Taylor & Francis
Taylor & Francis Group

International Journal of Production Research

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tprs20

A reinforcement learning-driven brain storm optimisation algorithm for multi-objective energy-efficient distributed assembly no-wait flow shop scheduling problem

Fuqing Zhao, Xiaotong Hu, Ling Wang, Tianpeng Xu, Ningning Zhu & Jonrinaldi

To cite this article: Fuqing Zhao, Xiaotong Hu, Ling Wang, Tianpeng Xu, Ningning Zhu & Jonrinaldi (2022): A reinforcement learning-driven brain storm optimisation algorithm for multi-objective energy-efficient distributed assembly no-wait flow shop scheduling problem, International Journal of Production Research, DOI: 10.1080/00207543.2022.2070786

To link to this article: https://doi.org/10.1080/00207543.2022.2070786

Published online: 12 May 2022.

INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH
https://doi.org/10.1080/00207543.2022.2070786

Taylor & Francis
Taylor & Francis Group

RESEARCH ARTICLE

A reinforcement learning-driven brain storm optimisation algorithm for multi-objective energy-efficient distributed assembly no-wait flow shop scheduling problem

Fuqing Zhao [a], Xiaotong Hu[a], Ling Wang[b], Tianpeng Xu[a], Ningning Zhu[a] and Jonrinaldi[c]

[a]School of Computer and Communication Technology, Lanzhou University of Technology, Lanzhou, People's Republic of China; [b]Department of Automation, Tsinghua University, Beijing, People's Republic of China; [c]Department of Industrial Engineering, Universitas Andalas, Padang, Indonesia

ABSTRACT

A reinforcement learning-driven brain storm optimisation idea (RLBSO) is proposed in this paper to solve multi-objective energy-efficient distributed assembly no-wait flow shop scheduling problem. The objectives of the problem include minimising the maximum assembly completion time (C_{max}), minimising the total energy consumption (TEC) and achieving resource allocation balanced . Four operations, which are critical factory insert, critical factory swap, critical factory insert to other factories, critical factory swap with other factories, are designed to optimise the objective of maximum assembly completion time. Q-learning mechanism is utilised to guide the selection of operations to avoid blind search in the iteration process. The learning mechanism based on clustering mechanism in brain storm optimisation algorithm is utilised to assign products to factories in the objective space according to the processing time of products to balance the resources allocation. The speed of operations on non-critical path is slowed down to reduce TEC regarded with the characteristics of no-wait flow shop scheduling problem. The experimental results under 810 large-scale instances by RLBSO show that the RLBSO outperforms the comparison algorithm for addressing the problem.

ARTICLE HISTORY
Received 4 December 2021
Accepted 14 April 2022

KEYWORDS
Energy-efficient; no-wait flow shop; brain storm optimisation; Q-learning mechanism; product assignment rule; clustering mechanism

1. Introduction

Energy efficient distributed two-stage assembly NWFSP, which is an important problem in distributed manufacturing systems (Wang et al. 2020b; Meng et al. 2019), has important application value in practical engineering (Wu et al. 2018; Wang and Wang 2019), including engine assembly and personal computer manufacturing.

In the actual distributed flow shop scheduling process (DPFSP) (Pan, Zou, and Duan 2018), the number of jobs and machines to be processed is large, and the number of processes involved in the production stage is huge. Distributed manufacturing makes full use of the resources of multiple production lines and adopts reasonable processing and resource allocation mode(Wang et al. 2018), which is contributed to maximise the benefits of production and manufacturing (Kemmoé-Tchomté, Lamy, and Tchernev 2017).

Reducing energy consumption (Tan et al. 2020; Hosseini, Carli, and Dotoli 2020) is an important issue that conforms to China's national conditions and benefits people's livelihood. Different energy consumption is generated at different machining speeds in flow shop scheduling problem. Reasonable setting of processing speed and reduction of standby time effectively reduces energy consumption.

Reasonable use of resources is the key factor to improve production efficiency. Unreasonable use prolongs the construction period, and makes the workload on each production line not balancing. Allocating the resource balancing (Liu et al. 2020) refers to the optimal allocation and rational use of existing resources to reduce excessive or uneven allocation of resources, and improve the efficiency in the use of resources.

Considering the practical significance of the above problems and the actual application scenarios, it is of great significance to study the problems aiming at assembly completion time, energy efficient and resource allocation balance (Shoaardebili and Fattahi 2015). RLBSO

CONTACT Fuqing Zhao fzhao2000@hotmail.com School of Computer and Communication Technology, Lanzhou University of Technology, Lanzhou 730050, People's Republic of China

Supplemental data for this article can be accessed here. https://doi.org/10.1080/00207543.2022.2070786

algorithm is proposed to solve the energy efficient distributed two-stage assembly NWFSP. The main contributions are summarised below.

- A mixed integer linear programming model of the energy-efficient distributed assembly no-wait flow shop scheduling problem (DANWFSP) is proposed. According to the actual production situation, three objectives of minimising C_{max}, total energy consumption (TEC) and resource allocation balanced (RAB) are proposed. A new objective of resource allocation balance, which is quantified by equipment utilisation per factory, is designed to improve resource utilisation.
- Q-learning mechanism is used to guide the use of the four operations, which avoids blind search in the solution space. Four operations are designed to take full advantage of information within and between factories, including critical factory insert, critical factory swap, critical factory insert to other factories and critical factory swap to other factories.
- The clustering mechanism in the brain storm optimisation algorithm (BSO) is mapped to the product assignment method in the problem. The products are clustered to the factories in the objective space according to their assembly completion time, which makes the product distribution in the factories balanced, and is conducive to the optimisation of resource distribution balance target.

The rest of the paper is organised as follows. Literature review is given in Section 2. The definition and analysis of the problem are presented in Section 3. The solving method for the problem is proposed in Section 4. The strategies of the RLBSO algorithm are analysed in Section 5. The results between the RLBSO algorithm and the comparison algorithm are given in Section 6. Section 7 summarises the conclusions and future work.

2. Literature review

Flow shop scheduling problem (FSP), which is a typical combinatorial optimisation problem, is divided into permutation FSP (Jiang and Wang 2019), no-wait FSP (Dong et al. 2021; Zhao et al. 2021c), no-idle FSP (Shen, Wang, and Wang 2015) and blocking FSP (Zhao et al. 2020).

2.1. Assembly FSP

Flow shop scheduling problem with assembly (AFSP) (Yokoyama and Santos 2005; Lei, Su, and Li 2020) is an important problem in a large amount of manufacturing systems, consisting of two stages: production and assembly. In AFSP (Mozdgir et al. 2013), the process of the jobs that make up a product is determined, and then the product is assembled on the assembly machine.

An improved iterative greedy algorithm (Li et al. 2021) is proposed to solve distributed assembly hybrid no-idle permutation flow shop scheduling problem with total delay objective. A cooperative teaching-learning based optimisation method (CTLBO) is presented to optimise the distributed two-stage assembly flow scheduling problem (Lei, Su, and Li 2020). A large number of experimental results show that CTLBO has very competitive performance in solving the DTAFSP. Three hybrid metaheuristic methods (HVNS, HGA-RVNS and HDDE-RVNS) are proposed to solve a new distributed two-stage assembly flow scheduling problem (Xiong et al. 2014). The objective is to allocate the work to multiple factories and arrange the job for each factory with the minimum total completion time (TCT). In (Deng et al. 2016), a competitive memetic algorithm is proposed to solve distributed two-stage assembly FSP.

An integrated scheduling problem under distributed production and flexible assembly settings is considered in (Zhang, Xing, and Cao 2018), aiming at shortening the construction period to the greatest extent. A new hybrid simulated annealing and insertion algorithm (SMI) (Aydilek, Aydilek, and Allahverdi 2017) is proposed to solve the two-stage assembly FSP, aiming at minimising delay time. Several developed dominant relations are added in the insertion step. A new distributed assembly permutation FSP (Pan et al. 2019) is proposed to minimise the C_{max}.

A distributed assembly no-idle FSP, is proposed in (Shao, Pi, and Shao 2019). Iterative local search (ILS) and variable neighbourhood search (VNS) are proposed to solve the problem which aims at minimising the C_{max} in assembly stage. A collaborative water wave optimisation algorithm (Zhao et al. 2021a) is proposed to solve the DANIFSP with the goal of minimising the maximum assembly completion time. An optimal block knowledge-driven backtracking search algorithm is proposed for distributed AFSP with the goal of minimising assembly completion time (Zhao et al. 2021). The results in the test set verify the superiority of BKBSA.

2.2. Energy-efficient FSP

Energy-efficient production scheduling, which is an effective way to reduce energy consumption and develop green industry, has attracted more and more attention of many manufacturing enterprises (Li et al. 2021; Zheng et al. 2020; Zhao, He, and Wang 2020; Liu et al. 2018; Gao et al. 2020).

A multi-objective optimisation model of blocking FSP with C_{max} and TEC as indexes is established, and a discrete evolutionary multi-objective optimisation algorithm (DEMO) is proposed (Han et al. 2020). A distributed heterogeneous welding FSP (Wang et al. 2021) is proposed to minimise TEC and C_{max}. A decomposition based multi-objective evolutionary algorithm (MOEA/D) is proposed, in which multiple genetic operators and local search strategies are designed for multi-stage optimisation.

In (Wang et al. 2020a), the distributed permutation FSP is divided into three sub-problems. A multi-objective whale swarm algorithm is proposed to solve DPFSP with C_{max} and TEC as the objectives. An energy saving no-wait FSP (Yuksel et al. 2020) is studied to find pareto solution set with minimum conflict between C_{max} and TEC. A variable block insertion heuristic algorithm and an efficient iterative greedy algorithm are proposed to solve NWFSP. A decomposition based three-stage multi-objective method (TMOA/D) (Zhang et al. 2020) is proposed to solve the energy-efficient hybrid FSP with the goal of minimising C_{max} and TEC.

A knowledge-based collaborative algorithm (KCA) (Wang and Wang 2018) is proposed to solve the energy-efficient DFSP, with the C_{max} and minimum TEC as the criteria. A knowledge-based two-population optimisation algorithm for distributed high-efficiency scheduling problems is proposed (Pan, Lei, and Wang 2020). In (Chen, Wang, and Peng 2019), a collaborative optimisation algorithm is proposed to study the distributed no-idle permutation FSP, and the goal is to simultaneously minimise the C_{max} and TEC. Numerous experiments verify the effectiveness of this algorithm.

2.3. The solution method for FSP

BSO algorithm has been applied to various scheduling problems in recent years (Guo et al. 2020; Wu 2019). In (Li et al. 2020), an improved brain storm optimisation algorithm (IBSO) is used to solve the distributed hybrid FSP. The results show that IBSO is an effective method to solve DHFSP. A self-adaptive brain storm optimisation method (Bhatt, Dimri, and Aggarwal 2020) is proposed to solve the problem of scheduling heterogeneous cloud resources under constraints. Experimental results show that the proposed algorithm has good performance.

An improved iterative greedy (IIG) algorithm is proposed to solve DPFSP problems with robot transport and order constraints (Li et al. 2022). In (Miyata and Nagano 2021), a multi-start iterative greed (MSIG) algorithm is proposed to minimise the maximum completion time of the DPFSP with preventive maintenance operation is studied. An iterative greedy algorithm (Mao et al. 2020) with variable search neighbourhood is proposed to solve the DFSP with related establishment time and maintenance operation sequence. A DPFSP with mixed no-idle constraints is studied in (Yzla et al. 2021), and an adaptive iterative greedy algorithm with sample length varying with search process is designed.

Reinforcement learning (Zhang et al. 2021; Heger and Voss 2021), which is a method framework for decision making, rewards and punishes by taking certain actions and behaviours in the process of decision making. An innovative reinforcement learning method is used as a super-heuristic method to dynamically adjust the K-value of ATCS ordering rules in complex manufacturing scenarios to find the optimal sequence in flexible flow shop (Heger and Voss 2021). The method of explainable reinforcement learning in production control is studied, and a method to improve the rationality of control strategy based on RL is proposed (Kuhnle et al. 2021). In order to minimise the C_{max}, a framework based on deep reinforcement learning and iterative greed algorithm for flow shop scheduling is proposed (Wang and Pan 2021). A deep reinforcement learning algorithm based on time-series difference method is proposed to solve the non-permutation FSP by transforming the scheduling problem into a multi-stage decision problem (Xiao et al. 2020). Reinforcement learning (Lee and Kim 2021) is used to solve a robot FSP to minimise the C_{max}.

In the above literature, experts and scholars have studied the distributed assembly FSP, energy saving FSP and the solution method. The researches on energy saving and resource allocation in DANWFSP is rare. Considering the time cost, environmental benefit and economic conditions, it is a value research problem to simultaneously optimise the completion time, energy consumption and resource allocation.

3. The definition of the problem

3.1. Notation

The notations and the meanings in this paper are defined as follows.

n	The number of jobs
m	The number of machines
q	The number of products
f	The index of factories
l	The index of a certain job in the job sequence
π_P^f	The sequence of products in factory f
P_h	The product h
i	The index of machines
j	The index of jobs
$O_{j,i}$	The operation of job$j$ on machine i

v	The processing speed of $O_{j,i}$
s	The set of speeds
V	The level of speed
p	The standard processing time
t	The real processing time
a	The real assembly time
$C_{l,i}^f$	The makespan of factory f
C_h	The assembly completion time of production h
h	The index of the products
k	The index of a certain product in the product sequence
Max	A sufficiently large positive number
PEC_f	Energy consumption of processing and assembly in factory f
SEC_f	Energy consumption of idle machines in factory f
TEC	The total energy consumption
MIT_f	Machine idle time in factory f
RAB	Resource allocation balancing

3.2. Mixed integer linear programming model of energy-efficient DANWFSP

For energy-efficient DANWFSP, P products are assigned to F factories for processing, and each factory has M processes for processing on M machines. There are N jobs to form P products, and the number of jobs of each product is different. An assembly machine, which exists in each factory, is used to assemble the jobs that make up a particular product after they are processed. The processes inside the distributed factories are the same. The no-wait constraint is added compared with the general flow shop scheduling problem. There is no waiting time for each job in each process. Each machine in the factory has a speed at which it processes. A speed selected for each operation is not able to be changed until the operation is complete. The time of speed switching between processes is not considered. Large processing speed leads to small completion time and high energy consumption. The mixed integer linear programming model is defined as follows:

$$\min\{C_{\max}(\pi_P^{f^*}), \mathrm{TEC}, \mathrm{RAB}\} \tag{1}$$

$$\sum_{l=1}^{n}\sum_{f=1}^{F} x_{j,l}^f = 1 \forall j \tag{2}$$

$$\sum_{j=1}^{n}\sum_{f=1}^{F} x_{j,l}^f = 1 \forall l \tag{3}$$

$$\sum_{v=1}^{s} z_{j,i}^v = 1 \forall j, i \tag{4}$$

$$p_{j,i} = t_{j,i} \cdot \sum_{v=1}^{s} \frac{z_{j,i}^v}{V_v} \forall j, i \tag{5}$$

$$C_{1,1}^f = \sum_{j=1}^{n} x_{j,1}^f \cdot p_{j,1} \forall f \tag{6}$$

$$C_{l,i}^f = C_{l,i-1}^f + \sum_{j=1}^{n} x_{j,l}^f \cdot p_{j,i} \forall l, i, f \tag{7}$$

$$C_{l,i}^f \geq C_{l-1,i}^f + \sum_{j=1}^{n} x_{j,l}^f \cdot p_{j,i} \forall l, i, f \tag{8}$$

$$\sum_{i=1}^{P_h} x_{i,l}^f = P_h \tag{9}$$

$$\sum_{k=0,k\neq h}^{r} P_{k,h} = 1 \forall h \tag{10}$$

$$\sum_{h=1,k\neq h}^{r} P_{k,h} \leq 1 \forall k \tag{11}$$

$$P_{k,h} + P_{h,k} \leq 1 \forall k \in \{1, 2, \cdots r-1\}, h > k \tag{12}$$

$$C_h \geq (C_{j,m} \cdot G_{j,h}) + t_h \tag{13}$$

$$C_h \geq C_k + t_h + (P_{k,h} - 1) \cdot \mathrm{Max} \forall k, r \tag{14}$$

$$C_{\max}(\pi_P^f) \geq C_h \tag{15}$$

$$x_{j,l}^f \in \{0, 1\}, \forall j, l, f \tag{16}$$

$$z_{j,i}^v \in \{0, 1\}, \forall j, i, v \tag{17}$$

$$P_{k,h} \in \{0, 1\}, \forall k, h \tag{18}$$

$$G_{j,h} \in \{0, 1\}, \forall j, h \tag{19}$$

Equation (1) represents the objective, including minimising $C_{\max}$, TEC and RAB. Constraint (2) indicates that each job is machined at only one location in a factory. Constraint (3) implies that the job is not allowed to be transferred from factory to factory during processing. (4) means that only one speed is used for an operation. Constraint (5) indicates the real processing time at a given speed. Constraints (6)–(8) satisfy the property of the no-wait flow shop problem. Constraint (9) ensures that jobs belonging to the same product must be assigned to the same factory for processing.

Constraints (10) and (11) indicate that a product has only one immediate precursor and one immediate successor. (12) means that the product is not able to be repeated in a processing sequence. (13) ensures that the assembly of the product is not started until the jobs that make up the product have been processed. (14) indicates that the assembly of the next product is not started until

the current product is assembled. (15) implies the maximum assembly completion time of the product. The decision variables are defined in (16)–(19). The calculation method of energy consumption is shown in (20)–(22). The calculation method of resource allocation balance is shown in (23) and (24).

$$\mathrm{PEC}_f = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{l=1}^{n}\left(x_{j,l}^{f}\cdot t_{j,i}\cdot\sum_{v=1}^{s}z_{j,i}^{v}\cdot PP_{f,i}^{v}\right) + \sum_{h=1}^{q}\sum_{k=1}^{q}\left(x_{h,k}^{f}\cdot a_h\cdot\sum_{v=1}^{s}z_{j,i}^{v}\cdot PP_{f,a}^{v}\right)\forall f \quad (20)$$

$$\mathrm{SEC}_f = \left(C_f - \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{l=1}^{n}(x_{j,l}^{f}\cdot t_{j,i})\right)\cdot SP_f + \left(C_f - \sum_{h=1}^{q}\sum_{k=1}^{q}(x_{h,k}^{f}\cdot a_h)\right)\forall f \quad (21)$$

$$\mathrm{TEC} = \sum_{f=1}^{F}(\mathrm{PEC}_f + \mathrm{SEC}_f)\forall f \quad (22)$$

$$\mathrm{MIT}_f = (m+1)*C_{\max}(\pi_P^{f^*}) - \left(\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{l=1}^{n}(x_{j,l}^{f}\cdot t_{j,i}) + \sum_{h=1}^{q}\sum_{k=1}^{q}(x_{h,k}^{f}\cdot a_h)\right)\forall f \quad (23)$$

$$\mathrm{RAB} = \sqrt{\sum_{f=1}^{F}\left(\mathrm{MIT}_f - \mathrm{mean}\left(\sum_{f=1}^{F}\mathrm{MIT}_f\right)\right)^2/(n-1)} \quad (24)$$

3.3. The objective analysis

An example is used to explain the relationship between the three objectives. Four products are assigned to two factories, which has three processing machines and one assembly machine. The standard processing time of the nine jobs that make up the product is $P_{j,i}$, and the standard assembly time of the four products is AP. The processing sequences in the factories are $\pi^1 = \{2,5,6\}, \pi^2 = \{1,7\}, \pi^3 = \{4,9\}, \pi^4 = \{3,8\}$. $V = \{0.8, 1, 1.2\}$ is the three levels of machining and assembly speed.

$$AP = \begin{pmatrix} 42 \\ 62 \\ 51 \\ 49 \end{pmatrix} \quad P_{j,i} = \begin{pmatrix} 30 & 52 & 42 \\ 24 & 32 & 14 \\ 32 & 29 & 65 \\ 5 & 43 & 32 \\ 17 & 48 & 99 \\ 58 & 54 & 23 \\ 64 & 85 & 52 \\ 42 & 24 & 17 \\ 63 & 54 & 13 \end{pmatrix}$$

$$vA_{\mathrm{rand}} = \begin{pmatrix} 1 \\ 1.2 \\ 1.2 \\ 1 \end{pmatrix} \quad v_{\mathrm{rand}} = \begin{pmatrix} 1 & 1.2 & 1 \\ 0.8 & 0.8 & 0.8 \\ 0.8 & 1.2 & 1 \\ 0.8 & 0.8 & 1 \\ 1 & 1 & 0.8 \\ 1.2 & 1.2 & 1.2 \\ 1.2 & 1.2 & 0.8 \\ 1.2 & 1 & 0.8 \\ 0.8 & 1 & 1 \end{pmatrix}$$

First, the processing speed of the jobs and the assembly speed of the products are set as maximum, random and minimum. When the speed is random, v_{rand} is the job processing speed and vA_{rand} is the product assemble speed. Second, the values of the three targets in the three cases are calculated below. With the increase of the speed, the assembly completion time decreases, and TEC increases, but the RAB changes randomly. The three objectives are in conflict from Figure 1.

$$\begin{bmatrix} \text{Max speed} \\ \text{Random speed} \\ \text{Min speed} \end{bmatrix} \quad \text{value} = \begin{bmatrix} 307, 1716, 11.7851 \\ 417, 1454, 67.5876 \\ 461, 1091, 17.6777 \end{bmatrix}$$

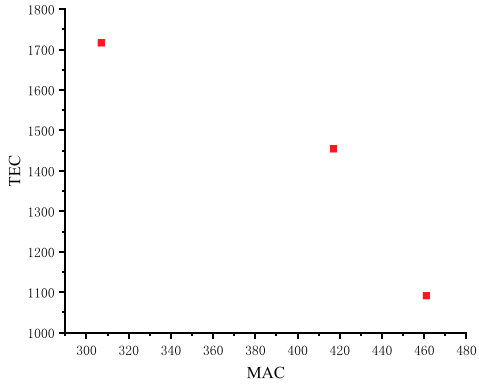
3.4. An instance

Consider an instance shown in Table 1 where nine jobs that make up four products are processed and assembled in two factories.

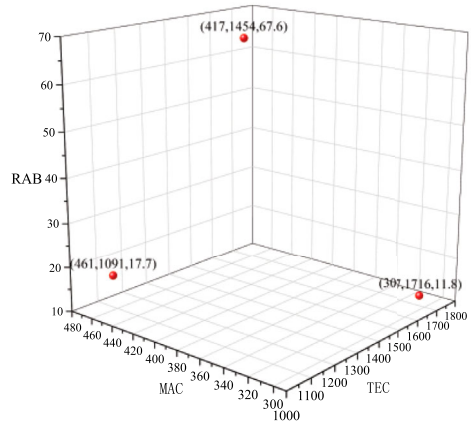
Each factory has three processing machines and one assembly machine. $P_{i,j}$ and $V_{i,j}$ represent the standard processing time and speed levels, respectively, $i \in \{1,2,3\}$. Job sequences of products are $\pi^1 = \{2,5,6\}$, $\pi^2 = \{1,7\}$, $\pi^3 = \{4,9\}$, $\pi^4 = \{3,8\}$. The processing sequences of the product in the factory are $f^1 = \{\pi^1, \pi^4\}$, $f^2 = \{\pi^2, \pi^3\}$. P_h and V_h represent the standard assembly time and assembly speed levels of four products in two factories, respectively. The detailed processing process of the example is shown in the Gantt chart in Figure 2.

$$\begin{bmatrix} Max\ speed \\ Random\ speed \\ Min\ speed \end{bmatrix}$$

$$value = \begin{bmatrix} 307,1716,11.7851 \\ 417,1454,67.5876 \\ 461,1091,17.6777 \end{bmatrix}$$



(a) Results of TEC and C_{max}.



(b) Results of RAB, TEC and C_{max}.

Figure 1. Results of the instance.

Table 1. The processing and composition of the products.

Job		J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9
M_1	$P_{1,j}$	3.6	3.2	3.6	5	7	6	7.2	4	6
	$V_{1,j}$	1	3	1	2	2	1	1	2	2
M_2	$P_{2,j}$	5	3	1.6	3.2	4.8	5	8	1.6	4
	$V_{2,j}$	2	2	3	3	1	2	2	3	2
M_3	$P_{3,j}$	4.8	1	6	2.4	7.2	1.6	5	0.8	3
	$V_{3,j}$	1	2	2	3	3	3	2	3	2
P_h			P_1 (J_2,J_5,J_6)		P_2 (J_1,J_7)		P_3 (J_4,J_9)		P_4 (J_3,J_8)	
M_A	P_h		4.8		6		4		7	
	V_h		1		2		3		2	

The assembly completion time is $C_{\max}(\pi_P^{f*}) = 40$. According to (20)–(24), the energy consumption is 146.9, and the resource allocation balancing is 1.414.

4. The RLBSO method

4.1. The brain storm optimisation algorithm

BSO algorithm (Ma, Cheng, and Shi 2020; Sun et al. 2020), which is a swarm intelligence algorithm inspired by human brainstorming behaviour, has significant advantages in solving high-dimensional and large-scale optimisation problems.

Clustering (Yan et al. 2020), which is the core idea of BSO, increases the convergence speed of the algorithm in the process of solving problems. The introduction of clustering strategy enables the algorithm to find local optimal solutions, so that global optimal solution is obtained from local optimal solutions. There are four ways for BSO to generate new individuals, including two types of intra-cluster search and two types of inter-cluster search. The in-cluster search method is to learn from one cluster to increase the exploitation ability of the algorithm and help to find the potential global optimal solution. The inter-cluster search method, which is to learn from two clusters, helps the algorithm to jump out of local optimum.

4.2. The initialisation method

The clustering mechanism in BSO is mapped into the product assignment. Individuals in BSO are mapped to products to be assigned in the product assignment stage, and the number of jobs in each product is the individual dimension. Unlike BSO, the mapped individuals have different dimensions. Clusters in BSO are mapped to factories, and the number of factories is the number of individual clusters in the BSO algorithm.

K-means method, which is used for clustering in the solution space in the original BSO algorithm, has high computational complexity. This paper operates in a one-dimensional objective space and clusters products according to the assembly completion time of products. The calculation time of this method only depends on the size of the products to be assigned, not the number of jobs, so the evaluation times are saved.

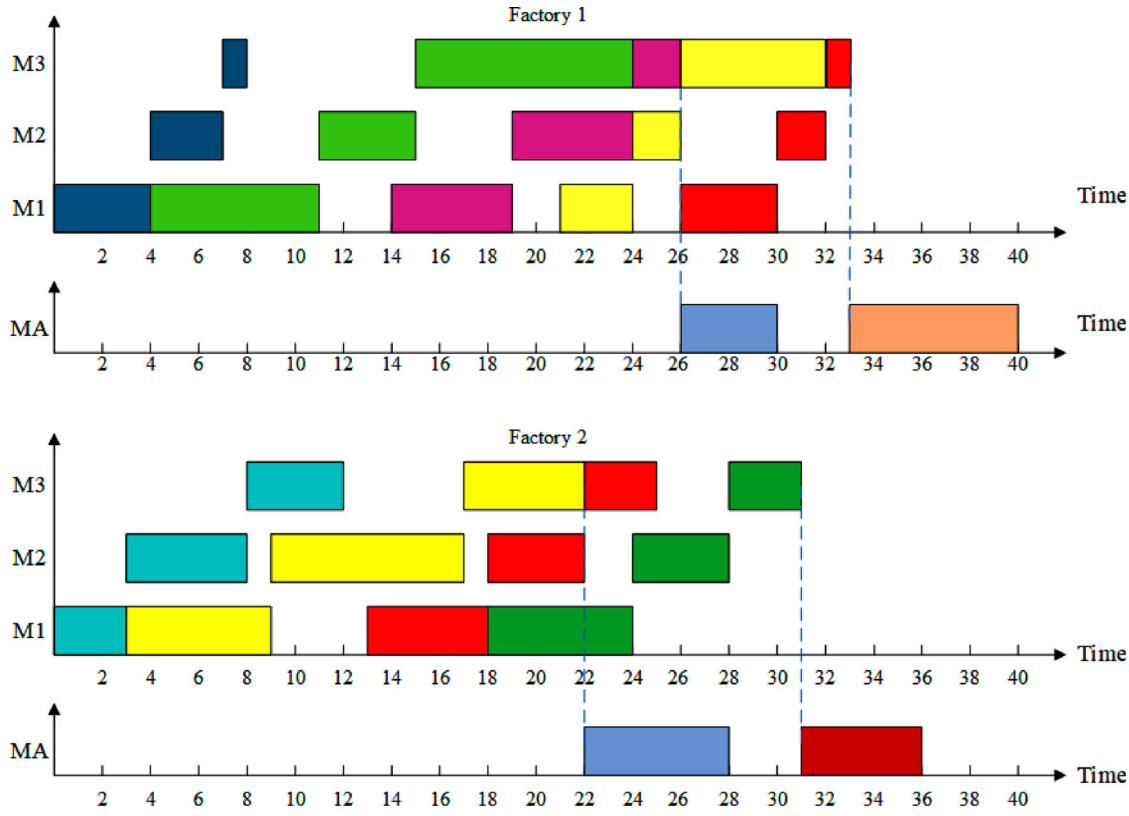


Figure 2. Gantt chart for the instance.

The main operation is shown in Figure 3, and the steps are as follows. Arrange the products in non-increasing order after calculating the assembly completion time of each product. f products are taken out from the sequence and assigned to f factories in the order of $F_1 \sim F_f$. In the second assignment, f products are taken from the remaining sequence and assigned to f factories in the order of $F_f \sim F_1$. The odd-numbered assignment is assigned according to the first allocation, and the even-numbered assignment is assigned according to the second allocation. This method evenly assigns products to factories and optimises resource allocation as much as possible.

The speed is set as three levels, including low, medium and high in the initialisation stage, and the processing speed of $O_{j,i}$ and the assembly speed of the products are randomly generated in V_1, V_2, V_3.

4.3. Sequence-related operation

The sequence of the product and the jobs is adjusted to optimise the assembly completion time of the product. Four strategies are used in BSO to generate new individuals, including two types of intra-cluster search and two types of inter-cluster search. Intra-clusters search increases the precision of the algorithm and helps the algorithm to find the optimal solution. Inter-cluster search enhances the rough searching ability of the algorithm and increases the probability of finding the global optimal solution. Four sequence-related operations are proposed imitating the individual generation strategy in BSO, including critical factory insert (CI), critical factory swap (CS), critical factory insert to other factories (CIO) and critical factory swap to other factories (CSO).

The internal operation of critical factories in Figure 4 (supplementary material) optimises the optimal solution found so far and avoids the blind operation of random multi-factories, reducing the computational complexity. Critical factory operates on all products from other factories instead of selecting a random factory, which reduces randomness and ensures the algorithm's ability to find excellent product sequences. The critical factory changes over the process of the four insert and swap operations, so the scope of the operations is all factories, which prevents the algorithm from missing potential optimal solutions. The detailed operation process is described as follows. f_c and f_o are used to represent critical factory and other factory, respectively.

(1) The products $\{P_1, P_2, \ldots, P_{h^c}\}$ in critical factory f_c are taken out in turn and inserted into any other possible location within the critical factory.

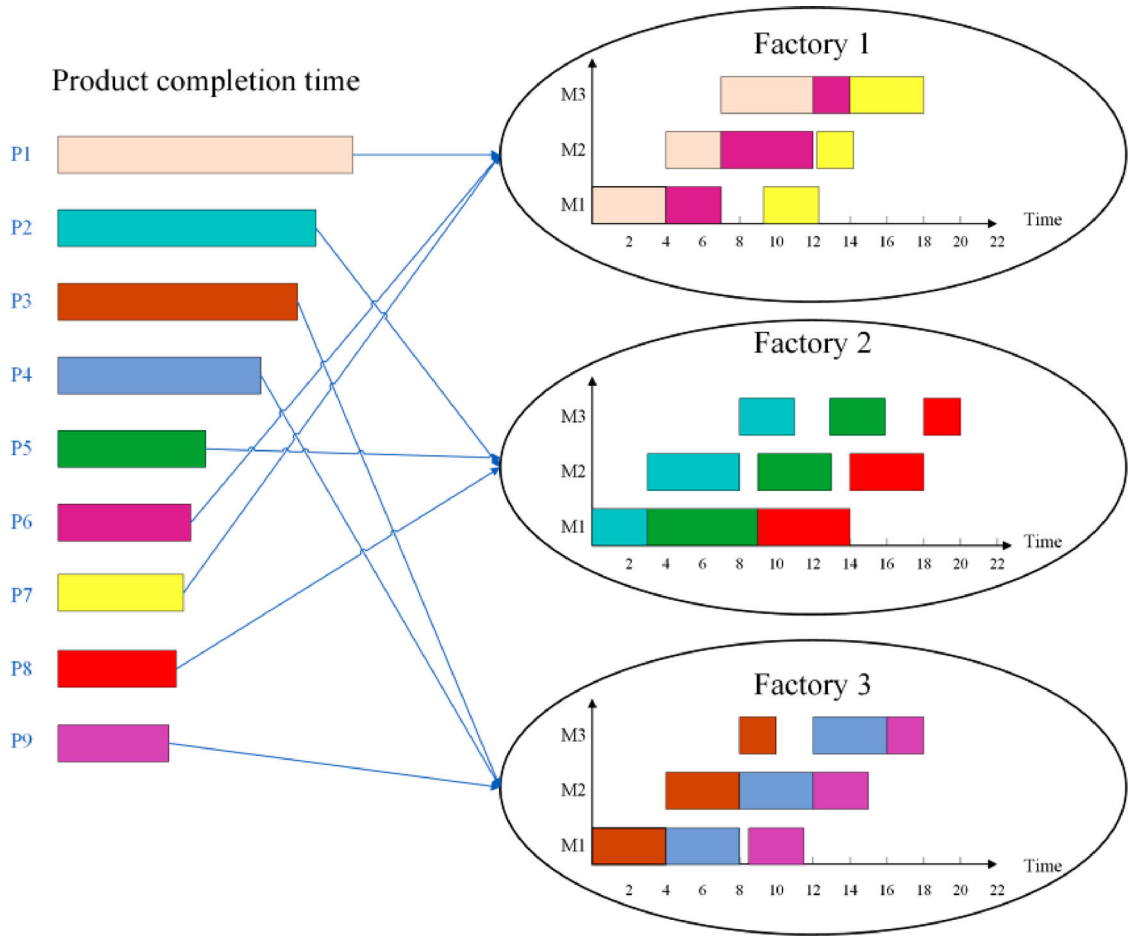


Figure 3. The product assignment method.

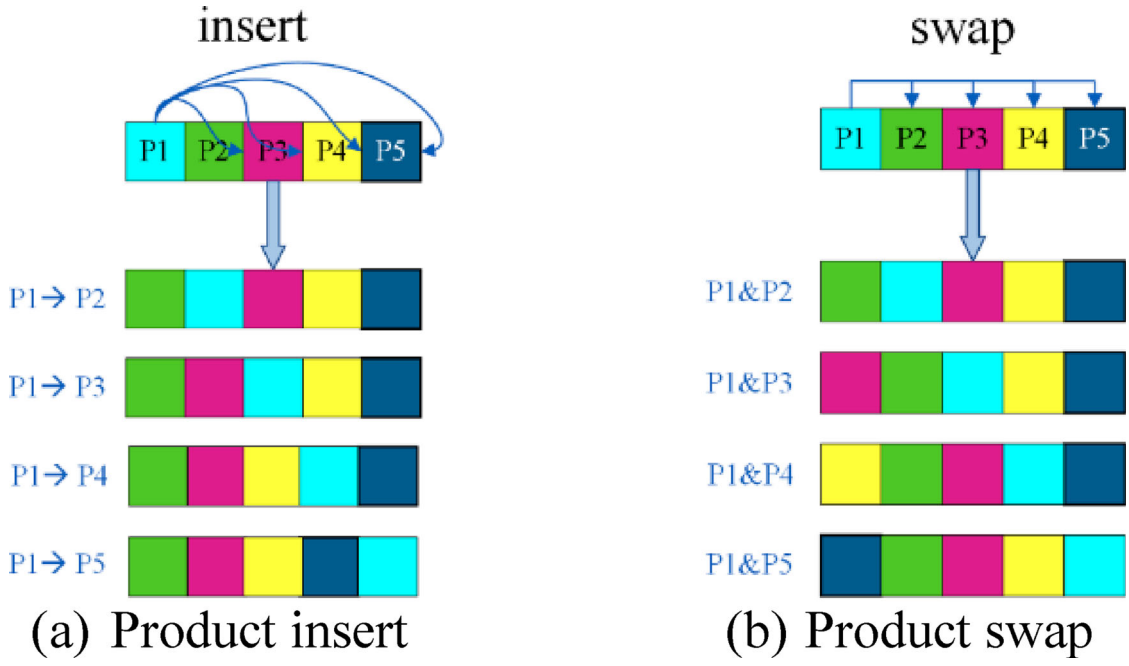


Figure 4. The operation in critical factory.

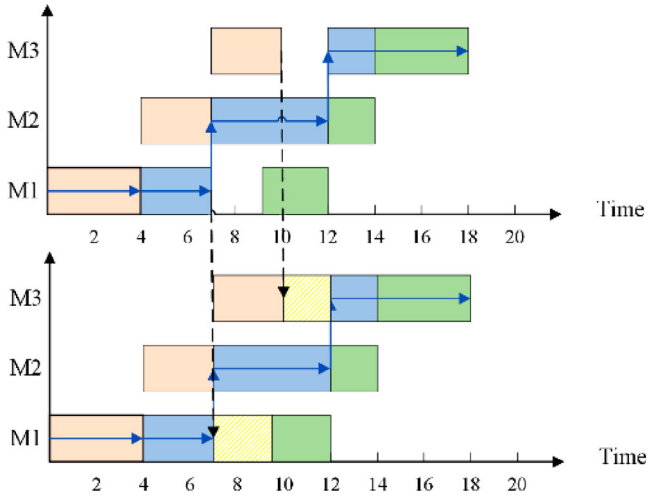


Figure 5. Reduce the speed of jobs on noncritical path.

(2) The products $\{P_1, P_2, \ldots, P_{h^c}\}$ in critical factory f_c are taken out in turn and exchanged with any other products remaining in the critical factory to obtain a new critical factory product sequence.
(3) The products $\{P_1, P_2, \ldots, P_{h^c}\}$ in critical factory f_c are taken out in turn and inserted into all possible locations of other factories f_o.
(4) The products $\{P_1, P_2, \ldots, P_{h^c}\}$ in critical factory f_c are taken out in turn and exchanged with all products in other factories f_o.

Q-learning is utilised to guide the selection of operations to avoid aimless searching of the four operations in the process of finding the optimal solution. Q-learning is a feedback-based method in reinforcement learning, and a certain behaviour at a certain moment is taken to get corresponding rewards (Xiao et al. 2020; Oliff et al. 2020). These actions and returns are established into a Q-table, according to which the algorithm determines the next action that obtain the optimal return. Successful individuals are defined as those whose assembly completion time is improved by using a certain operation. In the first N generations within the algorithm running time limit, the behaviour and return of successful individuals are used to construct the Q-table, where N is the total number of individuals in the pareto set.

$$\text{Q - table} = \begin{pmatrix} \text{oper}^1 & \text{score}^1 \\ \text{oper}^2 & \text{score}^2 \\ L & L \\ \text{oper}^i & \text{score}^i \end{pmatrix} \tag{25}$$

where, oper^i is the ith operation. oper^i is the rewards score for the ith operation,$i = \{1, 2, 3, 4\}$.

An operation is selected according to the Q-table matrix to update the current individual from the next generations. The score^i is updated after this iteration. score^i is updated as follows.

$$\text{score}(t+1)^i = \text{score}(t)^i + \varepsilon \tag{26}$$

$$\varepsilon = \frac{\sum_{n=1}^{\text{oper}^{i'}_n} |1|}{\sum_{n=1}^{\text{oper}^{i}_n} |1|} \tag{27}$$

where, oper^i_n is the number of iteration using operation oper^i. $\text{oper}^{i'}_n$ is the number of successful iterations after using operation oper^i.

4.4. Energy saving operations and resource balancing adjustment

High operating speed reduces the processing time of jobs, but leads to high energy consumption. Reducing processing speed reduces energy consumption, but leads to an increase in assembly completion time. In the no-wait flow shop scheduling problem, there is no waiting time when a job is processed on the machine, so idle time exists on the machines. The continuous working path with no idle time in the process from the beginning of the first job to the end of the last job is called critical path. The length of critical path in critical factory is assembly completion time. In order to reduce production energy consumption, two ways is introduced in this study to save energy consumption. Considering the characteristics of no-wait flow shop scheduling problem, non-critical operations in factories with the largest assembly completion time are slowed down to reduce energy consumption. In non-critical plants, random speed regulation is used for critical or non-critical operations. The path shown by the arrow is the critical path in Figure 5 (Supplementary Materials), and the processing time is slowed down to reduce energy consumption at the position of the yellow job block.

Procedure 1: Resource allocation balance

1. for $i = 1$ to F
2. calculate the $C_{max}(\pi^f_p)$
3. take the factory with maximum C_{max} as the critical factory
4. find the product with the least completion time $P_{\min}$ in critical factory
5. find the factory with minimum $C_{\max}$ as $F_{\min}$
6.end for
7.for $k = 1$ to $f_n + 1$
8. The $P_{\min}$ is inserted into $f_n + 1$ possible positions in factory $F_{\min}$, and the product assembly completion time $C_{\max}(\pi)$ is calculated.
9. if $C_{\max}(\pi) < \text{Min}$ then
10. $\text{Min} = C_{\max}(\pi)$
11. end if
12. end for
13. return π

A resource balancing method is proposed and the standard deviation of equipment utilisation rate between

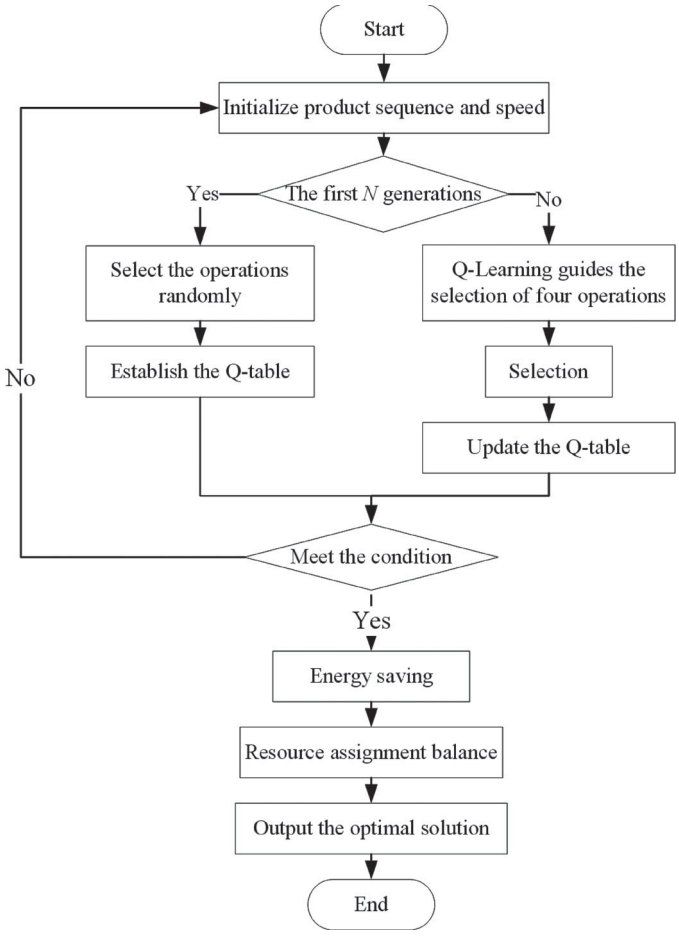


Figure 6. Algorithm flow chart.

factories is used to represent resource allocation balance. The idle time of other factories is large when the completion time of $f_{\max}$ is large, resulting in uneven utilisation of equipment between factories. Steps are as follows. Find the product $P_{\min}$ with the smallest processing time in the factory with the largest assembly completion time. Put $P_{\min}$ in the possible position of the factory $f_{\min}$ with the smallest assembly completion time. Obtain the product sequence with the balance of idle time between factories. The pseudocode is in Procedure 1. The flow chart of the algorithm is shown in Figure 6. The pseudo code of the proposed method is given in Procedure 2.

5. Experiments and analysis of the algorithm

81 instances in the test suite are selected to analyse the effectiveness of the strategies. The instances are composed of different numbers of factories $f = \{4, 6, 8\}$, jobs $n = \{100, 200, 500\}$, machines $m = \{5, 10, 20\}$ and products $P = \{30, 40, 50\}$. According to [14], the speed and energy consumption are set as $v = \{0.8, 1, 1.2\}, PP = \{0.6, 1, 1.5\}$. The algorithms are run on a PC with a 3.4

Procedure 2: The RLBSO for Energy-Efficient DANWFSP

1. input: termination condition, iteration times N
2. initialise product sequence and the speed of each operation
3. while the termination condition is not satisfied
4. if $t < N$
5. select the operations randomly
6. create the Q-table
7. else
8. use Q-learning guides the selection of the operations
9. select the sequence with minimum $C_{\max}(\pi)$ as the new sequence
10. update the Q-table
11. end if
12. end while
13. energy saving operation
14. resource allocation balance operation
15. output: the sequence

GHZ Intel (R), Core (TM) i7-6700 CPU, 8GB of RAM and 64-bit OS. The algorithms are run independently for five times to ensure the fairness, and the termination condition is $n \times m \times 10$ms.

The overall non-dominated vector generation (ONVG) and C Metric are used to evaluate the quality of the obtained pareto set. ONVG is the number of non-dominated individuals in the final pareto set. The C Metric is the dominant relationship between individuals in the two pareto sets A and B. The larger the C Metric is, the better the performance of the proof algorithm is. The calculation method of $C(A, B)$ is as follows.

$$C(A, B) = |\{b \in B | \exists a \in A, a \succ b \text{ or } a = b\}|/|B|$$

To test the influence of learning operation, energy-efficient strategy, and resource allocation balancing operation on algorithms, RPI is used to calculate the results, as shown below. The smaller the value of RPI is, the better the result of the algorithm is.

$$\text{RPI} = (T_r - T_b)/T_b * 100$$

where T_r is the value of the rth solution in the algorithm, and T_b is the best solution among all algorithms.

5.1. Effectiveness of the initialisation method

The CMetric is used to analyse the effectiveness of the initialisation strategy, measuring the dominance relationship between individuals in pareto sets with initialisation strategy (RL-I) and without initialisation strategy (RL-NI). From Table 2, the CMetric of RL-I (C(RLBSO, RLBSO − NI)) is larger than that of RL-NI (C(RLBSO − NI, RLBSO)), when the numbers of factories are 6 and 8 and the numbers of jobs are 200 and 500. In general, RL-I is superior to RL-NI on average at different factory and job sizes.

Table 2. The results of RL-Yes and RL-No.

	f	Yes	No	*n*	Yes	No
Initialisation method *CMetric*	4	0.162	**0.255**	100	0.206	**0.240**
	6	0.202	**0.256**	200	**0.204**	0.190
	8	**0.307**	0.084	500	**0.259**	0.166
	Average	**0.223**	0.199	Average	**0.223**	0.199
Q-learning RPI	4	**0.00**	4.31	100	**0.00**	7.08
	6	**0.00**	4.24	200	**0.00**	5.15
	8	**0.00**	4.70	500	**0.00**	3.43
	Average	**0.00**	4.41	Average	**0.00**	5.22
Energy saving RPI	4	**0.00**	0.12	100	**0.00**	0.37
	6	**0.00**	0.10	200	**0.00**	0.19
	8	**0.00**	0.19	500	**0.00**	0.07
	Average	**0.00**	0.14	Average	**0.00**	0.21
Resource allocation balance RPI	4	**0.00**	1.23	100	5.02	**0.00**
	6	1.48	**0.00**	200	**0.00**	4.24
	8	**0.00**	10.74	500	**0.00**	5.29
	Average	**0.49**	3.99	Average	**1.67**	3.18

Note: The bold represent the result is better than that of other values.

5.2. Effectiveness of Q-learning and its four operations

The algorithm that removes Q-learning and its operations is denoted as RL-NL, and the algorithm that retains Q-learning and its operations is denoted as RL-L. To verify the effectiveness of Q-learning guided operations, RL-L and RL-NL are compared in the objective of minimising the maximum assembly completion time. Classifications by factory and job are used, and Table 2 shows that the algorithm with Q-learning and its operations has a smaller RPI value in different number of factories and scale of jobs.

5.3. Effectiveness of energy saving strategy

The algorithm with energy saving strategy deleted (RL-NE) and the algorithm with energy saving strategy included (RL-E) are compared to analyse the optimisation performance of energy saving strategy on TEC. The RPI method is used to calculate the values of the two algorithms under different factory numbers and different job sizes. The results in Table 2 show that the RPI value of RL-E is smaller than that of RL-NE. Therefore, the energy saving strategy is effective in optimising TEC objective. The interval plot of RL-E and RL-NE is shown in Figure 7(a).

5.4. Effectiveness of resource allocation balance strategy

The algorithm that deleted the resource allocation balance strategy (denoted as RL-NR) is compared with the algorithm that added the resource allocation balance strategy (denoted as RL-R) to verify the optimisation ability of the resource allocation balance policy on the RAB objective. When the number of factories is 4 and 8, the RPI values of RAB are less than that of RL-NR in Table 2. When the job size is 200 and 500, the RPI values of RAB are less than that of RL-NR. The mean RPI of RL-R is smaller than that of RL-NR in both categories. The interval plot is shown in Figure 7(b).

5.5. Summary and analysis

In general, the operations have a significant impact on the outcome of the problem, optimising the maximum completion time, energy consumption and resource allocation balance. The clustering-based initialisation method arranges the jobs with long processing time reasonably and optimises the overall processing efficiency. In the process of searching for the optimal job sequence of distributed no-wait flow shop scheduling problem, reinforcement learning guides the next step strategy selection according to whether the job sequence value is found. The learning selection method, which effectively avoids the random arrangement of the job sequence, increases the probability of finding the global optimal solution, and improves the efficiency of the algorithm. The energy saving operation adjusts the speed of the machine. The deceleration operation on the non-critical path, which makes reasonable use of the idle time of the machine, saves energy consumption and ensures the completion time. The parts with the shortest processing time in key factory are made small adjustment to ensure proper allocation and utilisation of resources in each factory.

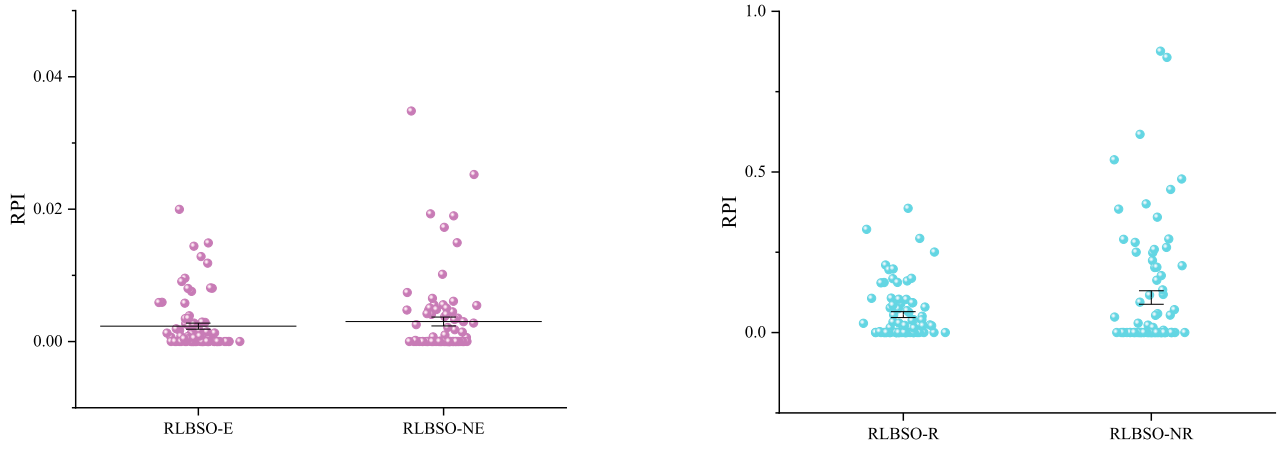
6. Comparison with the IG algorithm

6.1. Results of RLBSO and IG on DANWFSP

RLBSO and IG (Pan et al. 2019) are applied to solve the distributed two-stage assembly no-wait flow shop scheduling problem (DANWFSP) aiming at the maximum assembly completion time. Tests are performed on a test suite of 810 instances, with each algorithm running independently 20 times. The distributed no-wait flow shop scheduling problem test set with assembly procedure is used, which is the same as in the reference of the comparison algorithm IG. The stop criterion of the algorithm includes three levels, $T = n * m * \rho ms$, and $\rho = \{10, 20, 30\}$. Average relative percentage deviation (ARPD) is used to evaluate the performance of the algorithm as follows. The smaller the ARPD is, the better the performance of the algorithm is.

$$\mathrm{ARPD} = \frac{1}{\mathrm{run}} \sum_{r=1}^{\mathrm{run}} \frac{C^r - C^*}{C^*} \times 100$$

where, run is the independent running times of the experiment, and C^r is the solution of the instance



(a) RL-E and RL-NE (b)RL-R and RL-NR

Figure 7. Interval plot.

Table 3. The ARPD of different indicators under three levels.

		$\rho = 10$		$\rho = 20$		$\rho = 30$	
Indicator		RL	IG	RL	IG	RL	IG
f	4	**0.271**	1.356	**0.314**	1.404	**0.330**	1.342
	6	**0.074**	1.915	**0.098**	1.892	**0.078**	1.972
	8	**0.020**	2.674	**0.013**	2.630	**0.023**	2.573
n	100	**0.021**	3.517	**0.018**	3.477	**0.021**	3.448
	200	**0.041**	1.686	**0.047**	1.786	**0.070**	1.747
	500	**0.303**	0.741	**0.360**	0.664	**0.339**	0.692
m	5	**0.100**	2.081	**0.110**	2.063	**0.135**	1.946
	10	**0.124**	1.921	**0.179**	1.860	**0.166**	1.857
	20	**0.141**	1.942	**0.136**	2.004	**0.130**	2.085
P	30	**0.289**	0.689	**0.285**	0.762	**0.302**	0.717
	40	**0.066**	1.706	**0.111**	1.631	**0.100**	1.701
	50	**0.010**	3.550	**0.028**	3.533	**0.029**	3.470
Average		**0.122**	1.981	**0.142**	1.975	**0.143**	1.963

obtained when the algorithm reaches the stop condition in the *r*th experiment. C^* is the optimal solution of the two algorithms when they reach the stop criterion.

From Table 3, the ARPD value of RLBSO (RL) is less than that of IG under the three stop criteria in the case of factory grouping, job grouping, machine grouping and product grouping. The horizontal axis represents algorithms at different scales and the vertical axis represents ARPD values in Figure 8. From the interval plots, RLBSO values are all less than IG when grouped by different indicators under different termination criteria.

6.2. Results of RLBSO and IG on energy-efficient DANWFSP

The performance of RLBSO and IG (Pan et al. 2019) in solving energy-efficient DANWFSP is compared. The speed and energy consumption are set to $v = \{0.8, 1, 1.2\}, PP = \{0.6, 1, 1.5\}$. 810 instances are divided into 81 groups with 10 instances in each group, according to the number of factories $F = \{4, 6, 8\}$, number of jobs $n = \{100, 200, 500\}$, number of machines $m = \{5, 10, 20\}$ and number of products $P = \{30, 40, 50\}$. The algorithm is independently run for 20 times on each instance. The maximum CPU time is used as the stop criterion and the stop time is $T = n * m * \rho ms$, where $\rho = \{10, 20, 30\}$. Three groups of experiments are designed according to three different time levels.

*C*Metric and ONVG are used to evaluate the performance of RLBSO and IG, with large values indicating good performance of the algorithm. The *C* Metric results of the algorithms based on the three termination criteria are shown in Table 4, where RLBSO = *C*(RLBSO,IG) and IG = *C*(IG,RLBSO). In the line charts in Figure 9 (supplementary material), the first instance of each group is selected from 81 groups to demonstrate the algorithm performance under different processing conditions. The performance of the algorithms is different under different problem sizes in Figure 10. The *C* Metric of RLBSO is mostly higher than that of IG.

The pareto front is an important index to measure multi-objective optimisation. In the three-objective optimisation problem, the pareto front is a surface. The closer the solution is to the origin in three-dimensional coordinate space, the better the pareto front is. 16 large-scale instances are selected to draw the pareto front, where the red sphere represents the solution of RLBSO and the black sphere represents the solution of IG. The red sphere is closer to the origin of coordinates than the black sphere from Figure 10, which means that the solution of RLBSO dominates the solution of IG in the solution space.

The ONVG values of RLBSO and IG are given in Table 5. The ONVG value of RLBSO is larger

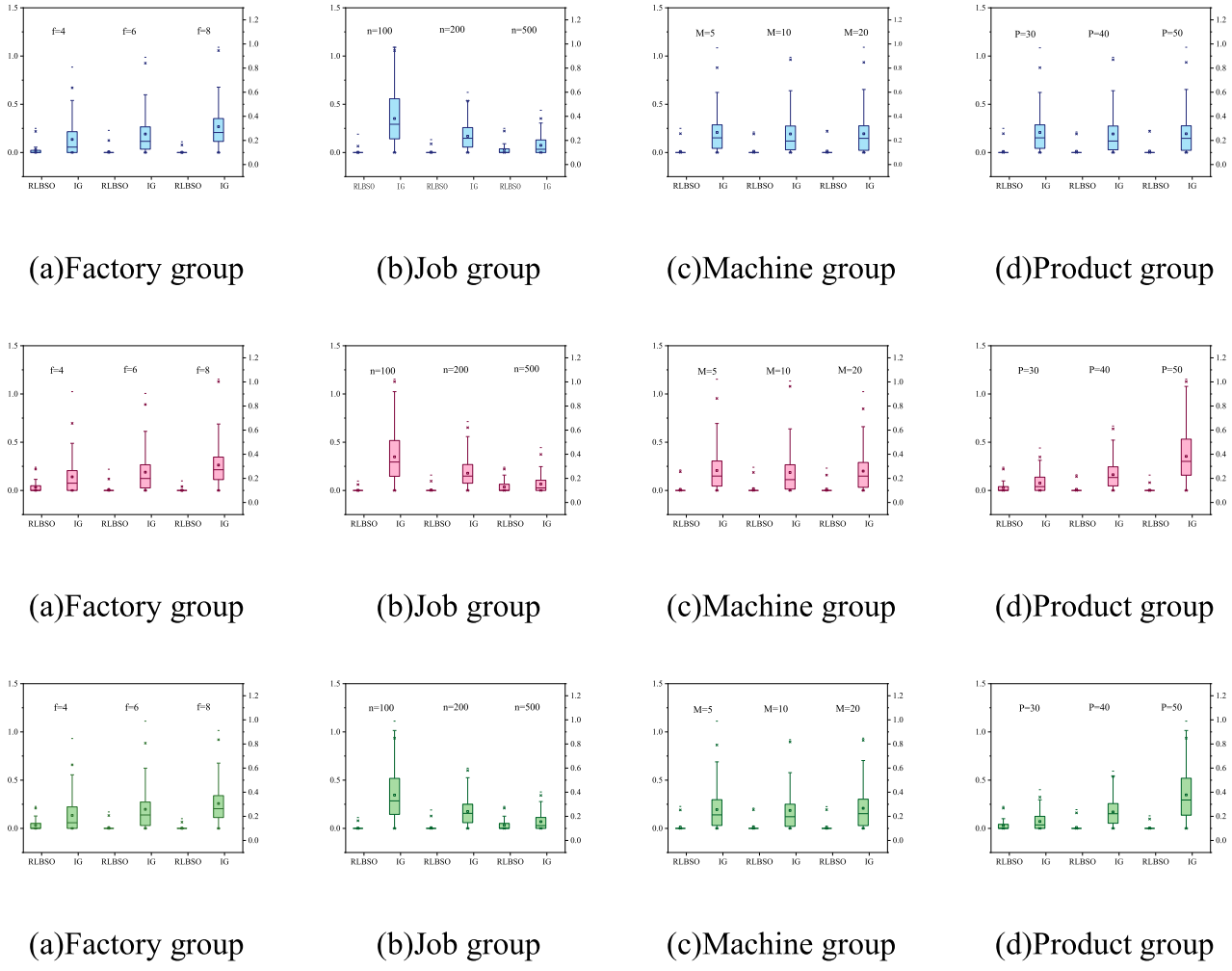


Figure 8. The boxplot of RLBSO and IG with $\rho = 10, 20, 30$.

Table 4. The *C*Metric of RLBSO and IG with $\rho = 10, 20, 30$ under four indicators.

F	RL	IG	*n*	RL	IG	*m*	RL	IG	*P*	RL	IG
4	**0.488**	0.065	100	**0.548**	0.018	5	**0.436**	0.027	30	**0.316**	0.041
6	**0.621**	0.018	200	**0.450**	0.031	10	**0.418**	0.030	40	**0.431**	0.029
8	**0.768**	0.006	500	**0.304**	0.035	20	**0.447**	0.028	50	**0.555**	0.015
Average	**0.626**	0.030	Average	**0.434**	0.028	Average	**0.434**	0.028	Average	**0.434**	0.028
4	**0.482**	0.072	100	**0.556**	0.020	5	**0.431**	0.030	30	**0.325**	0.046
6	**0.617**	0.019	200	**0.448**	0.035	10	**0.406**	0.033	40	**0.428**	0.031
8	**0.775**	0.007	500	**0.289**	0.037	20	**0.456**	0.030	50	**0.540**	0.016
Average	**0.625**	0.033	Average	**0.431**	0.031	Average	**0.431**	0.031	Average	**0.431**	0.031
4	**0.475**	0.072	100	**0.557**	0.020	5	**0.421**	0.030	30	**0.318**	0.044
6	**0.630**	0.018	200	**0.448**	0.034	10	**0.414**	0.032	40	**0.440**	0.028
8	**0.756**	0.008	500	**0.293**	0.038	20	**0.463**	0.029	50	**0.539**	0.020
Average	**0.621**	0.033	Average	**0.432**	0.031	Average	**0.432**	0.031	Average	**0.432**	0.031

Note: The bold represent the result is better than that of other values.

than that of IG, which verifies that the number of non-dominant solutions in RLBSO is larger than that in IG.

Wilcoxon's test results are listed in Table 6, which indicate that RLBSO outperforms IG under 95% confidence intervals ($\alpha = 0.05$) under three stop criteria and different factory grouping, job grouping, machine grouping and product grouping sizes.

According to the above numerical results and statistical analysis, the performance of RLBSO is better than IG in solving the energy-efficient DANWFSP.

7. Conclusion and future work

The RLBSO is introduced to address energy-Efficient DANWFSP, which aims to minimise the $C_{\max}$, TEC and

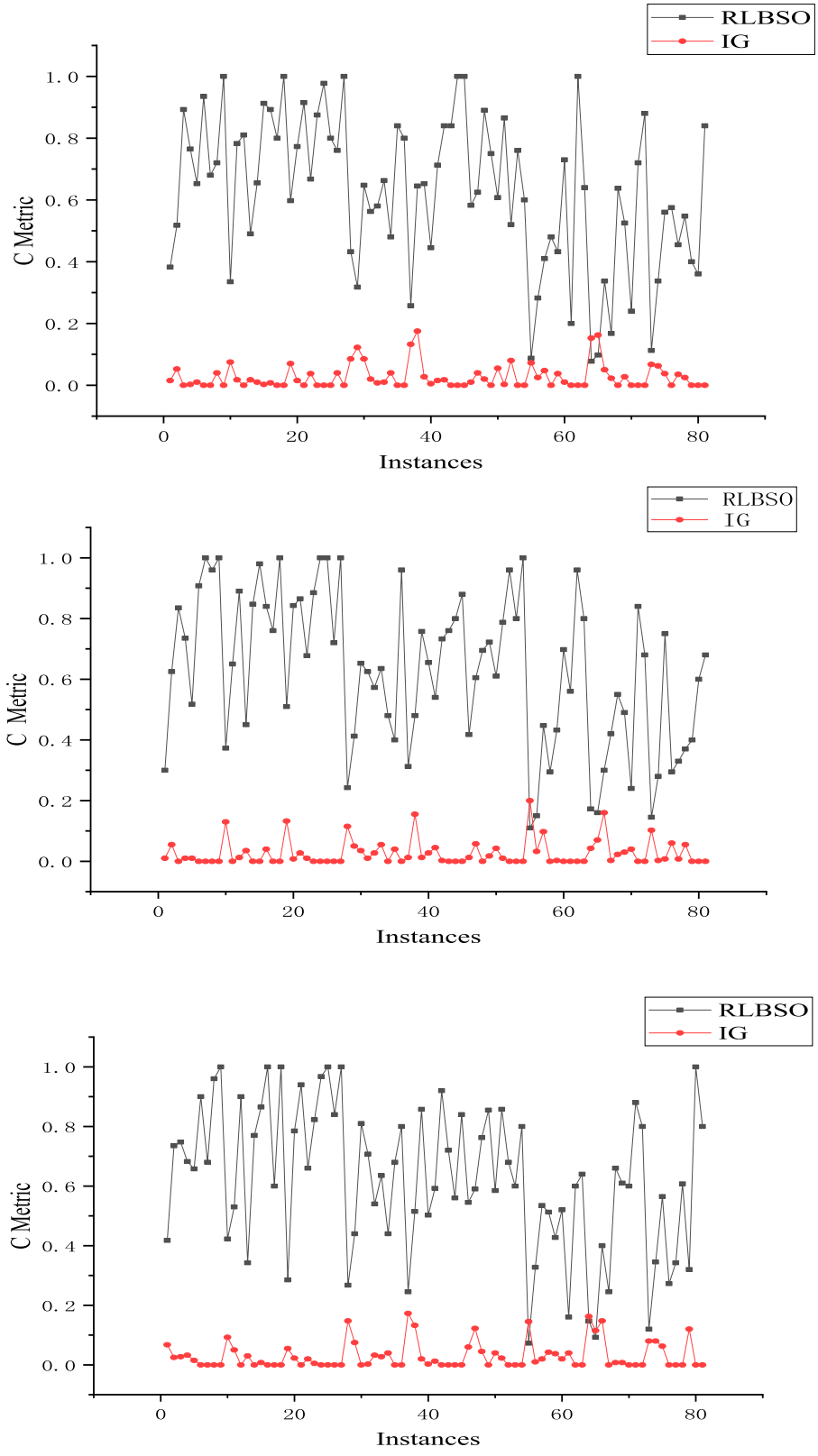


Figure 9. Line chart of RLBSO and IG with $\rho = 10, 20, 30$ on 81 instances.

RAB. A product assignment method in objective space is designed according to the characteristics of distributed flow shop scheduling problem and the core mechanism of BSO based on clustering. Experimental results show that the initialisation strategy significantly optimises the three objectives. Q-learning mechanism is introduced to

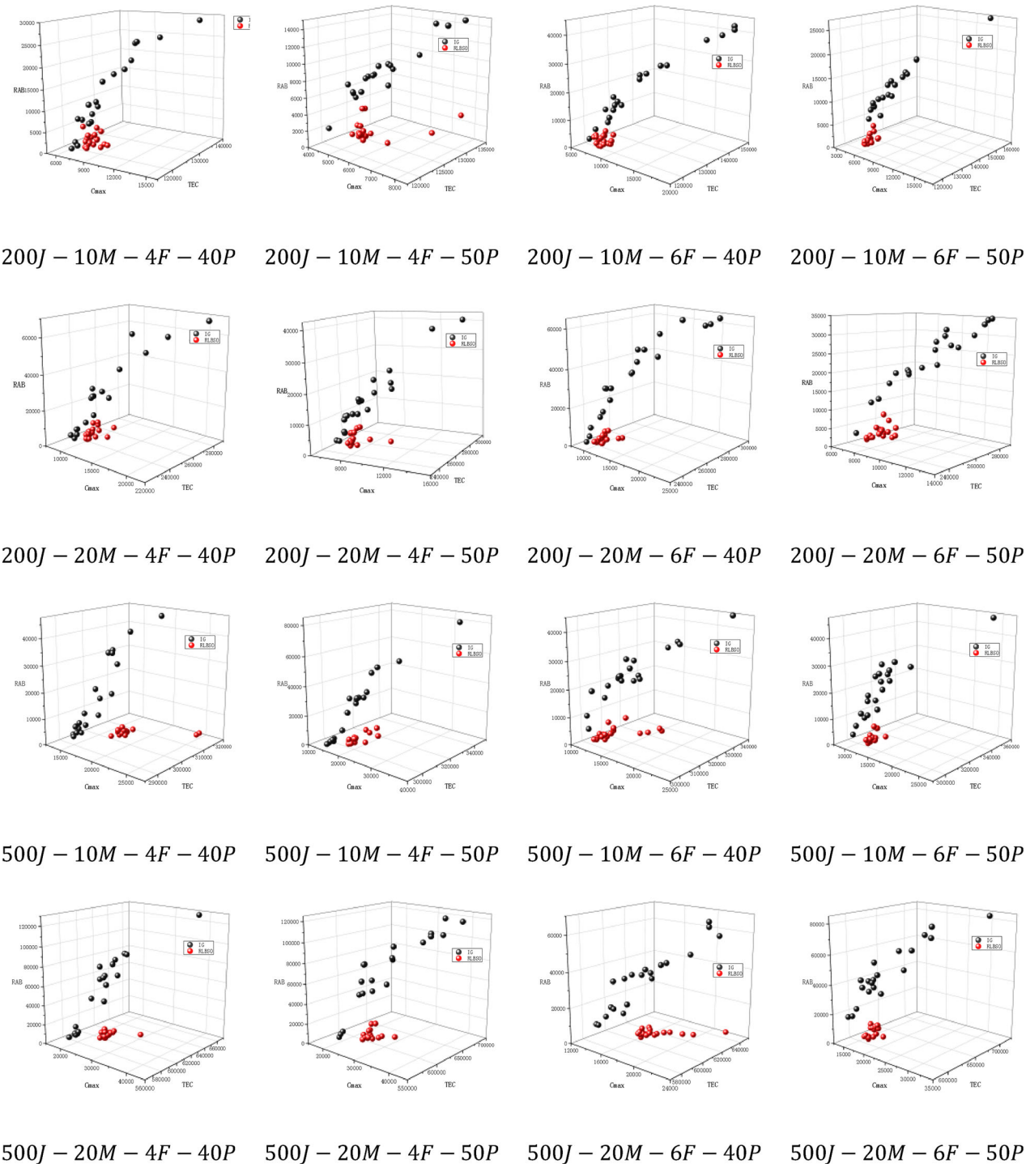


Figure 10. Pareto front on 16 large-scale instances.

guide the use of operations, avoiding the blindness of the solution process. The effectiveness analysis experiments show that both energy saving and resource allocation balancing strategies have good performance for the algorithm. Numerical comparison and statistical analysis results show that RLBSO has better solving ability than comparison algorithm on energy-efficient DANWFSP.

Constraints and optimisation indicators need to be refined to adapt to practical problems as actual production problems become more and more complex. The new constraints take into account the heterogeneity of distributed factories, limited wait times, and sort-related settings. Optimisation indicators need to take into account indicators that affect social production, such as the uniformity of workload of various production lines,

Table 5. The ONVG of RLBSO and IG.

Index		$\rho = 10$		$\rho = 20$		$\rho = 30$	
		RLBSO	IG	RLBSO	IG	RLBSO	IG
n	100	**2783**	2139	**2569**	2075	**2816**	2071
	200	**2750**	2252	**2796**	2022	**2852**	2112
	500	**2850**	2208	**2759**	2186	**2727**	2323
m	5	**2838**	2255	**2924**	2220	**2818**	2303
	10	**2816**	2169	**2633**	2127	**2797**	2071
	20	**2729**	2175	**2567**	1936	**2780**	2132
P	30	**2756**	2251	**2747**	2107	**2853**	2236
	40	**2793**	2196	**2688**	2001	**2749**	2216
	50	**2834**	2152	**2689**	2175	**2793**	2054
Average		**2794**	2200	**2708**	2094	**2798**	2169

Note: The bold represent the result is better than that of other values.

Table 6. The Wilcoxon test of RLBSO and IG with $\rho = 10, 20, 30$.

	RLBSO vs. IG		R^+	R^-	Z	p-value	α
$\rho = 10$	*f*	4	505.5	35809.5	−1.38E+01	1.90E−43	Yes
		6	0.0	36585.0	−1.42E+01	4.92E−46	Yes
		8	5.0	36580.0	−1.43E+01	3.54E−46	Yes
	n	100	1.0	36584.0	−1.43E+01	4.34E−46	Yes
		200	7.0	36578.0	−1.42E+01	5.27E−46	Yes
		500	468.0	35847.0	−1.39E+01	1.26E−43	Yes
	m	5	82.0	36503.0	−1.42E+01	1.21E−45	Yes
		10	62.5	36252.5	−1.42E+01	1.42E−45	Yes
		20	84.5	36500.5	−1.42E+01	1.24E−45	Yes
	P	30	505.5	35809.5	−1.38E+01	1.89E−43	Yes
		40	5.0	36580.0	−1.42E+01	5.14E−46	Yes
		50	0.0	36585.0	−1.42E+01	4.66E−46	Yes
$\rho = 20$	*f*	4	556.5	36028.5	−1.38E+01	2.21E−43	Yes
		6	0.0	36585.0	−1.42E+01	4.93E−46	Yes
		8	1.0	36584.0	−1.43E+01	3.49E−46	Yes
	n	100	1.0	36584.0	−1.42E+01	4.51E−46	Yes
		200	20.0	36565.0	−1.42E+01	6.10E−46	Yes
		500	442.5	36142.5	−1.39E+01	6.41E−44	Yes
	m	5	59.0	36526.0	−1.42E+01	9.41E−46	Yes
		10	127.5	36457.5	−1.41E+01	1.99E−45	Yes
		20	46.0	36539.0	−1.42E+01	7.97E−46	Yes
	P	30	459.0	36126.0	−1.39E+01	7.67E−44	Yes
		40	0.0	36585.0	−1.42E+01	4.88E−46	Yes
		50	0.0	36585.0	−1.42E+01	4.53E−46	Yes
$\rho = 30$	*f*	4	823.5	35761.5	−1.36E+01	3.88E−42	Yes
		6	1.0	36584.0	−1.42E+01	4.98E−46	Yes
		8	0.0	36585.0	−1.43E+01	3.63E−46	Yes
	n	100	3.0	36582.0	−1.42E+01	4.76E−46	Yes
		200	52.0	36533.0	−1.42E+01	8.71E−46	Yes
		500	554.5	36030.5	−1.38E+01	2.16E−43	Yes
	m	5	133.0	36452.0	−1.41E+01	2.14E−45	Yes
		10	127.5	36457.5	−1.41E+01	2.01E−45	Yes
		20	87.0	36498.0	−1.42E+01	1.27E−45	Yes
	P	30	650.5	35934.5	−1.37E+01	6.08E−43	Yes
		40	5.0	36580.0	−1.42E+01	5.17E−46	Yes
		50	2.0	36583.0	−1.42E+01	4.67E−46	Yes

total processing costs, energy consumption, resource utilisation, and environmental impact. In the future, the dynamic distributed intelligent scheduling problem in heterogeneous production lines and uncertain machine fault environments will be studied. Knowledge-driven collaborative swarm intelligence algorithm will be utilised to solve the energy saving distributed no-wait flow shop scheduling problem combining with the characteristics and rules of the problem.

Data availability statement

The data are openly available in 'CSDN' at https://download.csdn.net/download/huxt826/55586044.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was financially supported by the National Natural Science Foundation of China under grant 62063021. It was also supported by the Key talent project of Gansu Province (ZZ2021G50700016), the Key Research Programs of Science and Technology Commission Foundation of Gansu Province (21YF5WA086), Lanzhou Science Bureau project (2018-rc-98), and Project of Gansu Natural Science Foundation (21JR7RA204), respectively.

Notes on contributors

Fuqing Zhao received the B.Sc. and Ph.D. degrees from the Lanzhou University of Technology, Lanzhou, China, in 1994 and 2006, respectively. Since1998, he has been with the School of Computer Science Department, Lanzhou University of Technology, Lanzhou, China, where he became a Full Professor in 2012. He has been as the post Doctor with the State Key Laboratory of Manufacturing System Engineering, Xi'an Jiaotong University, Xi'an, China in 2009. He has been as a visiting scholar in Exeter Manufacturing Enterprise Center in Exeter University and Georgia Tech Manufacturing Institute in Georgia Institute of Technology from 2008 to 2019 and 2014 to 2015, respectively. He has authored two academic book and over 90 refereed papers. His current research interests include intelligent optimisation and scheduling.

Xiaotong Hu received the B.S. from School of Computer and Communication, Lanzhou University of Technology, Lanzhou, China, in 2018, where she is currently pursuing the M.S. degree in computer application technology. Her current research interests include intelligent optimisation and scheduling algorithms.

Ling Wang received the B.Sc. in automation and Ph.D. in control theory and control engineering from Tsinghua University, Beijing, China, in 1995 and 1999, respectively. Since 1999, he has been with the Department of Automation, Tsinghua University, where he became a full professor in 2008. His current research interests include intelligent optimisation and production scheduling. He has authored five academic books and more than 300 refereed papers. Professor Wang is a recipient of the National Natural Science Fund for Distinguished Young Scholars of China, the National Natural Science Award (Second Place) in 2014, the Science and Technology Award of Beijing City in 2008, the

Natural Science Award (First Place in 2003, and Second Place in 2007) nominated by the Ministry of Education of China. Professor Wang now is the Editor-in-Chief of *International Journal of Automation and Control* and the Associate Editor of *IEEE Transactions on Evolutionary Computation*, *Swarm and Evolutionary Computation*, etc.

Tianpeng Xu graduated from the School of Information Science and Engineering of Lanzhou University in 2005. He received the B.S. from School of Electrical and Information, Lanzhou University of Technology, Lanzhou, China, in 2013. From then on, he has been working at Lanzhou University of Technology. At present, he is pursuing his Ph.D. in manufacturing information system, and his current main research direction is intelligent optimisation and scheduling.

Ningning Zhu received the M.S. degree from School of Computer and Communication, Lanzhou University of Technology, Lanzhou, China, in 2012, where she is currently pursuing the Ph.D. degree in manufacturing information system. Her current research interests include intelligent optimisation and scheduling algorithms.

Jonrinaldi received the Ph.D. degree from University of Exeter, Exeter, UK in 2012. He is now the chair at the Department of Industrial Engineering, Universitas Andalas, Indonesia. He had published more than 40 refereed papers, and his research interests include logistics, inventory management, production/operations and supply chain optimisation.

ORCID

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

References

Aydilek, Asiye, Harun Aydilek, and Ali Allahverdi. 2017. "Minimising Maximum Tardiness in Assembly Flowshops with Setup Times." *International Journal of Production Research* 55 (24): 7541–7565. doi:10.1080/00207543.2017.1387300.

Bhatt, Ashutosh, Priti Dimri, and Ambika Aggarwal. 2020. "Self-Adaptive Brainstorming for Jobshop Scheduling in Multicloud Environment." *Software – Practice and Experience* 50 (8): 1381–1398. doi:10.1002/spe.2819.

Chen, Jing fang, Ling Wang, and Zhi ping Peng. 2019. "A Collaborative Optimization Algorithm for Energy-Efficient Multi-Objective Distributed No-Idle Flow-Shop Scheduling." *Swarm and Evolutionary Computation* 50 (April): 100557. doi:10.1016/j.swevo.2019.100557.

Deng, Jin, Ling Wang, Sheng Yao Wang, and Xiao Long Zheng. 2016. "A Competitive Memetic Algorithm for the Distributed Two-Stage Assembly Flow-Shop Scheduling Problem." *International Journal of Production Research* 54 (12): 3561–3577. doi:10.1080/00207543.2015.1084063.

Dong, Jianming, Hong Pan, Cunkui Ye, Weitian Tong, and Jueliang Hu. 2021. "No-Wait Two-Stage Flowshop Problem with Multi-Task Flexibility of the First Machine." *Information Sciences* 544 (6): 25–38. doi:10.1016/j.ins.2020.06.052.

Gao, Kaizhou, Yun Huang, Ali Sadollah, and Ling Wang. 2020. "A Review of Energy-Efficient Scheduling in Intelligent Production Systems." *Complex & Intelligent Systems* 6 (2): 237–249. doi:10.1007/s40747-019-00122-6.

Guo, Yinan, Huan Yang, Meirong Chen, Dunwei Gong, and Shi Cheng. 2020. "Grid-Based Dynamic Robust Multi-Objective Brain Storm Optimization Algorithm." *Soft Computing* 24 (10): 7395–7415. doi:10.1007/s00500-019-04365-w.

Han, Yuyan, Junqing Li, Hongyan Sang, Yiping Liu, Kaizhou Gao, and Quanke Pan. 2020. "Discrete Evolutionary Multi-Objective Optimization for Energy-Efficient Blocking Flow Shop Scheduling with Setup Time." *Applied Soft Computing Journal* 93: 106343. doi:10.1016/j.asoc.2020.106343.

Heger, J., and T. Voss. 2021. "Dynamically Adjusting the *k*-Values of the ATCS Rule in a Flexible Flow Shop Scenario with Reinforcement Learning." *International Journal of Production Research* 4: 1–15.

Hosseini, S. M., R. Carli, and M. Dotoli. 2020. "Robust Optimal Energy Management of a Residential Microgrid Under Uncertainties on Demand and Renewable Power Generation." *IEEE Transactions on Automation Science and Engineering* 99: 1–20.

Jiang, En da, and Ling Wang. 2019. "An Improved Multi-Objective Evolutionary Algorithm Based on Decomposition for Energy-Efficient Permutation Flow Shop Scheduling Problem with Sequence-Dependent Setup Time." *International Journal of Production Research* 57 (6): 1756–1771. doi:10.1080/00207543.2018.1504251.

Kemmoé-Tchomté, S., D. Lamy, and N. Tchernev. 2017. "An Effective Multi-Start Multi-Level Evolutionary Local Search for the Flexible Job-Shop Problem." *Engineering Applications of Artificial Intelligence* 62: 80–95. doi:10.1016/j.engappai.2017.04.002.

Kuhnle, Andreas, Marvin Carl May, Louis Schaefer, and Gisela Lanza. 2021. "Explainable Reinforcement Learning in Production Control of Job Shop Manufacturing System." *International Journal of Production Research*. doi:10.1080/00207543.2021.1972179.

Lee, Jun-Ho, and Hyun-Jung Kim. 2021. "Reinforcement Learning for Robotic Flow Shop Scheduling with Processing Time Variations." *International Journal of Production Research* 60 (7): 2346–2368. doi:10.1080/00207543.2021.1887533.

Lei, Deming, Bin Su, and Ming Li. 2020. "Cooperated Teaching-Learning-Based Optimisation for Distributed Two-Stage Assembly Flow Shop Scheduling." *International Journal of Production Research*. doi:10.1080/00207543.2020.1836422.

Li, Wenhan, Xiaolong Chen, Junqing Li, Hongyan Sang, Yuyan Han, and Shubo Du. 2022. "An Improved Iterated Greedy Algorithm for Distributed Robotic Flowshop Scheduling with Order Constraints." *Computers & Industrial Engineering* 164. doi:10.1016/j.cie.2021.107907.

Li, Yuan Zhen, Quan Ke Pan, Rubén Ruiz, and Hong Yan Sang. 2021. "A Referenced Iterated Greedy Algorithm for the Distributed Assembly Mixed No-Idle Permutation Flowshop Scheduling Problem with the Total Tardiness Criterion." *Knowledge-Based Systems* 239: 108036.

Li, Jun Qing, Hui Yu, Xiaolong Chen, Wenhan Li, Yu Du, and Yu Yan Han. 2020. "An Improved Brain Storm Optimization Algorithm for Fuzzy Distributed Hybrid Flowshop Scheduling with Setup Time." In *GECCO 2020 Companion – Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, 275–276. doi:10.1145/3377929.3389986.

Liu, Qi, Ruichao Mo, Xiaolong Xu, and Xu Ma. 2020. "Multi-Objective Resource Allocation in Mobile Edge Computing Using PAES for Internet of Things." *Wireless Networks* 3: 1–13. doi:10.1007/s11276-020-02409-w.

Liu, Xiao Fang, Zhi Hui Zhan, Jeremiah D. Deng, Yun Li, Tianlong Gu, and Jun Zhang. 2018. "An Energy Efficient Ant Colony System for Virtual Machine Placement in Cloud Computing." *IEEE Transactions on Evolutionary Computation* 22 (1): 113–128. doi:10.1109/TEVC.2016.2623803.

Ma, Lianbo, Shi Cheng, and Yuhui Shi. 2020. "Enhancing Learning Efficiency of Brain Storm Optimization via Orthogonal Learning Design." *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 99: 1–20. doi:10.1109/tsmc.2020.2963943.

Mao, Jia Yang, Quan Ke Pan, Zhong Hhua Miao, and Liang Gao. 2020. "An Effective Multi-Start Iterated Greedy Algorithm to Minimize Makespan for the Distributed Permutation Flowshop Scheduling Problem with Preventive Maintenance." *Expert Systems with Applications* 169 (1): 114495.

Meng, Leilei, Chaoyong Zhang, Xinyu Shao, Yaping Ren, and Caile Ren. 2019. "Mathematical Modelling and Optimisation of Energy-Conscious Hybrid Flow Shop Scheduling Problem with Unrelated Parallel Machines." *International Journal of Production Research* 57 (4): 1119–1145. doi:10.1080/00207543.2018.1501166.

Miyata, Hugo Hissashi, and Marcelo Seido Nagano. 2021. "Optimizing Distributed No-Wait Flow Shop Scheduling Problem with Setup Times and Maintenance Operations via Iterated Greedy Algorithm." *Journal of Manufacturing Systems* 61: 592–612.

Mozdgir, A., S. M. T. Fatemi Ghomi, F. Jolai, and J. Navaei. 2013. "Two-Stage Assembly Flow-Shop Scheduling Problem with Non-Identical Assembly Machines Considering Setup Times." *International Journal of Production Research* 51 (12): 3625–3642. doi:10.1080/00207543.2012.756151.

Oliff, H., Y. Liu, M. Kumar, and M. Williams. 2020. "Improving Human-Robot Interaction Utilizing Learning and Intelligence: A Human Factors-Based Approach." *IEEE Transactions on Automation Science and Engineering* 99: 1–14.

Pan, Quan Ke, Liang Gao, Li Xin-Yu, and Jose M. Framinan. 2019. "Effective Constructive Heuristics and Meta-Heuristics for the Distributed Assembly Permutation Flowshop Scheduling Problem." *Applied Soft Computing* 81. 105492 doi:10.1016/j.asoc.2019.105492.

Pan, Zixiao, Deming Lei, and Ling Wang. 2020. "A Knowledge-Based Two-Population Optimization Algorithm for Distributed Energy-Efficient Parallel Machines Scheduling." *IEEE Transactions on Cybernetics*, 1–13. doi:10.1109/tcyb.2020.3026571.

Pan, Jia Qi, Wen Qiang Zou, and Jun Hua Duan. 2018. "A Discrete Artificial Bee Colony for Distributed Permutation Flowshop Scheduling Problem with Total Flow Time Minimization." In *Chinese Control Conference, CCC*, 2018-July: 8379–8383. doi:10.23919/ChiCC.2018.8482716.

Shao, Weishi, Dechang Pi, and Zhongshi Shao. 2019. "Local Search Methods for a Distributed Assembly No-Idle Flow Shop Scheduling Problem." *IEEE Systems Journal* 13 (2): 1945–1956. doi:10.1109/JSYST.2018.2825337.

Shen, Jing Nan, Ling Wang, and Sheng Yao Wang. 2015. "A Bi-Population EDA for Solving the No-Idle Permutation Flow-Shop Scheduling Problem with the Total Tardiness Criterion." *Knowledge-Based Systems* 74 (jan.): 167–175. doi:10.1016/j.knosys.2014.11.016.

Shoaardebili, Niloofar, and Parviz Fattahi. 2015. "Multi-Objective Meta-Heuristics to Solve Three-Stage Assembly Flow Shop Scheduling Problem with Machine Availability Constraints." *International Journal of Production Research* 53 (3): 944–968. doi:10.1080/00207543.2014.948575.

Sun, Yuehong, Jianxiang Wei, Tingting Wu, Kelian Xiao, Jianyang Bao, and Ye Jin. 2020. "Brain Storm Optimization Using a Slight Relaxation Selection and Multi-Population Based Creating Ideas Ensemble." *Applied Intelligence* 50 (10): 3137–3161. doi:10.1007/s10489-020-01690-8.

Tan, Y., M. C. Zhou, Y. Zhang, X. Guo, and Y. Wang. 2020. "Hybrid Scatter Search Algorithm for Optimal and Energy-Efficient Steelmaking-Continuous Casting." *IEEE Transactions on Automation Science and Engineering* 99: 1–15.

Wang, Guangchen, Liang Gao, Xinyu Li, Peigen Li, and M. Fatih Tasgetiren. 2020a. "Energy-Efficient Distributed Permutation Flow Shop Scheduling Problem Using a Multi-Objective Whale Swarm Algorithm." *Swarm and Evolutionary Computation* 57 (November 2019): 100716. doi:10.1016/j.swevo.2020.100716.

Wang, Guangchen, Xinyu Li, Liang Gao, and Peigen Li. 2021. "Energy-Efficient Distributed Heterogeneous Welding Flow Shop Scheduling Problem Using a Modified MOEA/D." *Swarm and Evolutionary Computation* 62 (February): 100858. doi:10.1016/j.swevo.2021.100858.

Wang, Ling, and Zixiao Pan. 2021. "Scheduling Optimization for Flow-Shop Based on Deep Reinforcement Learning and Iterative Greedy Method." *Control and Decision* 36: 2609–2617.

Wang, Jing Jing, and L. Wang. 2018. "A Knowledge-Based Cooperative Algorithm for Energy-Efficient Scheduling of Distributed Flow-Shop." *IEEE Transactions on Systems Man & Cybernetics Systems* 50: 1805–1819.

Wang, Jing-jing, and Ling Wang. 2019. "Decoding Methods for the Flow Shop Scheduling with Peak Power Consumption Constraints." *International Journal of Production Research* 57 (10): 3200–3218. doi:10.1080/00207543.2019.1571252.

Wang, Shijin, Xiaodong Wang, Feng Chu, and Jianbo Yu. 2020b. "An Energy-Efficient Two-Stage Hybrid Flow Shop Scheduling Problem in a Glass Production." *International Journal of Production Research* 58 (8): 2283–2314. doi:10.1080/00207543.2019.1624857.

Wang, Shijin, Zhanguo Zhu, Kan Fang, Feng Chu, and Chengbin Chu. 2018. "Scheduling on a Two-Machine Permutation Flow Shop Under Time-of-Use Electricity Tariffs." *International Journal of Production Research* 56 (9): 3173–3187. doi:10.1080/00207543.2017.1401236.

Wu, Xiuli. 2019. "Brain Storm Optimization Algorithms for Flexible Job Shop Scheduling Problem." *Adaptation, Learning, and Optimization* 23. doi:10.1007/978-3-030-15070-9_10.

Wu, Chin-Chia, Du-Juan Wang, Shuenn-Ren Cheng, I-Hong Chung, and Win-Chin Lin. 2018. "A Two-Stage

Three-Machine Assembly Scheduling Problem with a Position-Based Learning Effect." *International Journal of Production Research* 56 (9): 3064–3079. doi:10.1080/00207543.2017.1401243.

Xiao, Q., C. Li, Y. Tang, and X. Chen. 2020. "Energy Efficiency Modeling for Configuration-Dependent Machining via Machine Learning: A Comparative Study." *IEEE Transactions on Automation Science and Engineering* 99: 1–14.

Xiong, Fuli, Keyi Xing, Feng Wang, Hang Lei, and Libin Han. 2014. "Minimizing the Total Completion Time in a Distributed Two Stage Assembly System with Setup Times." *Computers & Operations Research* 47 (Jul.): 92–105.

Yan, D., H. Cao, Y. Yu, Y. Wang, and X. Yu. 2020. "Single-Objective/Multiobjective Cat Swarm Optimization Clustering Analysis for Data Partition." *IEEE Transactions on Automation Science and Engineering* 17 (3): 1633–1646.

Yokoyama, Masao, and Daryl L. Santos. 2005. "Three-Stage Flow-Shop Scheduling with Assembly Operations to Minimize the Weighted Sum of Product Completion Times." *European Journal of Operational Research* 161 (3): 754–770. doi:10.1016/j.ejor.2003.09.016.

Yuksel, Damla, Mehmet Fatih Tasgetiren, Levent Kandiller, and Quan Ke Pan. 2020. "Metaheuristics for Energy-Efficient No-Wait Flowshops: A Trade-off between Makespan and Total Energy Consumption." 2020 *IEEE Congress on Evolutionary Computation, CEC 2020 – Conference Proceedings*. doi:10.1109/CEC48606.2020.9185554.

Yzla, B., A. Qkp, B. Jql, G. C. Liang, and D. Mft. 2021. "An Adaptive Iterated Greedy Algorithm for Distributed Mixed No-Idle Permutation Flowshop Scheduling Problems." *Swarm and Evolutionary Computation* 63: 100874.

Zhang, Biao, Quan Ke Pan, Liang Gao, Lei Lei Meng, Xin Yu Li, and Kun Kun Peng. 2020. "A Three-Stage Multiobjective Approach Based on Decomposition for an Energy-Efficient Hybrid Flow Shop Scheduling Problem." *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 50 (12): 4984–4999. doi:10.1109/TSMC.2019.2916088.

Zhang, Guanghui, Keyi Xing, and Feng Cao. 2018. "Scheduling Distributed Flowshops with Flexible Assembly and Set-up Time to Minimise Makespan." *International Journal of Production Research* 56 (9): 3226–3244. doi:10.1080/00207543.2017.1401241.

Zhang, W., D. Yang, H. Peng, W. Wu, and X. Shen. 2021. "Deep Reinforcement Learning Based Resource Management for DNN Inference in Industrial IoT." *IEEE Transactions on Vehicular Technology* 99: 1.

Zhao, F., X. He, and L. Wang. 2020. "A Two-Stage Cooperative Evolutionary Algorithm With Problem-Specific Knowledge for Energy-Efficient Scheduling of No-Wait Flow-Shop Problem." *IEEE Transactions on Cybernetics* 99: 1–13.

Zhao, Fuqing, Lixin Zhang, Jie Cao, and Jianxin Tang. 2021a. "A Cooperative Water Wave Optimization Algorithm with Reinforcement Learning for the Distributed Assembly No-Idle Flowshop Scheduling Problem." *Computers and Industrial Engineering* 153 (10): 107082. doi:10.1016/j.cie.2020.107082.

Zhao, Fuqing, Lexi Zhao, Ling Wang, and Houbin Song. 2020. "An Ensemble Discrete Differential Evolution for the Distributed Blocking Flowshop Scheduling with Minimizing Makespan Criterion." *Expert Systems with Applications* 160: 113678. doi:10.1016/j.eswa.2020.113678.

Zhao, Fuqing, Jinlong Zhao, Ling Wang, and Jianxin Tang. 2021c. "An Optimal Block Knowledge Driven Backtracking Search Algorithm for Distributed Assembly No-Wait Flow Shop Scheduling Problem." *Applied Soft Computing* 112: 107750. doi:10.1016/j.asoc.2021.107750.

Zheng, Xu, Shengchao Zhou, Rui Xu, and Huaping Chen. 2020. "Energy-Efficient Scheduling for Multi-Objective Two-Stage Flow Shop Using a Hybrid Ant Colony Optimisation Algorithm." *International Journal of Production Research* 58 (13): 4103–4120. doi:10.1080/00207543.2019.1642529.