20	0.000	0.006	0.010	90	0.000	0.010	0.016
21	0.000	0.004	0.003	91	0.000	0.013	0.029
22	0.000	0.007	0.012	92	0.000	0.010	0.019
23	0.000	0.009	0.009	93	0.000	0.010	0.013
24	0.000	0.010	0.007	94	0.000	0.007	0.012
25	0.000	0.005	0.007	95	0.000	0.007	0.008
26	0.000	0.011	0.014	96	0.000	0.016	0.010
27	0.000	0.012	0.014	97	0.000	0.019	0.026
28	0.000	0.009	0.007	98	0.000	0.010	0.020
29	0.000	0.008	0.010	99	0.000	0.009	0.008
30	0.000	0.007	0.013	100	0.000	0.007	0.009
31	0.000	0.006	0.010	101	0.000	0.002	0.004
32	0.000	0.007	0.010	102	0.000	0.005	0.009
33	0.000	0.010	0.010	103	0.000	0.011	0.019
34	0.000	0.007	0.005	104	0.000	0.008	0.007
35	0.000	0.008	0.010	105	0.000	0.009	0.006
36	0.000	0.006	0.008	106	0.000	0.006	0.010
37	0.000	0.012	0.007	107	0.000	0.005	0.008
38	0.000	0.007	0.007	108	0.000	0.012	0.023
39	0.000	0.005	0.008	109	0.000	0.005	0.009
40	0.000	0.007	0.014	110	0.000	0.006	0.007
41	0.000	0.006	0.005	111	0.000	0.009	0.010
42	0.000	0.007	0.010	112	0.000	0.015	0.033
43	0.000	0.007	0.010	113	0.000	0.009	0.010
44	0.000	0.007	0.010	114	0.000	0.011	0.017
45	0.000	0.008	0.008	115	0.000	0.010	0.016
46	0.000	0.011	0.012	116	0.000	0.010	0.016
47	0.000	0.008	0.016	117	0.000	0.015	0.012
48	0.000	0.007	0.014	118	0.000	0.007	0.010
49	0.000	0.008	0.013	119	0.000	0.008	0.015
50	0.000	0.007	0.009	120	0.000	0.010	0.024
51	0.000	0.008	0.012	121	0.000	0.007	0.002
52	0.000	0.008	0.014	122	0.000	0.003	0.007
53	0.000	0.006	0.007	123	0.000	0.013	0.012
54	0.000	0.008	0.011	124	0.000	0.012	0.016
55	0.000	0.007	0.012	125	0.000	0.006	0.011
56	0.000	0.008	0.015	126	0.000	0.015	0.029
57	0.000	0.013	0.013	127	0.000	0.012	0.007
58	0.000	0.010	0.011	128	0.000	0.008	0.018
59	0.000	0.009	0.018	129	0.000	0.007	0.010
60	0.000	0.006	0.008	130	0.000	0.017	0.031
61	0.000	0.006	0.001	131	0.000	0.007	0.018
62	0.000	0.010	0.008	132	0.000	0.005	0.006
63	0.000	0.018	0.033	133	0.000	0.017	0.020
64	0.000	0.007	0.005	134	0.000	0.007	0.013
65	0.000	0.006	0.007	135	0.000	0.010	0.009
66	0.000	0.006	0.007	136	0.000	0.006	0.015
67	0.000	0.012	0.019	137	0.000	0.012	0.016
68	0.000	0.007	0.016	138	0.000	0.014	0.021
69	0.000	0.007	0.009	139	0.000	0.008	0.018
70	0.000	0.006	0.012	140	0.000	0.007	0.011

Table 16. Computational results of QSFLA and comparative algorithms on *Spacing*

Instance	QSFLA	QSFLAE	NSGA-II	Instance	QSFLA	QSFLAE	NSGA-II
1	0.009	0.009	0.022	71	0.013	0.007	0.008
2	0.011	0.017	0.008	72	0.015	0.015	0.009
3	0.019	0.012	0.019	73	0.012	0.026	0.031
4	0.019	0.011	0.014	74	0.020	0.009	0.010
5	0.013	0.009	0.007	75	0.023	0.009	0.012
6	0.037	0.015	0.024	76	0.009	0.009	0.005
7	0.030	0.012	0.010	77	0.011	0.012	0.012
8	0.010	0.009	0.007	78	0.015	0.027	0.004
9	0.010	0.011	0.023	79	0.014	0.008	0.008
10	0.012	0.012	0.015	80	0.009	0.006	0.012
11	0.016	0.009	0.015	81	0.004	0.004	0.004
12	0.012	0.008	0.024	82	0.017	0.009	0.007
13	0.010	0.014	0.034	83	0.010	0.010	0.002
14	0.010	0.027	0.065	84	0.006	0.008	0.010
15	0.019	0.010	0.011	85	0.006	0.007	0.004
16	0.022	0.013	0.012	86	0.006	0.007	0.008
17	0.021	0.016	0.021	87	0.021	0.008	0.015
18	0.025	0.018	0.018	88	0.025	0.012	0.007
19	0.029	0.008	0.016	89	0.005	0.007	0.012
20	0.011	0.009	0.010	90	0.013	0.059	0.020
21	0.007	0.014	0.008	91	0.024	0.008	0.005
22	0.010	0.012	0.011	92	0.010	0.008	0.009
23	0.009	0.007	0.014	93	0.014	0.012	0.009
24	0.012	0.008	0.008	94	0.010	0.020	0.005
25	0.008	0.011	0.017	95	0.016	0.011	0.004
26	0.010	0.036	0.008	96	0.012	0.012	0.006
27	0.028	0.008	0.034	97	0.024	0.011	0.013
28	0.011	0.036	0.011	98	0.011	0.024	0.026
29	0.008	0.009	0.027	99	0.008	0.009	0.011
30	0.013	0.010	0.008	100	0.017	0.009	0.005
31	0.014	0.006	0.021	101	0.005	0.009	0.004
32	0.010	0.024	0.006	102	0.004	0.010	0.008
33	0.019	0.028	0.009	103	0.009	0.006	0.004
34	0.021	0.018	0.028	104	0.005	0.009	0.006
35	0.030	0.009	0.012	105	0.006	0.009	0.014
36	0.009	0.009	0.024	106	0.007	0.008	0.003
37	0.013	0.014	0.019	107	0.015	0.005	0.008
38	0.012	0.008	0.018	108	0.006	0.011	0.004
39	0.015	0.010	0.021	109	0.007	0.008	0.010
40	0.012	0.012	0.021	110	0.010	0.009	0.020
41	0.006	0.006	0.009	111	0.016	0.017	0.011
42	0.016	0.009	0.009	112	0.013	0.022	0.033
43	0.014	0.011	0.004	113	0.009	0.007	0.010
44	0.015	0.009	0.011	114	0.014	0.007	0.007
45	0.013	0.010	0.005	115	0.008	0.010	0.017
46	0.018	0.009	0.020	116	0.011	0.008	0.003
47	0.007	0.010	0.015	117	0.013	0.011	0.005
48	0.007	0.010	0.022	118	0.008	0.007	0.004
49	0.010	0.017	0.005	119	0.013	0.008	0.056
50	0.010	0.010	0.007	120	0.012	0.008	0.018
51	0.019	0.014	0.008	121	0.005	0.005	0.008
52	0.007	0.006	0.010	122	0.005	0.006	0.005
53	0.011	0.009	0.011	123	0.010	0.007	0.007
54	0.007	0.021	0.057	124	0.008	0.009	0.003
55	0.009	0.009	0.014	125	0.012	0.008	0.011
56	0.012	0.018	0.049	126	0.010	0.010	0.009
57	0.009	0.014	0.049	127	0.007	0.008	0.018
58	0.031	0.010	0.004	128	0.010	0.005	0.003
59	0.016	0.028	0.015	129	0.006	0.007	0.021
60	0.009	0.015	0.003	130	0.021	0.008	0.004
61	0.013	0.013	0.004	131	0.005	0.006	0.019
62	0.011	0.011	0.012	132	0.006	0.009	0.004
63	0.020	0.011	0.029	133	0.008	0.010	0.011
64	0.007	0.012	0.006	134	0.019	0.006	0.006
65	0.011	0.010	0.017	135	0.009	0.021	0.009
66	0.010	0.008	0.010	136	0.014	0.008	0.008
67	0.014	0.014	0.012	137	0.011	0.006	0.031
68	0.007	0.019	0.005	138	0.009	0.010	0.022
69	0.008	0.007	0.010	139	0.025	0.007	0.025
70	0.011	0.006	0.028	140	0.006	0.010	0.007

Table 17. Computational results of QSFLA and its 2 comparative algorithms on Δ

Instance	QSFLA	QSFLAE	NSGA-II	Instance	QSFLA	QSFLAE	NSGA-II
1	0.594	0.533	0.825	71	0.716	0.740	0.940
2	0.650	0.761	0.940	72	0.691	0.797	0.965
3	0.735	0.730	0.935	73	0.725	0.795	0.942
4	0.678	0.609	0.931	74	0.819	0.756	0.971
5	0.695	0.668	0.876	75	0.769	0.727	0.916
6	0.835	0.767	0.912	76	0.708	0.800	0.961
7	0.825	0.783	0.932	77	0.771	0.620	0.899
8	0.614	0.717	0.923	78	0.793	0.759	0.876
9	0.654	0.678	0.978	79	0.589	0.794	0.979
10	0.581	0.638	0.868	80	0.693	0.712	0.967
11	0.682	0.752	0.899	81	0.666	0.793	0.919
12	0.602	0.685	0.906	82	0.921	0.770	0.952
13	0.545	0.648	0.882	83	0.787	0.817	0.950
14	0.605	0.750	1.042	84	0.693	0.818	0.979
15	0.687	0.787	0.940	85	0.775	0.737	0.932
16	0.777	0.762	0.831	86	0.766	0.796	0.954
17	0.665	0.689	0.839	87	0.835	0.769	0.992
18	0.803	0.782	0.877	88	0.765	0.824	0.984
19	0.768	0.733	0.824	89	0.776	0.710	0.945
20	0.571	0.664	0.918	90	0.822	0.958	0.995
21	0.714	0.764	0.863	91	0.929	0.764	0.974
22	0.696	0.719	0.938	92	0.779	0.806	0.981
23	0.626	0.672	0.946	93	0.792	0.787	0.924
24	0.796	0.725	0.968	94	0.723	0.721	0.947
25	0.799	0.704	0.920	95	0.741	0.746	0.967
26	0.795	0.917	0.906	96	0.784	0.730	0.969
27	0.873	0.647	0.978	97	0.857	0.781	0.918
28	0.726	0.864	0.908	98	0.803	0.858	0.961
29	0.672	0.704	1.020	99	0.647	0.741	0.953
30	0.739	0.712	0.895	100	0.744	0.823	0.964
31	0.731	0.769	0.980	101	0.700	0.814	0.940
32	0.645	0.902	0.939	102	0.712	0.819	0.951
33	0.825	0.786	0.863	103	0.954	0.895	0.978
34	0.765	0.841	0.988	104	0.598	0.845	0.973
35	0.788	0.787	0.910	105	0.762	0.789	0.966
36	0.626	0.786	1.000	106	0.733	0.736	0.959
37	0.759	0.773	0.812	107	0.762	0.791	0.966
38	0.625	0.660	0.944	108	0.680	0.828	0.975
39	0.797	0.812	0.870	109	0.734	0.840	0.952
40	0.623	0.810	0.969	110	0.749	0.757	0.997
41	0.682	0.741	0.913	111	0.813	0.874	0.990
42	0.736	0.786	0.910	112	0.718	0.822	0.998
43	0.823	0.725	0.906	113	0.722	0.755	0.929
44	0.915	0.805	0.976	114	0.858	0.746	0.966
45	0.818	0.759	0.916	115	0.657	0.796	0.953
46	0.847	0.788	0.971	116	0.784	0.761	0.960
47	0.650	0.786	0.948	117	0.781	0.748	0.948
48	0.642	0.730	0.959	118	0.796	0.731	0.930
49	0.794	0.759	0.850	119	0.718	0.776	1.043
50	0.637	0.785	0.929	120	0.703	0.765	0.973
51	0.777	0.738	0.922	121	0.954	0.800	0.993
52	0.597	0.740	0.985	122	0.711	0.837	0.947
53	0.739	0.766	0.904	123	0.911	0.827	0.967
54	0.643	0.857	1.067	124	0.936	0.850	0.988
55	0.645	0.685	0.939	125	0.807	0.786	0.939
56	0.725	0.747	0.999	126	0.834	0.874	0.980
57	0.602	0.779	1.018	127	0.769	0.786	0.983
58	0.933	0.804	0.916	128	0.818	0.768	0.959
59	0.909	0.764	0.963	129	0.649	0.775	0.999
60	0.665	0.851	0.957	130	0.943	0.867	0.974
61	1.004	0.852	0.970	131	0.543	0.758	1.000
62	0.865	0.851	0.962	132	0.680	0.830	0.977
63	1.004	0.727	0.975	133	0.708	0.756	0.962
64	0.706	0.914	0.975	134	0.691	0.689	0.973
65	0.899	0.788	0.960	135	0.713	0.864	0.995
66	0.741	0.756	0.964	136	0.810	0.787	0.991
67	0.807	0.812	0.938	137	0.770	0.717	1.043
68	0.671	0.830	0.921	138	0.842	0.848	0.938
69	0.743	0.682	0.951	139	0.875	0.750	0.994
70	0.644	0.741	1.017	140	0.709	0.802	0.969

6.5 Impact of Q-learning

QSFLA and its 16 variants randomly run ten times on each instance to avoid the unfairness caused by the randomness of algorithms. Table 6-14 shows the computational results. Figure 8 provides the boxplot.

According to the number of factories, these instances are divided into five groups, each group contains 28 instances, and the average *GD* in each group is shown in Table 6. Table 7 provides the average *GD* in 7 groups, and these groups are divided according to the number of jobs, and each group contains 20 instances. Table 8 provides the average *GD* in 4 groups, and these groups are divided according to the number of stages, and each group contains 35 instances. As shown in Table 6, in any group, regardless of F equals 2, 3, 4, 5, or 6, QSFLA can obtain a smaller average *GD* than its variants, indicating that the performance of convergence of the algorithm deteriorates when the Q-learning strategy is eliminated. A single search strategy combination is used. As exhibited in Table 7-8, the average *GD* of QSFLA is less than that of 16 variants in the same group, and the same conclusion can be obtained.

Tables 9-11 display the average *Spacing* in each group divided according to the number of factories, jobs, and stages, and Tables 10-12 exhibit the average Δ. As shown in Tables 12-14, QSFLA can obtain the smaller average *Spacing* than its variants, so the performance of uniformity of QSFLA deteriorates when the Q-learning strategy is eliminated. QSFLA can obtain the average Δ close to those of its variants. The statistical results in Figure 8 also indicates the superior performance of QSFLA.

The above analysis shows that reinforcement learning plays a positive role in QSFLA because a single search strategy has limitations. A reasonable choice of multiple search combinations in the different stages can avoid such limitations and improve the search performance of the algorithm.

6.6 Impact of energy saving operator

QSFLA^{E-} randomly runs ten times on each instance like QSFLA. Table 15-17 provide the computational results on *GD*, *Spacing* and Δ. As shown in Table 15, *GD* of QSFLA is less than that of QSFLA^{E-} on 140 instances, so QSFLA converges better than QSFLA^{E-} in all instances. As stated in Table 16, *Spacing* of QSFLA is smaller than that of QSFLA^{E-} on 140 instances, hence the distribution of non-dominated solutions of QSFLA is more uniform than that of QSFLA^{E-}. As exhibited in Table 17, Δ of QSFLA is closed to that of QSFLA^{E-} on 140 instances, therefor the distribution of non-dominated solutions of QSFLA is expansive. The statistical results in Figure 8 also illustrates the superior performance of QSFLA.

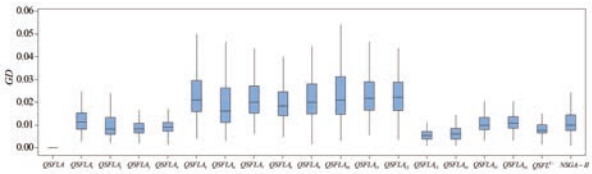
Energy saving strategy can reduce energy consumption as much as possible without increasing the maximum completion time, which reduces the search resources and improves the search efficiency, and this strategy has a positive effect on QSFLA.

6.7 Results and analyses

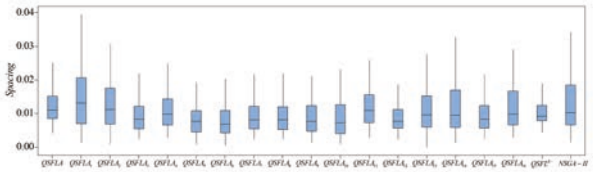
Through the previous analysis, it is evident that reinforcement learning plays a proactive role in QSFLA, and the energy-saving operator also contributes positively. To further validate the performance of QSFLA, a comparative analysis is conducted between QSFLA and NSGA-II, the classical multi-objective optimization algorithm. NSGA-II randomly runs ten times on 140 instance. Table 6-14 shows the computational results of *GD*, *Spacing* and Δ. Figure 8 provides the statistical results in the form of a box plot.

As shown in Table 6-14, the *GD* of QSFLA are smaller than that of NSGA-II on all instances, the *Spacing* of QSFLA are smaller than that of NSGA-II on more than half of all instances, and Δ of QSFLA are closed to that of NSGA-II. The statistical results of Figure 8 also explains the superior performance of QSFLA. So, QSFLA has good performance in convergence, uniformity, and spread.

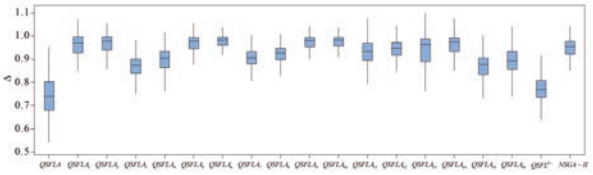
The excellent performance of QSFLA mainly benefits from its Q-learning process. The Q-learning process is used to adjust 16 search strategies and enhance the search ability dynamically. Each strategy is composed of global search and neighborhood search, which can balance exploration and development well. The energy-saving operator offers the possibility of rapidly reducing energy consumption without reducing production efficiency. Therefore, based on the above analysis, QSFLA is a very competitive method to solve DEHFSP.



(a) The plot box of 19 algorithms on metric *GD*



(b) The plot box of 19 algorithms on metric *Spacing*



(c) The plot box of 19 algorithms on metric Δ

Figure 8. The plot box of 19 algorithms on metric *GD*, *Spacing* and Δ

7 Conclusion

The research on DHFSP has received increasing attention in recent years, but the research on DEHFSP has not attracted enough attention. To solve DEHFSP, a mathematical model is established, the lower bound of two optimization objectives is given and proved, and then QSFLA is proposed to minimize maximum completion time and total energy consumption. In the Q-learning process, an action is selected according to the state of the current population after evaluation. Each action includes a global search and a local search. After the memeplexes search, all solutions from all memeplexes first execute an energy-saving operator and then are shuffled to form a new population. The computational results prove that the Q-learning process does have a positive effect, and QSFLA can provide promising results for DEHFSP.

It is a new exploration to combine reinforcement learning and swarm intelligent optimization algorithms to solve shop scheduling problems. In a swarm intelligent optimization algorithm, a series of methods such as adjusting parameter value, adjusting search strategy, adjusting learning object, and adjusting optimization object by using reinforcement learning dynamic are worth exploring. Distributed shop scheduling with high efficiency and energy saving is also a challenge facing the current manufacturing industry. We will continue to study distributed shop scheduling in the future.

Acknowledgments

This work was supported by the Research Initiation Foundation of Anhui Polytechnic University (2022YQQ002), Anhui Polytechnic University Research Project (Xjky2022002), the Open Research Fund of AnHui Key Laboratory of Detection Technology and Energy Saving Devices (JCKJ2022B01), Key Natural Science Research Projects of Colleges and Universities in Anhui Province (2023AH050935, 2022AH050978), Anhui Province University Excellent Top Talent Training Project(gxbjZD2022023), Wuhu science and technology project (2022jc26), the Open Research Fund of Anhui Province Key Laboratory of Detection Technology and Energy Saving Devices, Anhui Polytechnic University (JCKJ2021A06) and Anhui Polytechnic University-Jiujiang District Industrial Collaborative Innovation Special Fund Project (2022cyxtb6).

References

[1] J. C. Cai and D. M. Lei. A cooperated shuffled frog-leaping algorithm for distributed energy-efficient hybrid flow shop scheduling with fuzzy processing time. Complex & Intelligent Systems, 7(5):2235–2253, 2021.

[2] J. C. Cai, R. Zhou, and D. M. Lei. Dynamic shuffled frog-leaping algorithm for distributed hybrid flow shop scheduling with multiprocessor tasks. Engineering Applications of Artificial Intelligence, 90:103540, 2020.

[3] J. C. Cai, R. Zhou, and D. M. Lei. Fuzzy distributed two-stage hybrid flow shop scheduling problem with setup time: collaborative variable search. Journal of Intelligent & Fuzzy Systems, 38(3):3189–3199, 2020.

[4] J.C. Cai, D.M Lei, and M. Li. A shuffled frog-leaping algorithm with memeplex quality for bi-objective distributed scheduling in hybrid flow shop. International Journal of Production Research, 59(18):5404–5421, 2020.

[5] J.C. Cai, D.M Lei, J. Wang, and L. Wang. A novel shuffled frog-leaping algorithm with reinforcement learning for distributed assembly hybrid flow shop scheduling. International Journal of Production Research, 61(4):1233-1251, 2023.

[6] R.H. Chen, B. Yang, S. Li, and S.l. Wang. A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem.

Computers & Industrial Engineering, 149:106778, 2020.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Transactions on Evolutionary Computation, 6(2):182–197, 2002.

[8] Yu Du, Jun-qing Li, Chao Luo, and Lei-lei Meng. A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations. Swarm and Evolutionary Computation, 62, 2021.

[9] M. Eusuff, K. Lansey, and F. Pasha. Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. Engineering Optimization, 38(2):129–154, 2006.

[10] J. H. Hao, J. Q. Li, Y. Du, M. X. Song, P. Duan, and Y. Y. Zhang. Solving distributed hybrid flowshop scheduling problems by a hybrid brain storm optimization algorithm. IEEE Access, 7:66879–66894, 2019.

[11] E. D. Jiang, L. Wang, and Z. P. Peng. Solving energy-efficient distributed job shop scheduling via multi-objective evolutionary algorithm with decomposition. Swarm and Evolutionary Computation, 58, 2020.

[12] E. D. Jiang, L. Wang, and J. J. Wang. Decomposition-based multi-objective optimization for energy-aware distributed hybrid flow shop scheduling with multiprocessor tasks. Tsinghua Science and Technology, 26(5):646–663, 2021.

[13] D. M. Lei, L. Gao, and Y. L. Zheng. A novel teaching-learning-based optimization algorithm for energy-efficient scheduling in hybrid flow shop. Ieee Transactions on Engineering Management, 65(2):330–340, 2018.

[14] D. Lei and T. Wang. Solving distributed two-stage hybrid flowshop scheduling using a shuffled frog-leaping algorithm with memeplex grouping. Engineering Optimization, 52(9):1461–1474, 2019.

[15] J. Q. Li, J. K. Li, L. J. Zhang, H. Y. Sang, Y. Y. Han, and Q. D. Chen. Solving type-2 fuzzy distributed hybrid flowshop scheduling using an improved brain storm optimization algorithm. International Journal of Fuzzy Systems, 23(4):1194–1212, 2021.

[16] Y. L. Li, X. Y. Li, L. Gao, and L. L. Meng. An improved artificial bee colony algorithm for distributed heterogeneous hybrid flowshop scheduling problem with sequence-dependent setup times. Computers & Industrial Engineering, 147, 2020.

[17] Y.L. Li, F. Li, Q.K. Pan, L. Gao, and M. F. Tasgetiren. An artificial bee colony algorithm for the distributed hybrid flowshop scheduling problem. Procedia Manufacturing, 39:1158–1166, 2019.

[18] Y.L. Li, X.Y. Li, L. Gao, B. Zhang, Q.K. Pan, M. F. Tasgetiren, and Leilei Meng. A discrete artificial bee colony algorithm for distributed hybrid flowshop scheduling problem with sequence-dependent setup times. International Journal of Production Research, 59(13):3880–3899, 2021.

[19] C. Lu, L. Gao, Q. K. Pan, X. Y. Li, and J. Zheng. A multi-objective cellular grey wolf optimizer for hybrid flowshop scheduling problem considering noise pollution. Applied Soft Computing, 75:728–749, 2019.

[20] L. Meng, K. Gao, Y. Ren, B. Zhang, H. Sang, and C. Zhang. Novel milp and cp models for distributed hybrid flowshop scheduling problem with sequence-dependent setup times. Swarm and Evolutionary Computation, 71:101058, 2022.

[21] L. Meng, Y.g Ren, B. Zhang, J. Li, H. Sang, and C. Zhang. Milp modeling and optimization of energy-efficient distributed flexible job shop scheduling problem. Ieee Access, 8:191191–191203, 2020.

[22] Z. Pan, D. Lei, and L. Wang. A knowledge-based two-population optimization algorithm for distributed energy-efficient parallel machines scheduling. IEEE Trans Cybern, PP, 2020.

[23] H. Qin, T. Li, Y. Teng, and K. Wang. Integrated production and distribution scheduling in distributed hybrid flow shops. Memetic Computing, 13(2):185–202, 2021.

[24] H.X. Qin, Y.Y. Han, B. Zhang, L.L. Meng, Y.P. Liu, Q.K. Pan, and D.W. Gong. An improved iterated greedy algorithm for the energy-efficient blocking hybrid flow shop scheduling problem. Swarm and Evolutionary Computation, 69, 2022.

[25] W. S. Shao, Z. S. Shao, and D. C. Pi. Multi-objective evolutionary algorithm based on multiple neighborhoods local search for multi-objective distributed hybrid flow shop scheduling problem. Expert Systems with Applications, 183, 2021.

[26] G. Wang, X. Li, L. Gao, and P. Li. An effective multi-objective whale swarm algorithm for energy-efficient scheduling of distributed welding flow shop. Annals of Operations Research, page in press, 2021.

[27] J. Wang, D. Lei, and J. Cai. An adaptive artificial bee colony with reinforcement learning for distributed three-stage assembly scheduling with maintenance. Applied Soft Computing, page 108371, 2021.

[28] J.J. Wang and L. Wang. A bi-population cooperative memetic algorithm for distributed hybrid flow-shop scheduling. IEEE Transactions on Emerging Topics in Computational Intelligence, 5(6):947–961, 2020.

[29] J.J. Wang and L. Wang. A cooperative memetic algorithm with learning-based agent for energy-aware distributed hybrid flow-shop scheduling. IEEE Transactions on Evolutionary Computation, 2021.

[30] L. Wang and D. D. Li. Fuzzy distributed hybrid flow shop scheduling problem with heterogeneous factory and unrelated parallel machine: a shuffled frog leaping algorithm with collaboration of multiple search strategies. IEEE Access, 8:214209–214223, 2020.

[31] K. C. Ying and S. W. Lin. Minimizing makespan for the distributed hybrid flowshop scheduling problem with multiprocessor tasks. Expert Systems with Applications, 92:132–141, 2018.

[32] B. Zhang, Q. K. Pan, L. Gao, X. Y. Li, L. L. Meng, and K. K. Peng. A multiobjective evolutionary algorithm based on decomposition for hybrid flow-shop green scheduling problem. Computers & Industrial Engineering, 136:325–344, 2019.

[33] J. Zheng, L. Wang, and J. J. Wang. A cooperative coevolution algorithm for multi-objective fuzzy distributed hybrid flow shop. Knowledge-Based Systems, 194:105536, 2020.

Jingcao Cai received the Ph.D. degree in transportation information engineering and control from Wuhan University of Technology, Wuhan, China, in 2021. He is currently a Lecturer in the School of Mechanical Engineering, Anhui Polytechnic University, Wuhu, China. His current research interests include intelligent manufacturing system optimization and production scheduling.
https://orcid.org/0000-0002-1440-9849

Lei Wang received the Ph.D. degree in mechanical and electronic engineering from Nanjing University of Aeronautics and Astronautics, Nanjing, China in 2010. Since November 2010, he has been working at Anhui Polytechnic University in Wuhu, China, where he holds the position of professor. His current research interests include intelligent manufacturing system, job shop scheduling and mobile robot path planning.
https://orcid.org/0000-0002-3499-5355