COMPLEX SYSTEM MODELING AND SIMULATION
ISSN 2096-9929 02/06 pp 15–32
Volume 4, Number 1, March 2024
DOI: 10.23919/CSMS.2023.0021

Biased Bi-Population Evolutionary Algorithm for Energy-Efficient Fuzzy Flexible Job Shop Scheduling with Deteriorating Jobs

Libao Deng*, Yingjian Zhu, Yuanzhu Di, and Lili Zhang

Abstract: There are many studies about flexible job shop scheduling problem with fuzzy processing time and deteriorating scheduling, but most scholars neglect the connection between them, which means the purpose of both models is to simulate a more realistic factory environment. From this perspective, the solutions can be more precise and practical if both issues are considered simultaneously. Therefore, the deterioration effect is treated as a part of the fuzzy job shop scheduling problem in this paper, which means the linear increase of a certain processing time is transformed into an internal linear shift of a triangle fuzzy processing time. Apart from that, many other contributions can be stated as follows. A new algorithm called reinforcement learning based biased bi-population evolutionary algorithm (RB^{2}EA) is proposed, which utilizes Q-learning algorithm to adjust the size of the two populations and the interaction frequency according to the quality of population. A local enhancement method which combimes multiple local search stratgies is presented. An interaction mechanism is designed to promote the convergence of the bi-population. Extensive experiments are designed to evaluate the efficacy of RB^{2}EA, and the conclusion can be drew that RB^{2}EA is able to solve energy-efficient fuzzy flexible job shop scheduling problem with deteriorating jobs (EFFJSPD) efficiently.

Key words: bi-population evolutionary algorithm; Q-learning algorithm; fuzzy; deteriorating effect; energy; flexible job shop scheduling

1 Introduction

According to a 2019 report[1], in the previous five years, CO_2 emissions about energy consumption have increased by 1.3% each year on average. Industrial energy consumption and industrial processes account for a sizable share of global greenhouse gas emissions. For these reasons, energy efficiency in manufacturing has drawn considerable interest. In the background of green manufacturing, the field of job shop scheduling needs to seek solutions to reduce energy consumption. This provides various benefits for enterprises, including cost reduction, enhanced resource utilization, and promotion of sustainable development.

In recent years, the quantity of articles on energy efficiency and sustainability in diverse manufacturing systems, such as single machine[2, 3], two machines in line[4, 5], parallel machines[6, 7], and flow shop[8–13], has increased significantly. In particular, due to its significance in manufacturing, assembly line operations, maintenance and repair, etc., the flexible job shop scheduling problem (FJSP) has drawn great interest. In order to find conpromising solutions for total costs of production, Moon and Park[14] proposed mixed-integer programming and constraint programming approaches. Instead of makespan, Lei et al.[15] optimized total energy consumption and

• Libao Deng, Yingjian Zhu, and Yuanzhu Di are with the School of Information Science and Engineering, Harbin Institute of Technology, Weihai 264209, China. E-mail: denglibao_paper@163.com; meetzyj@gmail.com; diyz1121491761@163.com.
• Lili Zhang is with the Department of Computer Science, Maynooth University, Maynooth, W23 F2H6, Ireland, and also with the School of Computing, Dublin City University, Dublin, D09 V209, Ireland. E-mail: lili.zhang27@mail.dcu.ie.
* To whom correspondence should be addressed.
※ This article was recommended by Associate Editor Xinyu Li.
Manuscript received: 2023-09-06; revised: 2023-10-17; accepted: 2023-10-29

workload balance simultaneously. In Ref. [16], the number of machine turning on/offs was evaluated as a goal to minimize. Luo et al.[17] investigated the effect of various processing speeds on energy consumption in a flexible job shop, finding that faster running speeds consume more energy. A number of mixed-integer linear programming models were presented by Meng et al.[18] to optimize total energy consumption. In Ref. [19], energy consumption consists of four components: energy consumption when processing, standby energy consumption, energy consumption for setup, and energy consumption caused by transportation. Qin et al.[20] proposed an improved iterative greedy (IG) algorithm to optimize the energy consumption of job sequence. Many algorithms were designed to solve these energy-efficient scheduling problems. An improved genetic algorithm was proposed by Dai et al.[21] to solve the transportation constraints scheduling problem. Pan et al.[22] proposed an evolutionary algorithm with two populations, where the quality of each population is used to adjust the population size. Li et al.[23] adopted the idea of evaluating the quantity of population, combining Q-learning algorithm with multi-objective evolutionary algorithm based on decomposition (MOEA/D) to modify the number of neighborhood solutions dynamically. In order to address the FJSP with type-2 fuzzy processing time efficiently, an improved artificial immune system algorithm was presented by Li et al.[24] Zhao et al.[25] proposed a population-based iterated greedy algorithm to address distributed assembly no-wait flow shop scheduling problem.

However, many uncertainties in production mean that the current FJSP cannot meet the demand of the modern market. Thus, many scholars turned their attention to the FJSP with the fuzzy processing time (FFJSP). Sun et al.[26] combined genetic algorithm (GA) and particle swarm optimization (PSO) to optimize the fuzzy makespan. A hyper-heuristic approach based on backtracking search was presented by Lin[27], which can solve FFJSP effectively. A cooperated shuffled frog-leaping algorithm[28] was presented for simultaneously optimizing fuzzy makespan, fuzzy total energy consumption, and total agreement index. To solve FFJSP, Ref. [29] combined MOEA/D with a local searching strategy based on the success and failure memories. In these studies, the processing time is presented as a triangle fuzzy number. In Ref. [30], a weighted distance based approximation method was extended to schedule the operation sequence in a flow shop environment, utilizing interval-valued fuzzy sets in place of triangle fuzzy numbers (TFNs). The processing time was represented as type-2 fuzzy set in Ref. [31], thus uncertainties and constraints in the real-world factories can be more fully taken into account, which compensates for the drawback of the conventional fuzzy triangular number. Xi and Lei[32] investigated the distributed two-stage hybrid flow shop scheduling problem with fuzzy processing time in multiple factories.

Considering some factors, e.g., production interruptions, operator fatigue, and machine wear, the operations take more time if they start processing later, which is called deteriorating effect. Deteriorating scheduling has attracted considerable attention recently in various production environments, taking into account the actual factory conditions. References [33, 34] firstly built a deteriorating scheduling model for the single machine scheduling problem, where the processing time of operations was a linear increasing function of their start time. Fu et al.[35] utilized bi-population evolutionary algorithm to address the stochastic hybrid flow shop problem with deteriorating jobs. In Ref. [36], the deterioration effect of FJSP was considered, which is solved with the modified animal migration optimization algorithm. Actually, fuzzy FJSP and the deterioration effect have some characteristics in common, but few related literature try to connect them.

Recently, many researchers have proposed reinforcement learning (RL) based approaches to address scheduling problems due to its adaptability in different environments and scalability for large-scale problems. To address the dynamic job shop scheduling problem, Wang[37] developed a weighted Q-learning algorithm in combination with clustering and dynamic search. In Ref. [38], the state was represented as multi-channel images, and a deep convolutional neural network was adopted due to its real-time reaction and adaptation in various environment. An improved pointer network was presented by Wang and Pan[39] for the policy learning, in which the processing time of each operation is selected as the state. Wang et al.[40] selected the processing time matrix, the machine assigned matrix, and the processing status of operations matrix as the state and put them into neural network to learn the policy. For addressing the energy-aware distributed hybrid scheduling in flow shop, a cooperative memetic algorithm with an RL-based

policy agent was proposed by J. J. Wang and L. Wang[41]. In Ref. [42], Q-learning is adopted to choose a suitable heuristic strategy among predesigned heuristic methods based on historical information feedback. In Ref. [43], a bi-population cooperative framework based on double Q-learning was designed to further optimize distributed no-wait flow shop scheduling problem. Li et al.[44] combined artificial bee colony algorithm and Q-learning to solve the permutation flow shop scheduling problem with minimizing the makespan. A cooperative scatter search with Q-learning mechanism (QCSS) was presented in Ref. [45], which adopts Q-learning to balance the exploration and exploitation capabilities. RL will also be performed to improve algorithm's performance in this study.

Until now, most scholars have viewed FFJSP and deteriorating scheduling as two separate fields and ignored the connection between them. Actually, fuzzy FJSP and the deterioration effect have some characteristics in common, but few related literature try to connect them. The original intention of both problems is to simulate a more realistic factory environment. FFJSP emphasizes that the processing time can not be represented as a certain number due to many uncertainties in production. If fuzzy numbers are utilized to represent processing time while introducing deterioration effect, a new mathematical model can be established. Meanwhile, considering total energy consumption as one objective can be in line with the concept of green manufacturing. The model is called energy-efficient fuzzy flexible job shop scheduling problem with deteriorating jobs (EFFJSPD), which is able to simulate a more realistic processing environment. The scheduling scheme obtained by solving this model will be more practical.

In this paper, EFFJSPD is proposed, and a new algorithm called reinforcement learning based biased bi-population evolutionary algorithm (RB^{2}EA) is presented to minimize makespan and total energy consumption represented as TFN. The major contributions are summed up in the following. Propose a novel fuzzy FJSP model which is connected with deteriorating effect (in Section 2). RB^{2}EA (in Section 3) is presented to solve the EFFJSPD, which combines Q-learning algorithm with bi-population evolutionary algorithm. Q-learning is utilized to resize the population size according to the quality of population. Four heuristic strategies are designed to initialize the population (in Section 3.1). A local search strategy (in Section 3.3) based on the Q-learning and the interaction mechanism (in Section 3.5) are proposed.

2 Problem Description

2.1 Fuzzy set

In 1965, Zadeh[46] invented fuzzy set theory as a mathematical tool for describing uncertainty and ambiguity in human reasoning. Fuzzy sets are a mathematical representation of ambiguity and uncertainty in a system. In contrast to traditional sets, in which an element is either a member or not a member, a fuzzy set $\tilde{F}$ allows an element x to have a membership degree between 0 and 1. This degree of membership indicates the element's level of similarity to the set, which is defined as the membership function $\mu_{\tilde{F}}(x)$. Fuzzy set can be defined as follows:

If X is a definite set and x is a particular element of X, then a fuzzy set F defined on X can be written as a collection of ordered pairs.

$$\tilde{F} = \{(x, \mu_{\tilde{F}}(x)), x \in X\},\ 0 \leqslant \mu_{\tilde{F}}(x) \leqslant 1 \tag{1}$$

The membership function of a triangle fuzzy number is similar to the triangle, which has three parameters: a, b, and c and can be represented as a triple (a,b,c). The membership function can be formulated as follows:

$$\mu_{\tilde{F}}(x) = \begin{cases} 0, & x \leqslant a; \\ \dfrac{x-a}{b-a}, & a < x \leqslant b; \\ \dfrac{c-x}{c-b}, & b < x < c; \\ 0, & x \geqslant c \end{cases} \tag{2}$$

2.2 Fuzzy operation

The completion of a total scheduling requires three operations on processing time: addition, ranking, and maximum. The addition operator is used to calculate the ending time of an operation. Ranking and maximum operators are performed to determine when to start the process of an operation. Consequently, this section will introduce these three TFN operators. For two TFNs $\tilde{u} = (u_1, u_2, u_3)$ and $\tilde{v} = (v_1, v_2, v_3)$:

(1) Addition operator: $\tilde{u} + \tilde{t} = (u_1 + v_1, u_2 + v_2, u_3 + v_3)$.

(2) Ranking operator:

(a) $f_1(\tilde{x}) = \dfrac{x_1 + 2x_2 + x_3}{4}$, if $f_1(\tilde{u}) > f_1(\tilde{v})$, $\tilde{u} > \tilde{v}$; otherwise, $\tilde{u} < \tilde{v}$;

(b) $f_2(\tilde{x}) = x_2$, when $f_1(\tilde{u}) = f_1(\tilde{v})$, if $f_2(\tilde{u}) > f_2(\tilde{v})$, $\tilde{u} > \tilde{v}$; otherwise, $\tilde{u} < \tilde{v}$;

(c) $f_2(\tilde{x}) = x_3 - x_1$, when $f_2(\tilde{u}) = f_2(\tilde{v})$, if $f_3(\tilde{u}) > f_3(\tilde{v})$, $\tilde{u} > \tilde{v}$; else, $\tilde{u} < \tilde{v}$.

(3) Maximum operator: if $\tilde{u} > \tilde{v}$, then $\tilde{u} \vee \tilde{v} = \tilde{u}$, otherwise, $\tilde{u} \vee \tilde{v} = \tilde{v}$.

2.3 Fuzzy processing time considering deteriorating effect

The deteriorating effect means that the operations take more time when they are processed later. The significance of studying fuzzy scheduling is to account for all scheduling uncertainties, such as processing equipment, environmental or human factors, etc., so that the exact processing time cannot be determined. In other words, the fuzzy scheduling itself takes the influence of the deterioration effect on the makespan into consideration, hence it is conceptually reasonable to treat the deterioration effect as a part of the fuzzy scheduling.

A parameter called deterioration coefficient α is usually set for each stage of each job in the deteriorating scheduling[35]. The processing time of an operation grows linearly as the start time is delayed, and the speed of growth is just determined by α. This paper retains this linearity but transforms this linear increase into an internal linear shift of the TFN.

It is assumed that an operation can be processed on M_1, M_2, and M_3, and requires the same standard processing time. As shown in Fig. 1, the variation of the processing time in the normal deteriorating scheduling is shown in the upper part, and the standard processing time for this operation is a certain number t_0. As the start time is pushed back, the needed processing time for this operation increases. If the operation starts at $t = 2$, the extended processing time due to the deterioration effect is $\Delta t_1 = 1$; and if the operation begins at $t = 5$, the additional processing time is $\Delta t_2 = 2.5$. It is obvious from the figure that Δt_2 is greater than Δt_1.

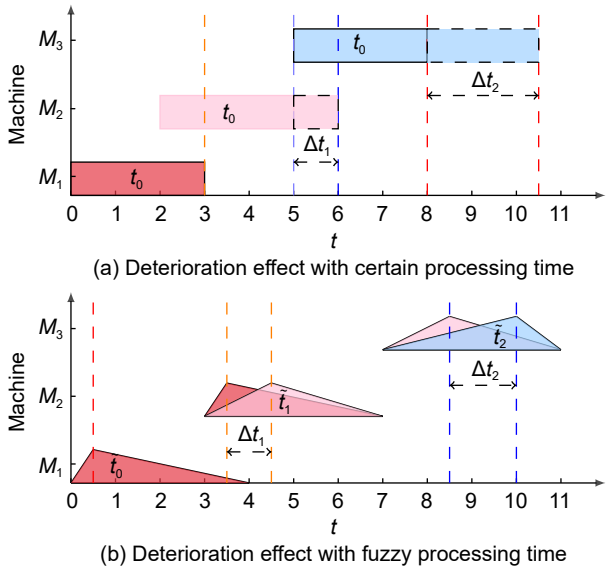

Fig. 1 Deterioration effect with fuzzy processing time.

However, this variation law is no longer applicable when the operation's processing time is a TFN. This is due to the fact that processing time is represented as fuzzy sets in fuzzy job scheduling, and this uncertainty has already been accounted for. In other words, the processing time represented by a TFN fluctuates within a particular range, which includes processing time extension due to deterioration effects. But the deteriorating effect can also be reflected in fuzzy scheduling, as depicted in Fig. 1's lower portion. The standard fuzzy time for an operation is $\tilde{t}_0 = (0, 0.5, 4)$, and the peak time in this TFN is 0.5, indicating that 0.5 is the most probable value for this operation's necessary time. As time goes by, let the operation starts processing at $t = 3$ ($\tilde{t} = (3, 3, 3)$ if expressed in TFN), then the processing time of this operation becomes $\tilde{t}_1 = (0, 1.5, 4)$, indicating that the peak time turns to 1.5 (calculated with $4.5 - 3$). If Δt means the shift of the peak time of TFN processing time, $\Delta t_1 = 1.5 - 0.5 = 1$. If the operation begins at $t = 7$, then the peak time shifts to 3 (calculated by $10 - 7$). $\Delta t_2 = 3 - 1.5 = 1.5$. This suggests that the peak time of the processing time, expressed in TFN, generates a greater linear shift the later the processing of the operation begins. This variation reflects the fact that the presence of deterioration effects increases the probability of the operation requiring more processing time as the start time becomes later and later.

2.4 Mathematical modeling of EFFJSPD

The notations used in this section are described as Table 1.

$$x_{ijk} = \begin{cases} 1, & \text{if process } o_{ij} \text{ on machine } k; \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$$y_{ijpqk} = \begin{cases} 1, & \text{if process } o_{ij} \text{ before } o_{pq} \text{ on machine } k; \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

EFFJSPD can be described as below. $J = \{J_1, J_2, \ldots, J_n\}$ is the job set, and $\Omega = \{M_1, M_2, \ldots, M_n\}$ is the machine set. Each job J_i has a set of h_i operations. Operation o_{ij} can be processed on any machines in a set Ω_{ij}, $\Omega_{ij} \subset \Omega$. The processing time considered

Table 1 Notations table.

Notation	Description
n	Number of jobs
m	Number of machines
o_{ij}	The j-th operation of job J_i
m_{ij}	Number of machines of o_{ij}
h_i	Number of operations of job J_i
$\tilde{p}_{ijk0}$	Standard fuzzy processing time of o_{ij} on machine k
$\tilde{p}_{ijk}$	Deteriorating fuzzy processing time of o_{ij} on machine k
x_{ijk}	Binary variable
y_{ijpqk}	Binary variable
$\widetilde{S}_{ij}$	Fuzzy start time of the j-th operation of job J_i
$\widetilde{C}_{ij}$	Fuzzy completion time of the j-th operation of job J_i
Ω	Total machine set
Ω_{ij}	Set of compatible machines for o_{ij}
$\widetilde{\mathrm{TEC}}$	Total fuzzy energy consumption
$\widetilde{C}_k$	Fuzzy completion time of the last operation on machine k
$\widetilde{C}_{\max 0}$	Standard fuzzy maximum completion time of all jobs
$\widetilde{C}_{\max}$	Deteriorating fuzzy maximum completion time of all jobs
E_k	Energy consumption per unit time in processing mode
SE_k	Energy consumption per unit idle time
L	A large number enough
$\preceq^{\Delta}$	Ranking notation of TFN, if $\max(\widetilde{s},\widetilde{t})=\widetilde{t}$, $\widetilde{s}\preceq^{\Delta}\widetilde{t}$

deteriorating effect of o_{ij} on machine M_k is expressed as $\tilde{p}_{ijk}=(p'_1,p'_2,p'_3)$, while the standard processing time is $\tilde{p}_{ijk0}=(p_1,p_2,p_3)$. The completion time of all jobs is also a TFN, represented as $\widetilde{C}_{\max 0}=(c_1,c_2,c_3)$, and the start time is $\widetilde{S}_{ij}=(s_1,s_2,s_3)$. Moreover, the deteriorating processing time can be obtained by the following equations.

$$p'_1=p_1 \tag{5}$$

$$p'_2=\frac{1}{2}\times\left(\frac{s_1}{c_1}+\frac{s_3}{c_3}\right)\times(p_3-p_2)+p_2 \tag{6}$$

$$p'_3=p_3 \tag{7}$$

The EFFJSPD consists of two subproblems: machine assignment and operation sequencing. Each operation chooses a machine from the candidates. And another one is to generate compromising scheduling for all operations across total machines. The problem has two targets to optimize, makespan $\widetilde{C}_{\max}$ and total energy consumption $\widetilde{\mathrm{TEC}}$. The formulations of the problem model are shown below:

$$\text{minimize } f_1=\widetilde{C}_{\max} \tag{8}$$

$$\text{minimize } f_2=\widetilde{\mathrm{TEC}} \tag{9}$$

$$\text{s.t.,}\quad \tilde{S}_{ij}+x_{ijk}\times\tilde{p}_{ijk}\preceq^{\Delta}\widetilde{C}_{ij},\ i=1,2,\ldots,n,\ j=1,2,\ldots,h_i,\ k=1,2,\ldots,m \tag{10}$$

$$\widetilde{C}_{ij}\preceq^{\Delta}\tilde{S}_{i(j+1)},\ i=1,2,\ldots,n,\ j=1,2,\ldots,h_i-1 \tag{11}$$

$$\widetilde{C}_{ih_i}\preceq^{\Delta}\widetilde{C}_{\max},\ i=1,2,\ldots,n \tag{12}$$

$$\tilde{S}_{ij}+\tilde{p}_{ijk}\preceq^{\Delta}\tilde{S}_{pq}+L(1-y_{ijpqk}),\ i=1,2,\ldots,n,\ p=1,2,\ldots,n,\ j=1,2,\ldots,h_i,\ q=1,2,\ldots,h_p,\ k=1,2,\ldots,m \tag{13}$$

$$\widetilde{C}_{ij}\preceq^{\Delta}\tilde{S}_{i(j+1)}+L(1-y_{pqi(j+1)k}),\ i=1,2,\ldots,n,\ p=1,2,\ldots,n,\ j=1,2,\ldots,h_i-1,\ q=1,2,\ldots,h_p,\ k=1,2,\ldots,m \tag{14}$$

$$\sum_{k=1}^{m_{ij}}x_{ijk}=1,\ i=1,2,\ldots,n,\ j=1,2,\ldots,h_i \tag{15}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{h_i}y_{ijpqk}=x_{pqk},\ k=1,2,\ldots,m,\ p=1,2,\ldots,n,\ q=1,2,\ldots,h_p \tag{16}$$

$$\sum_{p=1}^{n}\sum_{q=1}^{h_p}y_{ijpqk}=x_{ijk},\ k=1,2,\ldots,m,\ i=1,2,\ldots,n,\ j=1,2,\ldots,h_i \tag{17}$$

$$\{0,0,0\}\preceq^{\Delta}\tilde{S}_{ij},\ i=1,2,\ldots,n,\ j=1,2,\ldots,h_i \tag{18}$$

$$\{0,0,0\}\preceq^{\Delta}\widetilde{C}_{ij},\ i=1,2,\ldots,n,\ j=1,2,\ldots,h_i \tag{19}$$

Formulas (10) and (11) restrict the order based on the processing priority of each job. Formula (12) represents a limitation on the job's completion time, i.e., the completion time of each job cannot exceed the completion time of all jobs. Formulas (13) and (14) are utilized to ensure that each machine can process no more than one job at a time. Constrained by Formula (15), each job's operation can only be assigned to one machine. Formulas (16) and (17) guarantee that the operations on one machine are prioritized. Formula (16) selects the front job, and Formula (17) picks the next job. Formulas (18) and (19) restrict that the start and finish time to positive value.

$\widetilde{C}_{\max 0}$ is the completed time of all jobs using the standard processing fuzzy time, whose primary effect is to calculate $\widetilde{C}_{\max}$. $\widetilde{C}_{\max}$ is defined as $\widetilde{C}_1\vee\widetilde{C}_2\vee\cdots\vee\widetilde{C}_m$, which means the latest completion time of all machines

is calculated by maximum operator (confer Section 2.2) of all $\widetilde{C}_k = (C_{k1}, C_{k2}, C_{k3})$. The total energy consumption (TEC) is divided into two parts, the energy consumption $\widetilde{\mathrm{EC}}_1$ when the machine is working and the energy consumption $\widetilde{\mathrm{EC}}_2$ of the idle time. The total fuzzy energy consumption can be computed as follows:

$$\widetilde{\mathrm{TEC}} = \widetilde{\mathrm{EC}}_1 + \widetilde{\mathrm{EC}}_2 = \sum_{k=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{h_i} \tilde{p}_{ijk}x_{ijk} \times E_k + \sum_{k=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{h_i} \mathrm{SE}_k \times (\widetilde{C}_k - \tilde{p}_{ijk}x_{ijk}) = \sum_{k=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{h_i} \tilde{p}_{ijk}x_{ijk} \times (E_k - \mathrm{SE}_k) + \widetilde{C}_k \times \mathrm{SE}_k \quad (20)$$

2.5 Illustrative example

To illustrate the problem, Fig. 2 presents an example Gantt chart of an EFFJSPD problem. The two Gantt charts represent one complete scheduling process, and $C_{\max 0}$ is necessary to calculate the deteriorating operation time, so $C_{\max 0}$ should be calculated by a fuzzy scheduling using the standard processing time. For example, when determining the start time of $O_{1,3}$, the time of the last process of Machine 3 selected by $O_{1,3}$ is $(4,6,9)$, while the time of the last process of $O_{1,3}$ is $(4,6,10)$. It can be determined through ranking and maximum operator (confer Section 2.2). The earliest processing time for $O_{1,3}$ should be $(4,6,10)$. It can be found from the first Gantt chart that the $C_{\max 0}$ is (7, 11, 17).

The bottom half of the Fig. 2 shows the core part of the EFFJSPD, which considers the effect of the deterioration effect in fuzzy scheduling. For $O_{3,2}$, its completion time changes from $(4,6,9)$ to $(4,6.4,9)$. The start time of the job $(3,4,6)$ does not change, but by Eq. (6), the new processing time of $O_{3,2}$ changes to $\left(1,2+\frac{1}{2}\times\left(\frac{3}{7}+\frac{6}{17}\right)\times(3-2),3\right)$, that is, $(1,2.4,3)$. Thus, the ending time of $O_{3,2}$ is $(3,4,6)+(1,2.4,3)=(4,6.4,9)$. For all jobs whose start time is not $(0,0,0)$, the processing time will be impacted. By comparison, it

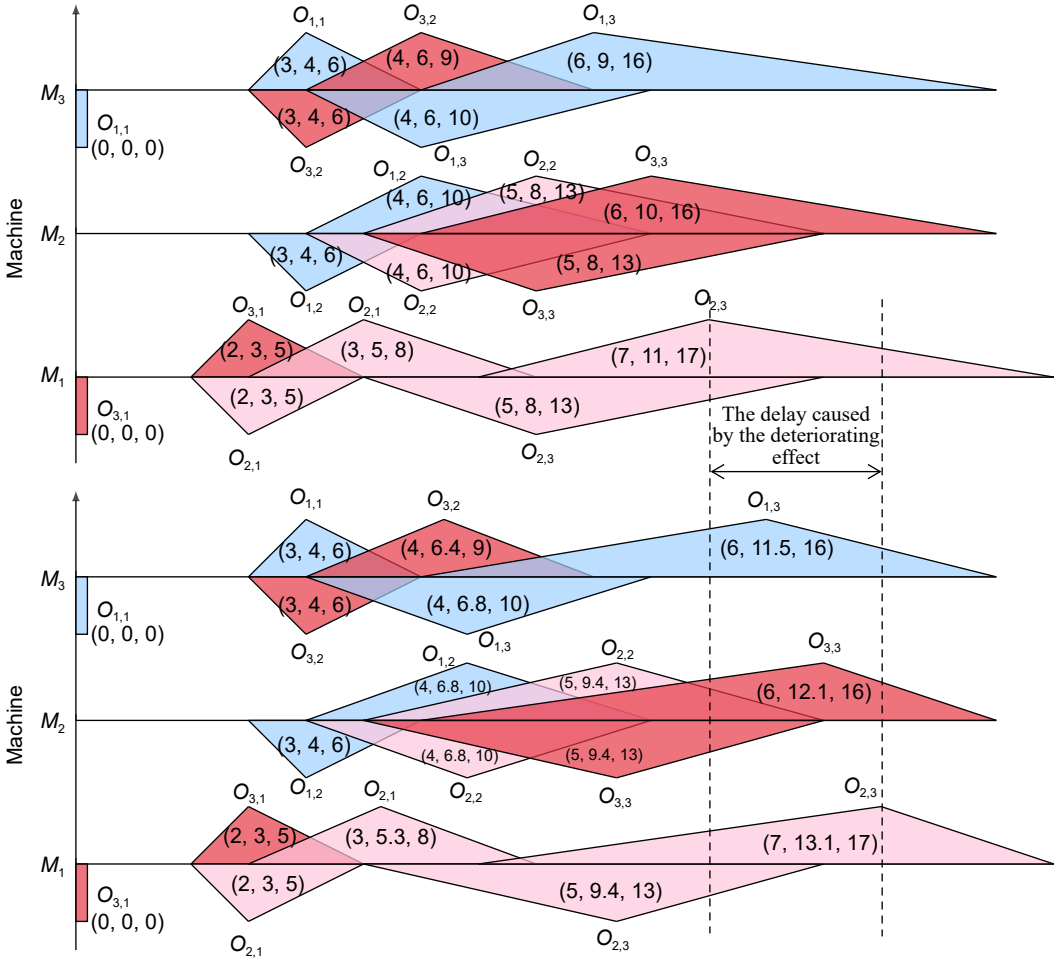

Fig. 2 Example Gantt chart of EFFJSPD problem.

can be intuitively found that the center of gravity of the Gantt chart after taking into account the deterioration effect is clearly shifted to the right, and the two dashed lines represent the shift of makespan after taking into account the deterioration effect. It is important to note that for a complete EFFJSPD scheduling, the shapes of two Gantt charts are not necessarily identical. The reason for this is that the update for deteriorating processing time may lead to a shift in the start time of some operations.

3 RB²EA for EFFJSPD

Bi-population or multipopulation is a common framework in metaheuristic algorithms, including multi-objective evolutionry algorithms (such as the traditional MOEA/D[47]), differential evolution (DE)[48], and PSO[49]. Nevertheless, these multi-population algorithms typically initialize two identical or similar populations, most frequently by initializing a huge population and then dividing it into many subpopulations. This is analogous to having the search algorithm's beginning point at the same location, and the likelihood of not searching for the global optimal solution stays high despite the diverse search paths that follow. In this paper, a biased bi-population is proposed for the dual objective problem, where each population has a certain preference for one objective problem, while the overall framework is a multi-objective genetic algorithm. This enables the bi-population to search for the best solution along diverse paths from different beginning points in the search space, thereby broadening the search range and preventing it from getting trapped in a local optimal solution. Figure 3 shows this idea. Two dashed circles represent initial populations with preferences for makespan and TEC,

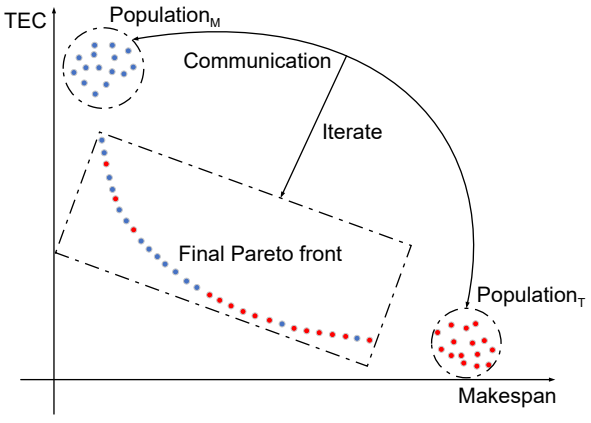

Fig. 3 Schematic diagram of RB²EA.

named population$_M$ and population$_T$, respectively. During the iteration, the two populations continuously carry out population communication strategy, and finally obtain a non-dominated solution set. Furthermore, inspired by the bi-population evolutionary algorithm with feedback (FBEA)[22], a reinforcement learning based population size adjustment mechanism is introduced to enable the dual population to dynamically adjust the population size in real time based on the population's quality.

3.1 Initialization

To obtain a higher quality initial population, a combination of the four initialization techniques will be employed. Since RB²EA is a bi-population algorithm with preference, such a preference should also be reflected in the initialization algorithm. Two of these methods are based on the fitness value, so they are named fitness selection (FS).

FS1 is detailed below. Firstly generate population randomly, which contains N_p individuals, and determine the fitness of each individual. Choose the best 5% of the makespan and the TEC separately as the offspring.

FS2 and FS1 are comparable. FS2 selects the worst 5% of the makespan and the TEC separately and reverses the operation sequence as the offspring, which distinguishes it from FS1.

The other two heuristics are derived from FJSP, which are global selection (GS) and local selection (LS) strategies. GS and LS mainly consider the load problem of machine selection, so that the workload of each selected machine is balanced as much as possible, and the utilization rate of the machine is fully improved.

GS: Create an array of the same size as the size of machine set to store the working time of each machine. Choose at random a job from the job set to complete all of its operations sequentially and update the array. For each operation, the processing machine with the shortest operating time in the array is chosen.

LS: The operation processes of LS and GS are nearly identical, with the exception that LS selects jobs sequentially, beginning with the first and ending with the last.

Aiming to have the bi-population with an objective value preference, a population allocation procedure (allocating process) was created. The populations formed by FS1 and FS2 based on makespan values are assigned to population$_M$, while those based on TEC

values are allocated to $\text{population}_\text{T}$. The populations generated using other initialization methods were equally assigned to $\text{population}_\text{M}$ and $\text{population}_\text{T}$. The detailed steps for population initialization with totaly N_p solutions are described in Algorithm 1.

3.2 Q-learning for population size adjustment

In this section, the population size adjusting mechanism based on the Q-learning algorithm will be described in detail. Watkins and Dayan[50] firstly proposed Q-learning in 1992, which is a model-free off-policy reinforcement learning algorithm. The algorithm learns the optimal action-value function, which reflects the highest expected reward for choosing a certain action in a given state, and then utilizes this policy to determine the best action to take in each state.

3.2.1 State definition

Defining the state as quantity that reflects the current bi-population quality allows higher quality population to attain larger population sizes and increases the likelihood to seek the global optimal solution. Two metrics will be introduced to define the state.

δ indicates the percentage of non-dominated solutions of $\text{population}_\text{M}$. A fast non-dominated sort is performed on the current total population, and the number of solutions in the non-dominated set obtained is N_ND, where there are n_M individuals belonging to $\text{population}_\text{M}$.

Algorithm 1 Initialize population

input : N_p

output: $\text{population}_\text{M}$ and $\text{population}_\text{T}$

1 Random generate a population, size N_p;
2 Utilize FS1 to generate offspring P_1, size $(0.1N_p)$;
3 Employ FS2 to generate offspring P_2, size $(0.1N_p)$;
4 Perform GS to generate offspring P_3, size $(0.2N_p)$;
5 Execute LS to generate offspring P_4, size $(0.2N_p)$;
6 Randomly generate offspring P_5, size $(0.4N_p)$;
7 Perform allocating process to obtain $\text{population}_\text{M}$ and $\text{population}_\text{T}$;

$$\delta = \frac{n_\text{M}}{N_\text{ND}} \tag{21}$$

η indicates relative diversity of $\text{population}_\text{M}$. dv_M is the diversity metric of $\text{population}_\text{M}$, and dv_T is that of $\text{population}_\text{T}$. Metric dv (diversity) is proposed in Ref. [51], but there are some differences. dv can be calculated by Eq. (23),

$$\eta = \frac{\text{dv}_\text{M}}{\text{dv}_\text{M} + \text{dv}_\text{T}} \tag{22}$$

$$\text{dv} = \frac{\sum_{i=1}^{N-1} |d_i - \bar{d}|}{(N_\text{ND} - 1)\bar{d}} \tag{23}$$

where d_i is the Euclid distance between two adjacent Pareto front (PF) points. $\bar{d}$ represents the mean value of d_i. N_ND is the number of solutions in the set of non-dominated. The greater the dv, the better the diversity. To sum up, the metrics δ and η of the bi-population are chosen as the state of the Q-learning state.

3.2.2 Action definition

The first action is to adjust the size of the two populations, and the second action is to modify the parameter freq. The resizing process is the size of one population plus 5 and the other one minus 5. It is worth mentioning that the minimum size of each population is limited to $0.2 \times N_p$. Another process is increasing or decreasing freq, which is an important parameter in Section 3.3. Table 2 gives a Q-table example in 15×8×2 instance after 200 iterations, where N_M and N_T are the size of $\text{population}_\text{M}$ and $\text{population}_\text{T}$, respectively.

3.2.3 Reward definition

If the mean value of the total population's fitness dominates that of previous iteration, let $\text{dmt} = 1$, otherwise, let $\text{dmt} = 0$. Based on the illustration above, the strategy utilizing Q-learning method to adjust population size, namely Q-APS, is proposed. The whole steps of Q-APS are stated in Algorithm 2.

$$\Delta\text{dv} = \text{dv}_i - \text{dv}_{i-1} \tag{24}$$

Table 2 Q-table after 200 iterations.

Condition	Reward			
	$N_\text{M} = N_\text{M} + 5$ $N_\text{T} = N_\text{T} - 5$ $\text{freq} = \text{freq} - 1$	$N_\text{M} = N_\text{M} - 5$ $N_\text{T} = N_\text{T} + 5$ $\text{freq} = \text{freq} - 1$	$N_\text{M} = N_\text{M} + 5$ $N_\text{T} = N_\text{T} - 5$ $\text{freq} = \text{freq} + 1$	$N_\text{M} = N_\text{M} - 5$ $N_\text{T} = N_\text{T} + 5$ $\text{freq} = \text{freq} + 1$
$\delta < 0.5$, $\eta < 0.5$	0	2.2138	0	0.6
$\delta \geqslant 0.5$, $\eta < 0.5$	14.1579	0	3.5754	1.8354
$\delta < 0.5$, $\eta \geqslant 0.5$	0	2.8156	0.8556	0
$\delta \geqslant 0.5$, $\eta \geqslant 0.5$	15.4823	1.5651	16.0873	1.5998

Algorithm 2 Q-APS

input : greedy factor ε, learning rate α, discount factor γ, and population
output: Q_table
Q_table $\leftarrow 0, S_t \leftarrow 1$;
Calculate zmean_i and dv_i of the total population;
while the stopping criterion is not satisfied **do**
  if rand < ε **then**
    Select the max $Q(\text{state}, A_i)$ action A_t, i= 1, 2, 3, 4
  else
    Randomly select an action A_t
  end
  Execute the action A_t for new population size and freq;
  Perform evolutionary algorithm (EA) to evolve population and get the PF;
  $\text{zmean}_{i-1} = \text{zmean}_i$, $\text{dv}_{i-1} = \text{dv}_i$, then get the reward R;
  Caculate the new state S_{t+1};
  $Q(S_t, A_t) = Q(S_t, A_t)+\alpha[R+\gamma\max(Q(S_{t+1}, A_t)-Q(S_t, A_t)]$
end

$$\text{Reward} = \begin{cases} 10, & \Delta\text{dv} \geqslant 0,\ \text{dmt} = 1; \\ 6, & \Delta\text{dv} < 0,\ \text{dmt} = 1; \\ 4, & \Delta\text{dv} \geqslant 0,\ \text{dmt} = 0; \\ 0, & \Delta\text{dv} < 0,\ \text{dmt} = 0 \end{cases} \tag{25}$$

3.3 Local enhancement based on Q-learning

As stated in Section 3.2, the Q-APS will modify the parameter freq, which determines the frequency of the local enhancement. The operation sequence (OS) has two techniques for local searching.

LS1: Swap the position of last operation with random another operation.

LS2: Insert the position of last operation in front of random another operation.

In order to maintain the preference of the two populations, $\text{population}_\text{M}$ and $\text{population}_\text{T}$ take different local search methods, respectively, for machine assignment (MS).

For $\text{population}_\text{M}$:

LS3: Find the last finished operation o_i and move o_i to another machine whose processing time is the shortest.

LS4: Choose one operation $o_{i,j}$ randomly and move it to another machine whose processing time is the shortest.

For $\text{population}_\text{T}$:

LS3: Find the last finished operation o_i and move o_i to another machine with minimum energy consumption (simple processing is the processing time multiplied by the energy consumption factor per unit time of the machine in working condition).

LS4: Randomly select an operation $o_{i,j}$ and move it to another machine with minimum operation time.

The neighborhood selection mechanism explains how to select optimal solution from neighborhood solutions. For each non-dominated solution x_i, compare it with each neighborhood solution in turn. If the dominance relationship exists, the non-dominated solution becomes x_i. Otherwise, the solution with the shortest makespan is selected in $\text{population}_\text{M}$, and the solution with less TEC is x_i in $\text{population}_\text{T}$. N represents the size of non-dominated solution set and the neighborhood size $T = 10$. The biased local search adopting Q-learning algorithm (Q-BLS) is stated as Algorithm 3.

3.4 Genetic operation in RB^{2}EA

The genetic operations at the heart of genetic algorithms allow populations to evolve more efficiently and search for the global optimal solution more easily. The algorithm is shown in Algorithm 4, in which genetic operations are carried out on two populations, $\text{population}_\text{M}$ and $\text{population}_\text{T}$. Perform crossover and mutation on each individual in each population, merge the old and new individuals into a new population, and then choose the best individuals from the new population using tournament selection.

Unlike previous tournament selection operations, the novel binary tournament means that different selection

Algorithm 3 Q-BLS

input : N, T, freq, and gen
output: enhanced solution
if mod(gen, freq)=0 **then**
  for $i \leqslant N$ **do**
    for $j \leqslant T$ **do**
      if $\text{rand}_1 < 0.5$ **then**
        Perform LS1 for x_i and generate y_j;
      else
        Execute LS2 for x_i and generate y_j;
      end
      if $\text{rand}_2 < 0.5$ **then**
        Perform LS3 for y_j;
      else
        Execute LS4 for y_j;
      end
    end
  end
  Perform neighborhood selection mechanism;
end

Algorithm 4 Genetic operation

input: $\text{population}_\text{M}$ and $\text{population}_\text{T}$
output: $\text{population}_\text{M}$ and $\text{population}_\text{T}$
for $i \leqslant N_\text{M}$ **do**
  Perform crossover and mutation for x_i and x_{i+1} in $\text{population}_\text{M}$;
  Get new y_i and y_{i+1};
  $i = i + 2$;
end
Combining x and y into a new population;
Execute novel binary tournament for $\text{population}_\text{M}$;
for $i \leqslant N_\text{T}$ **do**
  Perform crossover and mutation for x_i and x_{i+1} in $\text{population}_\text{T}$;
  Get new y_i and y_{i+1};
  $i = i + 2$;
end
Combining x and y into a new population;
Execute novel binary tournament for $\text{population}_\text{T}$;

mechanisms were applied to the two different populations. For $\text{population}_\text{M}$, two individuals are selected at a time from the merged population, and if a dominance relationship exists between them, the non-dominated solution is chosen. If not, the solution with the shortest makespan is selected. This procedure is repeated until the total number of selected individuals equals to the original $\text{population}_\text{M}$ population size. With $\text{population}_\text{T}$, the selection mechanism is nearly identical, with the exception that smaller TEC individual will be selected in the absence of a dominance relationship.

Precedence operation crossover (POX)[52] is selected as the crossover operation of OS, and a uniform crossover (UX) technique is performed on the MS. Three mutation methods are used for OS, and two mutation techniques are designed for MS.

OS_1: Swap the order of two operations in OS randomly.

OS_2: Choose two operations randomly and put the latter in front of the location of the other one in OS string.

OS_3: Reverse the operation sequence between random two locations in OS string.

MS_1: Randomly select one tenth of all operations and assign them to the machine with the shortest processing time.

MS_2: Randomly select one tenth of all operations and assign them to the machine which consumes minimum energy.

It is worth mentioning that in RB^{2}EA the mutation rate and the crossover rate are adaptively modified by two parameters δ and η, which are used to determine the state. The process of mutation and crossover are shown in Algorithm 5 in detail.

3.5 Interaction mechanism

The preference of the two populations is always presented in RB^{2}EA, ensuring that the search routes of the two populations in the search space are distinct. However, this may lead to a weak convergence of the algorithm as a whole. To tackle this issue and maximize the benefits of the bi-population algorithm, a communication mechanism between the two populations is designed.

The interaction mechanism can be understood simply as the exchange of individuals with features in two populations. The feature is reflected in RB^{2}EA as preferences for two objective values, so the interaction mechanism is essentially the replication of individuals with one high objective value from one population to another. This exchange cannot occur too regularly, otherwise each subpopulation will be unable to converge as it continues to receive more individuals. Consequently, the algorithm is allowed to run a fixed number of times before allowing the two populations to perform the interaction mechanism at the optimal time, which is determined by the parameter β. The detailed operation of this mechanism is shown in Algorithm 6.

3.6 Framework of RB^{2}EA

The whole framework of RB^{2}EA is displayed in Algorithm 7, in which the stopping criterion is the predefined maximum iterations. It is important to emphasize that the Q-BLS and iteraction mechanism will only be performed when meeting their own condition (confer Sections 3.3 and 3.5). Firstly, the

Algorithm 5 Mutation and crossover strategy

input : δ, η, x, and y
output: x' and y'
if $x, y \in \text{population}_\text{M}$ **then**
  $\text{rate} = 0.5(1-\delta) + 0.5\eta$
else
  $\text{rate} = 0.5(1-\eta) + 0.5\delta$
end
if rand < rate **then**
  POX and UX methods are performed;
  Pick an OS mutation strategy for x and y separately with roulette algorithm;
  Select an MS mutation method for x and y separately with roulette algorithm;
end

Algorithm 6 Interaction mechanism

input : β, g, MAXGEN, $\text{population}_\text{M}$, and $\text{population}_\text{T}$
output: $\text{population}_\text{M}$ and $\text{population}_\text{T}$
if $g = \beta \times \text{MAXGEN}$ **then**
  find x_1 with the worst TEC in $\text{population}_\text{M}$;
  find x_2 with the best TEC in $\text{population}_\text{T}$;
  $x_1 \leftarrow x_2$;
  find x_3 with the worst makespan in $\text{population}_\text{T}$;
  find x_4 with the best makespan in $\text{population}_\text{M}$;
  $x_3 \leftarrow x_4$;
end

Algorithm 7 RB²EA

input: population size N, greedy factor ε, learning rate α, and discount factor γ
output: non-dominated solutions
1 Initialize the populations $\text{population}_\text{M}$ and $\text{population}_\text{T}$ sizing $\frac{N}{2}$;
2 Initialize all variables and parameters for algorithm;
3 **while** the stopping criterion is not satisfied **do**
4 Apply Q-learning to resize the bi-population and modify freq;
5 Execute genetic operators in $\text{population}_\text{M}$ and $\text{population}_\text{T}$ separately and obtain new populations $\text{population}'_\text{M}$ and $\text{population}'_\text{T}$;
6 Update populations according to the new size with the novel tournament method;
7 Perform Q-BLS algorithm;
8 Employ interaction mechanism;
9 Update population state and Q-table;
10 **end**
11 Calculate the non-dominated solution set;

heuristic methods are adopted to obtain the initial $\text{population}_\text{M}$ and $\text{population}_\text{T}$. Then some parameters need to be configured, such as the initial state and Q-learning parameters. Next, apply Q-learning to choose the action and resize the bi-population and change the freq. Moreover, execute genetic operators for $\text{population}_\text{M}$ and $\text{population}_\text{T}$ to get the new $\text{population}'_\text{M}$ and $\text{population}'_\text{T}$, respectively. Mix the old and new populations and perform the novel binary tournament algorithm to get the offspring $\text{population}_\text{M}$ and $\text{population}_\text{T}$. Furthermore, if the condition is met, the Q-BLS and the interaction mechanism for the population will be adopted. Finally, it is necessary to update the Q-table. If the terminating condition is not met, back to Line 4 and continue iterating.

4 Numerical Result and Comparison

4.1 Experiment setting

EFFJSPD is actually an updated FFJSP, therefore we can utilize fuzzy FJSP instances or generate instances based on the standard dataset. However, these instances lack energy consumption parameters, therefore they can be extended by adding energy consumption information E and SE as proposed in Ref. [22]. Particularly for typical instances, the determined certain processing time needs to be fuzzified. In the end, we selected 22 instances from Brandimarte[53], Saidi-Mehrabad and Fattahi[54], and Behnke and Geiger[55] at various scales. The instance format is $j \times m \times n$, where j represents the size of job set, m means the size of machine set, and n is the average number of machines per operation. The smallest instance size is $3 \times 5 \times 2$, while the highest is $20 \times 20 \times 6$. The above instances are utilized to evaluate the efficacy of the proposed RB²EA for solving EFFJSPD.

All algorithms are performed using MATLAB2021b and executed on a computer with a 12th Generation Intel Core i7-12700K, 3.61 GHz, 32 GB RAM, and Windows 10 operating system. To compare fairly, the algorithm testing termination criterion is set to the maximum number of iterations, $\text{MAXGEN} = 0.1 \times j \times m \times n$. MAXGEN is 10 specifically for small-scale instances. The population size is also a crucial instance scalability parameter, thus we set $\text{size} = \lceil 8 \times \log_{10}(j \times m \times n) \rceil \times 10$.

4.2 Parameter setting

RB²EA includes four key parameters: (1) the ratio used to choose when to interact with another population β; (2) the learning rate α; (3) the discount factor γ; and (4) the greedy factor ε. The Taguchi method[56] of design-of-experiments is employed to explore the effect of parameter settings on the performance of RB²EA and to determine the best combination for these parameters. Four levels are set for each parameter, as shown in Table 3.

With this method, only 16 combinations instead of 256 need to be tested, which are included in an orthogonal array. To guarantee fairness, each instance generates a test instance to perform the experiment. The RB²EA with each combination (α, β, γ, ε) on each

Table 3 Parameters in different levels.

Level	Parameter			
	β	α	γ	ε
1	0.70	0.10	0.80	0.70
2	0.75	0.15	0.85	0.75
3	0.80	0.20	0.90	0.80
4	0.85	0.25	0.95	0.85

test instance runs 10 times independently and records the hypervolume (HV)[57] value every time. HV is calculated by Eq. (26).

$$\mathrm{HV}(P_{\mathrm{ref}}, r) = \bigcup_{x \in P_{\mathrm{ref}}}^{P_{\mathrm{ref}}} v(x, r) \tag{26}$$

where P_{ref} is the reference point corresponding to the real Pareto front.

For calculating HV value, the x must be normalized by establishing a reference point r. It is important to note that all instances of x should be normalized using the same reference point, which is the worst object value multiplied by 2 in the original population. The algorithm performs better in diversity and convergence with a bigger HV value. In Table 4, the HV values and orthogonal array are listed. In addition, the HVs and significance rank of each parameter are presented in Table 5, where Delta represents the biggest gap between its average HVs at different levels. Moreover, a small Delta value suggests that the performance sensitivity of the parameter is low. It is evident from the data that the discount factor γ is the most sensitive to performance. ε and α are ranked the second and third, respectively. The ratio used to choose when to interact with another population β is the last. According to the mean HV values, the ideal combination for parameter selection should be β=0.75, α=0.2, γ=0.85, and ε=0.85.

Table 4 Orthogonal array and HV values.

Experiment No.	Factor level				HV
	β	α	γ	ε	
1	1	1	1	1	0.631 205 5
2	1	2	2	2	0.628 482 8
3	1	3	3	3	0.627 707 3
4	1	4	4	4	0.627 290 5
5	2	1	2	3	0.628 778 8
6	2	2	1	4	0.631 223 9
7	2	3	4	1	0.626 560 6
8	2	4	3	2	0.628 589 2
9	3	1	3	4	0.627 454 2
10	3	2	4	3	0.628 272 6
11	3	3	1	2	0.628 114 0
12	3	4	2	1	0.629 891 8
13	4	1	4	2	0.626 158 7
14	4	2	3	1	0.626 647 8
15	4	3	2	4	**0.632 412 5**
16	4	4	1	3	0.627 570 7

Table 5 Influence and rank of each parameter.

Level	β	α	γ	ε
1	0.628 671 5	0.628 399 3	0.629 528 5	0.628 576 4
2	**0.628 788 1**	0.628 656 8	**0.629 891 5**	0.627 836 2
3	0.628 433 1	**0.628 698 6**	0.627 599 6	0.628 082 4
4	0.628 197 4	0.628 335 5	0.627 070 6	**0.629 595 3**
Delta	0.000 590 7	0.000 363 1	**0.002 820 9**	0.001 759 1
Rank	4	3	1	2

4.3 Effect of algorithm designs

Next, the efficacy of the RB^{2}EA's special designs in resolving EFFJSPD is evaluated, including heuristic initialization, population adjusting dynamically method with reinforcement learning, local intensification, and bi-population communication mechanism. RB^{2}EA$_1$, RB^{2}EA$_2$, RB^{2}EA$_3$, and RB^{2}EA$_4$ represent four algorithms that lack a method each. RB^{2}EA$_1$ initializes the population at random, RB^{2}EA$_2$ lacks population adjusting dynamically with reinforcement learning, RB^{2}EA$_3$ does not process the local search technique for enhancement, and RB^{2}EA$_4$ has no interaction between the two populations. Each of these four algorithms is compared against the whole RB^{2}EA to determine the effect of each technique on the overall performance. The C metric is adopted to compare the non-dominated object set obtained by different algorithms to show the performance of the algorithm.

$$C(A, B) = \frac{|\{g \in B : \exists h \in A, h \succ g\}|}{|B|} \tag{27}$$

where A and B are non-dominated solution sets, and g is a solution in the non-dominated solution set.

Clearly, the greater $C(A, B)$ value indicates a superior Pareto front Algorithm 1 obtains. To eliminate randomness, 10 test fuzzy instances are created from each instance and executed 10 times in each test instance, with the final C metric result presented in Table 6. To determine whether the difference between the RB^{2}EA and other algorithms is statistically significant, Table 6 also includes the p values of pairwise comparisons with a 95% confidence level. It is evident that C_1 is greater than C_2 in all instances and that p is generally always less than 0.05. From these results, it can be concluded that the four techniques help RB^{2}EA in determining the optimal Pareto front.

4.4 Comparison with other algorithms

Although EFFJSPD is a novel problem that has received little attention, the algorithms developed for EFFJSP can be generalized to solve it as well. In order to demonstrate the RB^{2}EA's superiority in solving

Table 6 Average of RB²EA and four variants on metric C.

(j,m,n)	RB²EA vs. RB²EA$_1$			RB²EA vs. RB²EA$_2$			RB²EA vs. RB²EA$_3$			RB²EA vs. RB²EA$_4$		
	C_1	C_2	p	C_1	C_2	p	C_1	C_2	p	C_1	C_2	p
(3, 5, 2)	**0.94**	0.64	0.00	**0.89**	0.56	0.02	**0.89**	0.59	0.00	**0.87**	0.59	0.00
(4, 5, 1.7)	**0.92**	0.68	0.00	**0.93**	0.62	0.00	**0.94**	0.63	0.00	**0.95**	0.57	0.00
(5, 6, 2.2)	**0.68**	0.20	0.00	**0.74**	0.19	0.00	**0.74**	0.16	0.00	**0.69**	0.19	0.00
(8, 7, 2.6)	**0.71**	0.14	0.00	**0.65**	0.20	0.00	**0.67**	0.21	0.02	**0.64**	0.23	0.00
(9, 8, 2.4)	**0.70**	0.20	0.00	**0.75**	0.12	0.00	**0.70**	0.18	0.00	**0.75**	0.15	0.00
(10, 6, 2)	**0.72**	0.11	0.00	**0.63**	0.18	0.00	**0.59**	0.08	0.00	**0.70**	0.17	0.00
(10, 6, 3.5)	**0.65**	0.24	0.00	**0.63**	0.26	0.02	**0.63**	0.25	0.02	**0.65**	0.26	0.02
(15, 8, 2)	**0.83**	0.05	0.00	**0.72**	0.11	0.00	**0.67**	0.02	0.00	**0.81**	0.09	0.00
(15, 8, 3)	**0.71**	0.11	0.00	**0.72**	0.13	0.00	**0.64**	0.10	0.00	**0.72**	0.14	0.00
(15, 4, 1.5)	**0.61**	0.17	0.00	**0.68**	0.16	0.00	**0.68**	0.14	0.00	**0.66**	0.17	0.00
(10, 10, 3)	**0.79**	0.09	0.00	**0.74**	0.13	0.00	**0.75**	0.12	0.00	**0.75**	0.11	0.00
(20, 5, 3)	**0.75**	0.16	0.00	**0.76**	0.15	0.00	**0.75**	0.16	0.00	**0.72**	0.21	0.00
(20, 10, 1.5)	**0.79**	0.12	0.00	**0.71**	0.13	0.00	**0.81**	0.11	0.00	**0.82**	0.12	0.00
(20, 10, 3)	**0.78**	0.18	0.00	**0.60**	0.23	0.00	**0.67**	0.21	0.00	**0.79**	0.12	0.00
(20, 15, 3)	**0.50**	0.28	0.02	**0.54**	0.18	0.00	**0.57**	0.26	0.02	**0.58**	0.29	0.00
(30, 5, 1.5)	**0.77**	0.12	0.00	**0.74**	0.15	0.00	**0.74**	0.16	0.00	**0.78**	0.12	0.00
(30, 10, 1.5)	**0.66**	0.19	0.00	**0.62**	0.19	0.00	**0.71**	0.16	0.00	**0.75**	0.15	0.00
(30, 10, 3.4)	**0.73**	0.19	0.00	**0.77**	0.16	0.00	**0.71**	0.17	0.00	**0.78**	0.12	0.00
(30, 15, 1.6)	**0.60**	0.28	0.00	**0.63**	0.24	0.00	**0.65**	0.20	0.00	**0.68**	0.20	0.00
(30, 15, 3)	**0.80**	0.13	0.00	**0.69**	0.13	0.00	**0.64**	0.13	0.00	**0.76**	0.14	0.00
(10, 20, 6)	**0.63**	0.19	0.00	**0.67**	0.21	0.00	**0.53**	0.24	0.02	**0.69**	0.20	0.00
(20, 20, 6)	**0.68**	0.22	0.02	**0.68**	0.18	0.00	**0.46**	0.18	0.00	**0.72**	0.19	0.00

EFFJSPD, four algorithms have been chosen for comparison with it, including non-dominated sorting genetic algorithm II (NSGA-II)[51], MOEA/D[47], FBEA[22], and MOEA/D based on reinforcement learning (RMOEA/D)[23]. All of these algorithms utilize the same encoding and decoding methods and stopping criterion to assure fairness. In addition, each initial instance generates 10 fuzzy instances, and each fuzzy instance runs 10 times. Apart from C metric, inverted generational distance (IGD)[58] is selected to compare the performance of these algorithms, which can be calculated by Eq. (28).

$$\mathrm{IGD}(P,P^*)=\frac{\sum_{x\in P^*}\min_{y\in P}\mathrm{dis}(x,y)}{|P^*|} \tag{28}$$

where P is the object value set obtained by the algorithm, and P^* represents the ideal Pareto front, which does not exist in EFFJSPD. dis(x, y) denotes the Euclidean distance between solution x and solution y. Thus, $(0,1),(0,0.5),(0,0),(0.5,0)$, and $(0,1)$ are set as the approximate PF. The average C metric and IGD values of instances in different scales are listed in Tables 7 and 8.

According to Table 7, C_1 is greater than C_2 in all instances, and p is always less than 0.05, which means C_1 and C_2 are significantly different at the 95% confidence level.

In Table 8, FBEA and RMOEA/D are generally less than NSGA-II and RMOEA/D. That is because NSGA-II and MOEA/D are algorithms proposed for multi-objective optimization theory but not practical problems. Although the same encoding and decoding strategies are adopted, they also lack heuristic initialization methods, problem-specific genetic operators, local enhancement strategies, etc. So the performance of NSGA-II and MOEA/D is worse than FBEA and RMOEA/D, which are designed to solve practical scheduling problem.

FBEA adjusts the size of the two populations based on a quality feedback mechanism, while RB²EA utillizes Q-learning algorithm instead. The two populations in FBEA take into account all the optimization objectives simultaneously, while the two populations in RB²EA have special preference for certain optimization objectives, ensuring the diversity

Table 7 Average of RB²EA and four other algorithms on metric C.

(j,m,n)	RB²EA vs. NSGA-II			RB²EA vs. MOEA/D			RB²EA vs. FBEA			RB²EA vs. MOEA/D		
	C_1	C_2	p	C_1	C_2	p	C_1	C_2	p	C_1	C_2	p
(3, 5, 2)	**0.94**	0.50	0.00	**0.78**	0.16	0.02	**0.91**	0.66	0.00	**0.78**	0.23	0.00
(4, 5, 1.7)	**0.99**	0.43	0.00	**0.82**	0.45	0.02	**0.96**	0.73	0.00	**0.81**	0.47	0.02
(5, 6, 2.2)	**0.82**	0.06	0.00	**0.90**	0.04	0.00	**0.72**	0.19	0.00	**0.84**	0.06	0.00
(8, 7, 2.6)	**0.98**	0.00	0.00	**0.95**	0.01	0.00	**0.68**	0.16	0.00	**0.91**	0.01	0.00
(9, 8, 2.4)	**1.00**	0.00	0.00	**0.61**	0.18	0.00	**0.56**	0.22	0.00	**0.62**	0.17	0.00
(10, 6, 2)	**1.00**	0.00	0.00	**0.61**	0.06	0.00	**0.89**	0.00	0.00	**0.74**	0.03	0.00
(10, 6, 3.5)	**1.00**	0.00	0.00	**0.88**	0.01	0.00	**0.78**	0.08	0.00	**0.86**	0.02	0.00
(15, 8, 2)	**1.00**	0.00	0.00	**0.65**	0.06	0.02	**0.86**	0.02	0.00	**0.80**	0.06	0.02
(15, 8, 3)	**1.00**	0.00	0.00	**0.56**	0.03	0.02	**0.83**	0.05	0.00	**0.67**	0.01	0.00
(15, 4, 1.5)	**0.98**	0.01	0.00	**0.92**	0.00	0.00	**0.74**	0.11	0.00	**0.90**	0.00	0.00
(10, 10, 3)	**1.00**	0.00	0.00	**0.72**	0.00	0.00	**0.92**	0.03	0.00	**0.78**	0.00	0.00
(20, 5, 3)	**1.00**	0.00	0.00	**0.73**	0.07	0.00	**0.83**	0.07	0.00	**0.55**	0.16	0.02
(20, 10, 1.5)	**1.00**	0.00	0.00	**0.73**	0.02	0.00	**0.76**	0.08	0.00	**0.72**	0.06	0.00
(20, 10, 3)	**1.00**	0.00	0.00	**0.42**	0.03	0.00	**0.78**	0.07	0.00	**0.50**	0.04	0.00
(20, 15, 3)	**1.00**	0.00	0.00	**0.72**	0.07	0.00	**0.78**	0.08	0.00	**0.76**	0.04	0.00
(30, 5, 1.5)	**1.00**	0.00	0.00	**0.81**	0.05	0.00	**0.73**	0.12	0.00	**0.73**	0.10	0.02
(30, 10, 1.5)	**1.00**	0.00	0.00	**0.54**	0.05	0.02	**0.71**	0.14	0.00	**0.77**	0.03	0.00
(30, 10, 3.4)	**1.00**	0.00	0.00	**0.70**	0.00	0.00	**0.84**	0.04	0.00	**0.84**	0.00	0.00
(30, 15, 1.6)	**1.00**	0.00	0.00	**0.83**	0.03	0.00	**0.88**	0.04	0.00	**0.85**	0.04	0.00
(30, 15, 3)	**1.00**	0.00	0.00	**0.44**	0.08	0.00	**0.91**	0.01	0.00	**0.58**	0.10	0.00
(10, 20, 6)	**1.00**	0.00	0.00	**0.66**	0.20	0.00	**0.88**	0.00	0.00	**0.71**	0.18	0.00
(20, 20, 6)	**1.00**	0.00	0.00	**0.70**	0.19	0.02	**0.84**	0.02	0.00	**0.68**	0.19	0.02

Table 8 IGD metric of all algorithms.

(j,m,n)	IGD				
	RB²EA	NSGA-II	MOEA/D	FBEA	RMOEA/D
(3, 5, 2)	0.4638	0.4780	0.4759	0.4640	**0.4632**
(4, 5, 1.7)	**0.4755**	0.4926	0.4829	0.4859	0.4800
(5, 6, 2.2)	**0.4799**	0.5024	0.5050	0.4944	0.4952
(8, 7, 2.6)	**0.4263**	0.4591	0.4411	0.4412	0.4398
(9, 8, 2.4)	**0.5035**	0.5213	0.5109	0.5073	0.5102
(10, 6, 2)	**0.5089**	0.5215	0.5172	0.5104	0.5149
(10, 6, 3.5)	0.4557	0.4941	0.4866	0.4696	**0.4548**
(15, 8, 2)	**0.5068**	0.5382	0.5241	0.5214	0.5240
(15, 8, 3)	**0.5211**	0.5384	0.5369	0.5331	0.5290
(15, 4, 1.5)	0.5377	0.5537	0.5445	**0.5334**	0.5382
(10, 10, 3)	**0.5076**	0.5240	0.5231	0.5180	0.5178
(20, 5, 3)	**0.4791**	0.5006	0.5099	0.4980	0.4938
(20, 10, 1.5)	**0.5502**	0.5522	0.5527	0.5520	0.5515
(20, 10, 3)	**0.5177**	0.5473	0.5313	0.5242	0.5212
(20, 15, 3)	0.5133	0.5312	0.5191	0.5181	**0.5128**
(30, 5, 1.5)	**0.5189**	0.5433	0.5200	0.5254	0.5196
(30, 10, 1.5)	**0.5084**	0.5359	0.5313	0.5263	0.5166
(30, 10, 3.4)	**0.4960**	0.5272	0.5297	0.5156	0.5195
(30, 15, 1.6)	**0.5312**	0.5420	0.5337	0.5345	0.5325
(30, 15, 3)	**0.5117**	0.5274	0.5274	0.5214	0.5232
(10, 20, 6)	**0.4494**	0.4779	0.4672	0.4620	0.4575
(20, 20, 6)	**0.5005**	0.5228	0.5210	0.5030	0.5007

of the entire population. It also helps RB²EA develop a larger search space compared to FBEA. The optimization objectives chosen in RMOEA/D are makespan and total workload but do not include TEC. So strategies designed for TEC are missing in RMOEA/D, which results in a poor performance in EFFJSPD. In addition, compared with the dual population algorithm proposed in this article, it is more limited in search space and has weak convergence. It can be also concluded that the IGD of RB²EA is smaller than FBEA and RMOEA/D in Table 8, which validates the previous analysis.

The comparison of the average IGD values listed in Table 8 cannot reveal the obvious difference, so the box-plot of IGD in two instances is shown in Fig. 4, which indicates that RB²EA solves the problem more effectively than the other four algorithms. Figure 5 shows the approximate Pareto front by all algorithms when solving instance 20×15×3 and 30×10×3, where the solutions obtained by RB²EA have a better performance than other algorithms.

5 Conclusion and Future Work

Few studies have investigated the energy-efficient flexible job shop scheduling problem with fuzzy processing time in combination with deteriorating effect. In this study, the deterioration effect is treated as a part of the fuzzy scheduling. As for the processing time, the linear increase of a certain number is transformed into an internal linear shift of a TFN. Fuzzy makespan and fuzzy total energy consumption are chosen as the two optimization objectives. Besides, a new algorithm RB²EA is presented, which combines the bi-population evolutionary algorithm and Q-learning algorithm. Four heuristics are proposed to produce the initial population. A local search strategy based on the Q-learning and the interaction mechanism is proposed. In the experiments, RB²EA is compared to

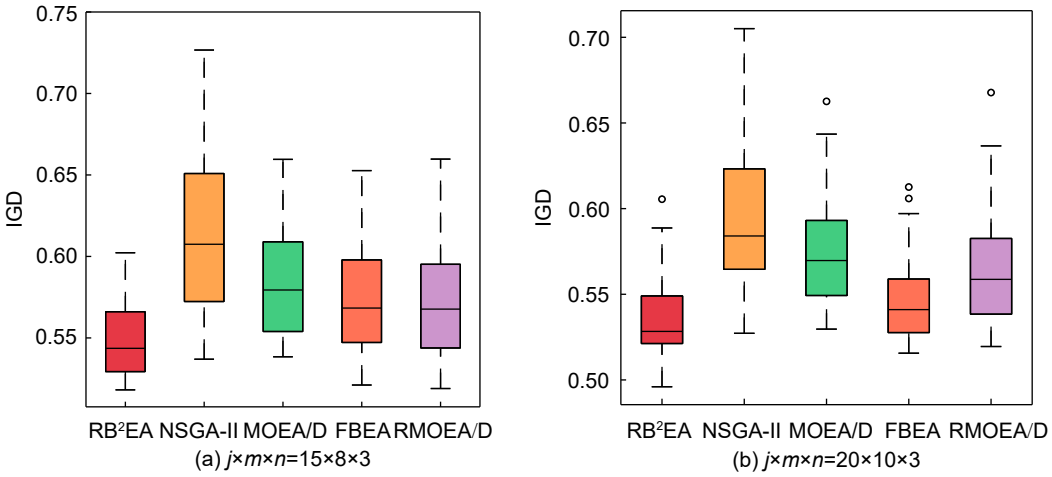

Fig. 4 IGD in instance 15×8×3 and 20×10×3.

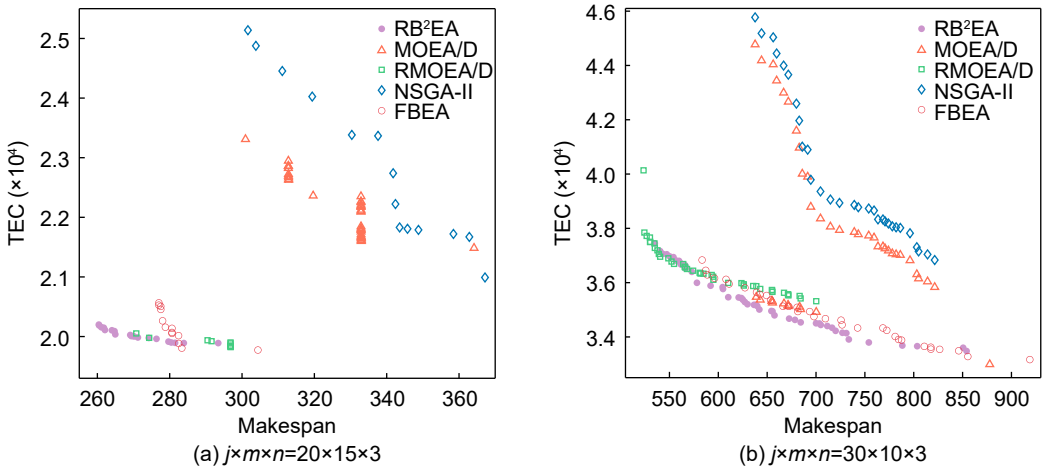

Fig. 5 Approximate Pareto front in instance 20×15×3 and 30×10×3.

other four algorithms to evalute its efficacy. RB^{2}EA is able to solve EFFJSPD efficiently.

Several main future surveys are summarized in the following:

(1) Distributed hybrid flow shop scheduling with fuzzy processing time and deteriorating effect will be investigated.

(2) We will try to adopt the type-2 fuzzy processing time and combine it with deteriorating effect.

(3) The deep reinforcement learning technique will be another direction.

References

[1] IRENA, Global energy transformation: A roadmap to 2050, https://www.h2knowledgecentre.com/content/researchpaper1605, 2019.

[2] S. Wang, M. Liu, F. Chu, and C. Chu, Bi-objective optimization of a single machine batch scheduling problem with energy cost consideration, *J. Clean. Prod.*, vol. 137, pp. 1205–1215, 2016.

[3] S. Zhang, A. Che, X. Wu, and C. Chu, Improved mixed-integer linear programming model and heuristics for bi-objective single-machine batch scheduling with energy cost consideration, *Eng. Optim.*, vol. 50, no. 8, pp. 1380–1394, 2018.

[4] S. Assia, I. E. Abbassi, A. E. Barkany, M. Darcherif, and A. E. Biyaali, Green scheduling of jobs and flexible periods of maintenance in a two-machine flowshop to minimize makespan, a measure of service level and total energy consumption, *Adv. Oper. Res.*, vol. 2020, pp. 1–9, 2020.

[5] K. Fang, W. Luo, and A. Che, Speed scaling in two-machine lot-streaming flow shops with consistent sublots, *J. Oper. Res. Soc.*, vol. 72, no. 11, pp. 2429–2441, 2021.

[6] L. P. Cota, V. N. Coelho, F. G. Guimarães, and M. J. F. Souza, Bi-criteria formulation for green scheduling with unrelated parallel machines with sequence-dependent setup times, *Int. Trans. Oper. Res.*, vol. 28, no. 2, pp. 996–1017, 2021.

[7] H. Zhang, Y. Wu, R. Pan, and G. Xu, Two-stage parallel speed-scaling machine scheduling under time-of-use tariffs, *J. Intell. Manuf.*, vol. 32, no. 1, pp. 91–112, 2021.

[8] F. Zhao, X. He, and L. Wang, A two-stage cooperative evolutionary algorithm with problem-specific knowledge for energy-efficient scheduling of no-wait flow-shop problem, *IEEE Trans. Cybern.*, vol. 51, no. 11, pp. 5291–5303, 2021.

[9] F. Zhao, R. Ma, and L. Wang, A self-learning discrete jaya algorithm for multiobjective energy-efficient distributed no-idle flow-shop scheduling problem in heterogeneous factory system, *IEEE Trans. Cybern.*, vol. 52, no. 12, pp. 12675–12686, 2022.

[10] 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 Evol. Comput*., vol. 69, p. 100992, 2022.

[11] A. Goli, A. Ala, and M. Hajiaghaei-Keshteli, Efficient multi-objective meta-heuristic algorithms for energy-aware non-permutation flow-shop scheduling problem, *Expert Syst. Appl.*, vol. 213, p. 119077, 2023.

[12] X. Wu, Z. Cao, and S. Wu, Real-time hybrid flow shop scheduling approach in smart manufacturing environment, *Complex System Modeling and Simulation*, vol. 1, no. 4, pp. 335–350, 2021.

[13] F. Zhao, B. Zhu, and L. Wang, An estimation of distribution algorithm-based hyper-heuristic for the distributed assembly mixed no-idle permutation flowshop scheduling problem, *IEEE Trans. Syst. Man Cybern, Syst.*, vol. 53, no. 9, pp. 5626–5637, 2023.

[14] J. Y. Moon and J. Park, Smart production scheduling with time-dependent and machine-dependent electricity cost by considering distributed energy resources and energy storage, *Int. J. Prod. Res*., vol. 52, no. 13, pp. 3922–3939, 2014.

[15] D. Lei, Y. Zheng, and X. Guo, A shuffled frog-leaping algorithm for flexible job shop scheduling with the consideration of energy consumption, *Int. J. Prod. Res.*, vol. 55, no. 11, pp. 3126–3140, 2017.

[16] X. Wu and Y. Sun, A green scheduling algorithm for flexible job shop with energy-saving measures, *J. Clean. Prod.*, vol. 172, pp. 3249–3264, 2018.

[17] S. Luo, L. Zhang, and Y. Fan, Energy-efficient scheduling for multi-objective flexible job shops with variable processing speeds by grey wolf optimization, *J. Clean. Prod.*, vol. 234, pp. 1365–1384, 2019.

[18] L. Meng, C. Zhang, X. Shao, and Y. Ren, MILP models for energy-aware flexible job shop scheduling problem, *J. Clean. Prod.*, vol. 210, pp. 710–723, 2019.

[19] M. Li and D. Lei, An imperialist competitive algorithm with feedback for energy-efficient flexible job shop scheduling with transportation and sequence-dependent setup times, *Eng. Appl. Artif. Intell.*, vol. 103, p. 104307, 2021.

[20] H. Qin, Y. Han, Q. Chen, L. Wang, Y. Wang, J. Li, and Y. Liu, Energy-efficient iterative greedy algorithm for the distributed hybrid flow shop scheduling with blocking constraints, *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 7, no. 5, pp. 1442–1457, 2023.

[21] M. Dai, D. Tang, A. Giret, and M. A. Salido, Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints, *Robot. Comput. Integr. Manuf.*, vol. 59, pp. 143–157, 2019.

[22] Z. Pan, D. Lei, and L. Wang, A bi-population evolutionary algorithm with feedback for energy-efficient fuzzy flexible job shop scheduling, *IEEE Trans. Syst. Man Cybern, Syst.*, vol. 52, no. 8, pp. 5295–5307, 2022.

[23] R. Li, W. Gong, and C. Lu, A reinforcement learning based RMOEA/D for bi-objective fuzzy flexible job shop

scheduling, *Expert Syst. Appl.*, vol. 203, p. 117380, 2022.

[24] J. Q. Li, Z. M. Liu, C. Li, and Z. X. Zheng, Improved artificial immune system algorithm for type-2 fuzzy flexible job shop scheduling problem, *IEEE Trans. Fuzzy Syst*., vol. 29, no. 11, p. 3234–3248, 2021.

[25] F. Zhao, Z. Xu, L. Wang, N. Zhu, T. Xu, and J. Jonrinaldi, A population-based iterated greedy algorithm for distributed assembly no-wait flow-shop scheduling problem, *IEEE Trans. Ind. Inf.*, vol. 19, no. 5, pp. 6692–6705, 2023.

[26] L. Sun, L. Lin, M. Gen, and H. Li, A hybrid cooperative coevolution algorithm for fuzzy flexible job shop scheduling, *IEEE Trans. Fuzzy Syst.*, vol. 27, no. 5, pp. 1008–1022, 2019.

[27] J. Lin, Backtracking search based hyper-heuristic for the flexible job-shop scheduling problem with fuzzy processing time, *Eng. Appl. Artif. Intell.*, vol. 77, pp. 186–196, 2019.

[28] J. Cai and D. Lei, A cooperated shuffled frog-leaping algorithm for distributed energy-efficient hybrid flow shop scheduling with fuzzy processing time, *Complex Intell. Syst.*, vol. 7, no. 5, pp. 2235–2253, 2021.

[29] R. Li, W. Gong, and C. Lu, Self-adaptive multi-objective evolutionary algorithm for flexible job shop scheduling with fuzzy processing time, *Comput. Ind. Eng.*, vol. 168, p. 108099, 2022.

[30] Y. Dorfeshan, R. Tavakkoli-Moghaddam, S. M. Mousavi, and B. Vahedi-Nouri, A new weighted distance-based approximation methodology for flow shop scheduling group decisions under the interval-valued fuzzy processing time, *Appl. Soft Comput.*, vol. 91, p. 106248, 2020.

[31] R. Li, W. Gong, C. Lu, and L. Wang, A learning-based memetic algorithm for energy-efficient flexible job-shop scheduling with type-2 fuzzy processing time, *IEEE Trans. Evol. Computat.*, vol. 27, no. 3, pp. 610–620, 2023.

[32] B. Xi and D. Lei, Q-learning-based teaching-learning optimization for distributed two-stage hybrid flow shop scheduling with fuzzy processing time, *Complex System Modeling and Simulation*, vol. 2, no. 2, pp. 113–129, 2022.

[33] J. N. D. Gupta and S. K. Gupta, Single facility scheduling with nonlinear processing times, *Comput. Ind. Eng.*, vol. 14, no. 4, pp. 387–393, 1988.

[34] S. Browne and U. Yechiali, Scheduling deteriorating jobs on a single processor, *Oper. Res.*, vol. 38, no. 3, pp. 495–498, 1990.

[35] Y. Fu, M. Zhou, X. Guo, and L. Qi, Scheduling dual-objective stochastic hybrid flow shop with deteriorating jobs via bi-population evolutionary algorithm, *IEEE Trans. Syst. Man Cybern, Syst.*, vol. 50, no. 12, pp. 5037–5048, 2020.

[36] T. Jiang, H. Zhu, L. Liu, and Q. Gong, Energy-conscious flexible job shop scheduling problem considering transportation time and deterioration effect simultaneously, *Sustain. Comput. Inform. Syst.*, vol. 35, p. 100680, 2022.

[37] Y. F. Wang, Adaptive job shop scheduling strategy based on weighted Q-learning algorithm, *J. Intell. Manuf.*, vol. 31, no. 2, pp. 417–432, 2020.

[38] B. A. Han and J. J. Yang, Research on adaptive job shop scheduling problems based on dueling double DQN, *IEEE Access*, vol. 8, pp. 186474–186495, 2020.

[39] L. Wang and Z. Pan, Scheduling optimization for flow-shop based on deep reinforcement learning and iterative greedy method, *Control and Decision*, vol. 36, no. 11, pp. 2609–2617, 2021.

[40] L. Wang, X. Hu, Y. Wang, S. Xu, S. Ma, K. Yang, Z. Liu, and W. Wang, Dynamic job-shop scheduling in smart manufacturing using deep reinforcement learning, *Comput. Netw.*, vol. 190, p. 107969, 2021.

[41] J. J. Wang and L. Wang, A cooperative memetic algorithm with learning-based agent for energy-aware distributed hybrid flow-shop scheduling, *IEEE Trans. Evol. Computat*., vol. 26, no. 3, pp. 461–475, 2022.

[42] F. Zhao, S. Di, and L. Wang, A hyperheuristic with Q-learning for the multiobjective energy-efficient distributed blocking flow shop scheduling problem, *IEEE Trans. Cybern.*, vol. 53, no. 5, pp. 3337–3350, 2023.

[43] F. Zhao, T. Jiang, and L. Wang, A reinforcement learning driven cooperative meta-heuristic algorithm for energy-efficient distributed no-wait flow-shop scheduling with sequence-dependent setup time, *IEEE Trans. Ind. Inf.*, vol. 19, no. 7, pp. 8427–8440, 2023.

[44] H. Li, K. Gao, P. Y. Duan, J. Q. Li, and L. Zhang, An improved artificial bee colony algorithm with Q-learning for solving permutation flow-shop scheduling problems, *IEEE Trans. Syst. Man Cybern, Syst*., vol. 53, no. 5, pp. 2684–2693, 2023.

[45] F. Zhao, G. Zhou, and L. Wang, A cooperative scatter search with reinforcement learning mechanism for the distributed permutation flowshop scheduling problem with sequence-dependent setup times, *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 53, no. 8, pp. 4899–4911, 2023.

[46] L. A. Zadeh, Fuzzy sets, *Inf. Contr.*, vol. 8, no. 3, pp. 338–353, 1965.

[47] Q. Zhang and H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, *IEEE Trans. Evol. Computat.*, vol. 11, no. 6, pp. 712–731, 2007.

[48] R. Storn and K. Price, Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces, *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[49] R. Liu, J. Li, J. Fan, C. Mu, and L. Jiao, A coevolutionary technique based on multi-swarm particle swarm optimization for dynamic multi-objective optimization, *Eur. J. Oper. Res.*, vol. 261, no. 3, pp. 1028–1051, 2017.

[50] C. J. C. H. Watkins and P. Dayan, Technical note: Q-learning, *Mach. Learn.*, vol. 8, nos. 3&4, pp. 279–292, 1992.

[51] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Trans. Evol. Computat.*, vol. 6, no. 2, pp. 182–197,

2002.
[52] K. M. Lee, T. Yamakawa, and K. M. Lee, A genetic algorithm for general machine scheduling problems, in *Proc. 1998 2nd Int. Conf. Knowledge-Based Intelligent Electronic Systems, Proc. KES'98* (*Cat. No. 98EX111*), Adelaide, Australia, 1998, pp. 60–66.
[53] P. Brandimarte, Routing and scheduling in a flexible job shop by tabu search, *Ann. Oper. Res.*, vol. 41, no. 3, pp. 157–183, 1993.
[54] M. Saidi-Mehrabad and P. Fattahi, Flexible job shop scheduling with tabu search algorithms, *Int. J. Adv. Manuf. Technol.*, vol. 32, nos. 5&6, pp. 563–570, 2007.
[55] D. Behnke and M. J. Geiger, Test instances for the flexible job shop scheduling problem with work centers, https://openhsu.ub.hsu-hh.de/handle/10.24405/436, 2012.
[56] D. C. Montgomery, R. H. Myers, W. H. Carter, and G. G. Vining, The hierarchy principle in designed industrial experiments, *Qual. Reliab. Engng. Int.*, vol. 21, no. 2, pp. 197–201, 2005.
[57] L. While, P. Hingston, L. Barone, and S. Huband, A faster algorithm for calculating hypervolume, *IEEE Trans. Evol. Computat.*, vol. 10, no. 1, pp. 29–38, 2006.
[58] C. A. Coello and N. C. Cortés, Solving multiobjective optimization problems using an artificial immune system, *Genet. Program. Evolvable Mach.*, vol. 6, no. 2, pp. 163–190, 2005.

Libao Deng received the BSc, MSc, and PhD degrees from Harbin Institute of Technology, Harbin, China in 2004, 2007, and 2012, respectively. He is currently a professor at the School of Information Science and Engineering, Harbin Institute of Technology, Weihai, China. His research interests are in computational intelligence and optimal scheduling.

Lili Zhang received the bachelor and master degrees in instrumental science and technology from Harbin Institute of Technology, China in 2017 and 2019, respectively, and the PhD degree in computer science f'rom Dublin City University, Ireland in 2022. Now she is a postdoctoral researcher at Dublin City University, Ireland and a lecturer at Maynooth University, Ireland. Her research area involves intelligent computing and computational psychiatry.

Yingjian Zhu is currently pursuing the BSc degree at Harbin Institute of Technology, Weihai, China. His main research interest includes flexible job shop scheduling with intelligent optimization.

Yuanzhu Di received the BSc and MSc degrees from Harbin Institute of Technology, Weihai, China in 2021 and 2023, respectively. She is currently pursuing the PhD degree at Harbin Institute of Technology, Weihai, China. Her main research directions include the distributed and green scheduling with computational intelligence.