IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 12 May 2023, accepted 25 May 2023, date of publication 30 May 2023, date of current version 7 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3281364

RESEARCH ARTICLE

Multi-Strategy Dynamic Evolution-Based Improved MOEA/D Algorithm for Solving Multi-Objective Fuzzy Flexible Job Shop Scheduling Problem

ZHENGGANG LIU, XU LIANG, LINGYAN HOU, DALI YANG, AND QIANG TONG
Beijing Information Science and Technology University, Beijing 100101, China
Corresponding author: Xu Liang (20212525@bistu.edu.cn)

ABSTRACT A scheduling model was developed to optimize the maximum completion time, total machine load, and maximum machine load for the fuzzy flexible job shop problem with uncertain processing times. To solve this problem, a multi-strategy dynamic evolution-based improved multi-objective evolutionary algorithm based on decomposition(IMOEA/D) was proposed. In order to enhance the quality of the non-dominated solution set and improve the algorithm efficiency. The algorithm firstly employs a strategy based on minimum processing time and workload, along with a non-dominated solution prioritization mechanism to generate the initial population. Secondly, three evolutionary strategies are incorporated, and their probabilities are dynamically adjusted with the increase of evolution generations. Finally, a variable neighborhood search method is introduced to improve the search performance of the algorithm. The effectiveness of the proposed algorithm was demonstrated through experimental validation.

INDEX TERMS Fuzzy flexible job shop scheduling, evolutionary algorithm, multi-objective optimization, dynamic evolution, variable neighborhood search.

I. INTRODUCTION

With the fierce competition in the global market, as well as the growing concerns over energy consumption and environmental sustainability, modern manufacturing enterprises are increasingly paying attention to various performance indicators throughout the production process. Efficient scheduling methods play a key role in enhancing the productivity of manufacturing enterprises, both in decision-making for production and in the allocation of resources for intelligent manufacturing. In recent years, due to the widespread application of multi-objective production scheduling problems, the research on multi-objective flexible job shop scheduling problems (MOFJSP) has attracted increasing attention from scholars. Due to the widespread application of multi-objective production scheduling problems, the research on MOFJSP has received increasing attention from scholars.

In recent years, there has been a growing body of research by domestic and foreign scholars on MOFJSP. Devi et al. [1] introduced a hybrid adaptive firefly algorithm that employs two adaptive strategies and targets completion time, maximum machine load, and total workload as optimization objectives. This algorithm demonstrated favorable performance in multi-objective optimization. Wen et al. [2] considered the green manufacturing paradigm and developed a green multi-objective flexible job shop scheduling model. They also designed a two-stage solving framework based on NSGA-II to generate scheduling plans. Baykasoğlu and Madenoğlu [3] proposed a construction algorithm for MOFJSP with machine capacity constraints and sequence-dependent setup times. The algorithm utilizes a greedy randomized adaptive search algorithm. Geng et al. [4] proposed an improved multi-objective ant

The associate editor coordinating the review of this manuscript and approving it for publication was Okyay Kaynak.

colony optimization algorithm (IMOALO) for the reentrant hybrid flow shop scheduling problem, with the goal of minimizing both makespan and energy consumption costs. Jiang et al. [5] formulated a batch division scheduling model with the objectives of minimizing total energy consumption, manufacturing span, and processing cost, based on a real case of energy-saving scheduling problem for complex aerospace components. They also proposed a novel and improved crossover artificial bee colony algorithm. Gao et al. [6] introduced a meta-heuristic algorithm, namely the Jaya algorithm, to address the rescheduling problem in a flexible job shop with the addition of new jobs. The objective of the algorithm is to optimize the completion time, machine workload, and overall processing time.

In the above-mentioned research, the machine processing time is considered as a constant value. However, in actual production, there are many dynamic and uncertain factors in the manufacturing process, such as unexpected machine breakdowns and maintenance, rework of defective products, varying worker skill levels, and changes in order demand, so the start and completion times of processes often fluctuate within a certain time interval. Therefore, the fuzzy theory, which is based on fuzzy set theory, can be used in actual manufacturing workshop production to represent the uncertain processing time. Based on fuzzy flexible job shop scheduling problem (FFJSP), it can reduce the impact of uncertain events in the production process and better meet the needs of actual production scheduling. Gnanavelbabu et al. [7] investigated the the FFJSP with worker flexibility and uncertain processing times. Their objective was to minimize both the makespan and the standard deviation of the makespan, and they integrated Monte Carlo simulation into a multi-objective improved backtracking search algorithm framework. Zhu and Zhou [8] investigated the FFJSP with job priority constraints and proposed a multi-objective hierarchical optimization algorithm based on interval grey degree, total workload of interval grey degree, and average lateness index. Lei [9] proposed a cooperative coevolutionary algorithm (CGA) to minimize the fuzzy maximum completion time. The populations of job sequencing and machine assignment evolve independently and cooperate to converge to the optimal solution of the problem. Liu et al. [10] nvestigated the dynamic flexible job shop scheduling problem with fuzzy processing times. They simplified the original dynamic shop to a traditional static fuzzy flexible job shop problem and employed an estimation of distribution algorithm (EDA) to solve the transformed problem.. The FFJSP meets the demands of practical applications, but obtaining near-optimal solutions is challenging and further research is required.

Efficient optimization algorithms can help enterprises improve production efficiency, and intelligent optimization algorithms have always been the main solution for scheduling problems. Multi-objective evolutionary algorithm based on decompositions can decompose a multi-objective problem into multiple sub-problems for solution. They have strong search capabilities, efficient fitness evaluation, and good convergence speed, making them a hot topic in the research of workshop scheduling in recent years. Li et al. [11] proposed a MOEA/D algorithm based on reinforcement learning and designed a parameter adaptation strategy based on Q-learning to guide the population to select the best parameters and increase diversity. Yang et al. [12] considered the problem of minimizing both manufacturing time and energy consumption under strong transportation constraints. They applied two methods, namely non-dominated sorting genetic algorithm-II and decomposition-based multi-objective evolutionary algorithm, to solve the problem in a real-world case study. Wang and Peng [13] studied the distributed job shop scheduling problem and proposed an improved decomposed multi-objective evolutionary algorithm (MMOEA/D). They achieved good results by using collaborative search. Zhou and Liao [14] proposed a subpopulation hybrid decomposed multi-objective evolutionary algorithm based on MOEA/D and particle swarm optimization for the green scheduling problem in flexible job shop with crane transportation. Xixing et al. [15] described a flexible job shop scheduling problem with dual resource constraints and proposed a decomposition-based multi-objective evolutionary algorithm to simplify the solution process. From the current research results achieved domestically and abroad, there are relatively few achievements in effectively solving the multi-objective fuzzy flexible job shop scheduling problem (MOFFJSP). Indeed, the standard MOEA/D algorithm works well for low-dimensional and simple Pareto front multi-objective optimization problems, but for complex Pareto front problems and high-dimensional multi-objective optimization problems, its distribution performance will be greatly compromised.

Improving the distribution and convergence of the non-dominated solution set by enhancing classical algorithms has become an important direction in the research of multi-objective evolutionary algorithms. Crossing and mutation are the main drivers of evolution, and appropriate crossing and mutation strategies can effectively improve evolutionary efficiency. Local search methods can strengthen the concentrated exploration of the solution space and improve the quality of solutions. Applying local search strategies to evolutionary algorithms can effectively improve the convergence of the algorithm. Inspired by the above ideas, to efficiently solve the MOFFJSP. This article proposes an improved MOEA/D algorithm that changes its single crossover and mutation strategy by using a multi-strategy approach to generate new solutions, and allows each strategy to perform crossover and mutation within its suitable range. Additionally, a variable neighborhood search strategy is embedded into the MOEA/D algorithm. On one hand, the algorithm uses the neighborhood search strategy to improve the convergence of the non-dominated solution set. On the other hand, by using a multi-strategy crossover and mutation approach, the algorithm improves the global search

capability and enhances the uniformity of the distribution of the non-dominated solution set.

The main contributions of this paper go in four directions.

(1) Using three strategies based on local minimum processing time, global minimum machine load, and randomly generated chromosomes to generate an excellent initial population using the non-dominated solution priority selection principle.

(2) Changing the traditional single crossover and mutation strategy, we propose three crossover strategies. This involves performing crossover and mutation for each strategy within its suitable range, and dynamically adjusting the probabilities of the three crossover strategies based on the evolutionary generation. Simultaneously enhancing the algorithm's global search ability while improving its local search capability.

(3) We designed a variable neighborhood search combined with four local search strategies to further enhance the search ability of the algorithm.

(4) The proposed algorithm is compared and analyzed with two other optimization algorithms to demonstrate its superiority.

II. PROBLEM DESCRIPTION

The MOFFJSP can be described as follows: the job shop consists of n jobs $(J_1, J_2, \ldots, J_n)$ and m machines $(M_1, M_2, \ldots, M_m)$, where each job J_i comprises one or more operations $O_{ij}, j :\in (1, 2, \ldots, n_i)$, resulting in a total number of n_i operations for job J_i. The processing of the job's operations follows a given routing, and operation O_{ij} can be optionally processed on machine $M_k :\in M$. The processing time of the j operation of job J_i on machine M_k is represented by a triangular fuzzy number $\tilde{T_{ijk}} = (t_1, t_2, t_3)$. t_1 represents the minimum processing time of the operation, t_2 represents the most probable processing time, and t_3 represents the maximum processing time of the operation.

The scheduling objective is to assign a suitable machine to each operation, and sequence all operations assigned to each machine to determine their start time, in order to achieve the optimal optimization objective.

In addition, the following assumptions are also considered: All machines are assumed to be independent of each other and available at time zero. At any given time, each machine can process only one operation at a time. An operation cannot be interrupted once it has started processing. Precedence constraints exist between operations of the same job, while there are no precedence constraints between operations of different jobs. All jobs have the same priority. Machine breakdown and setup time are not considered.

III. PROBLEM DESCRIPTION AND MATHEMATICAL MODELING

A. TRIANGULAR FUZZY NUMBER OPERATION RULES

This paper uses triangular fuzzy numbers [16] to represent operation processing time.

$$\tilde{T} = (t_1, t_2, t_3) \tag{1}$$

In the formula, t_1 represents the optimistic operation processing time, which is the shortest processing time of the operation; t_2 represents the most likely processing time, which is assumed to be the median of the historical processing time distribution; t_3 represents the pessimistic operation processing time, which is the longest processing time of the operation. The membership function of the triangular fuzzy number is given by:

$$\mu(x) = \begin{cases} 0, & x < t_1 \\ \dfrac{t - t_1}{t_2 - t_1}, & t_1 \le t \le t_2 \\ \dfrac{t_3 - t}{t_3 - t_2}, & t_2 < t \le t_3 \\ 0, & x > t_3 \end{cases} \tag{2}$$

For two triangular fuzzy numbers $\tilde{X}(x_1, x_2, x_3)$ and $\tilde{Y}(y_1, y_2, y_3)$, their arithmetic operations are defined as follows:

Addition operation: $\tilde{X} + \tilde{Y} = (x_1 + y_1, x_2 + y_2, x_3 + y_3)$

Comparison operation: $F(\tilde{X}) = (x_1 + 2x_2 + x_3)/4$, $F(\tilde{Y}) = (y_1 + 2y_2 + y_3)/4$.

if $F(\tilde{X}) > F(\tilde{Y})$, then$\tilde{X} > \tilde{Y}$. on the contrary $\tilde{X} < \tilde{Y}$.

if $F(\tilde{X}) = F(\tilde{Y})$, then compare x_2 with y_2. if $x_2 > y_2$, then $\tilde{X} > \tilde{Y}$. on the contrary $\tilde{X} < \tilde{Y}$.

if $F(\tilde{X}) = F(\tilde{Y})$, then compare x_2 with y_2. if $x_2 > y_2$, then $\tilde{X} > \tilde{Y}$. on the contrary $\tilde{X} < \tilde{Y}$.

Max operation: if $\tilde{X} > \tilde{Y}$, then $\tilde{X} \vee \tilde{Y} = \tilde{X}$. on the contrary $\tilde{X} \vee \tilde{Y} = \tilde{Y}$

Fuzzy number addition operation is used to determine the fuzzy completion time of the operation, fuzzy number comparison operation is used to compare the fuzzy completion times of all jobs to obtain the maximum fuzzy completion time of the entire schedule, and fuzzy number maximum operation is used to determine the fuzzy start time of each operation.

B. THE MATHEMATICAL MODEL OF THE MOFFJSP

$$minF_1 = max\tilde{T}_i, 1 \le i \le n \tag{3}$$

$$minF_2 = \sum_{k=1}^{M} W_k, 1 \le k \le m \tag{4}$$

$$minF_3 = maxW_k, 1 \le k \le m \tag{5}$$

$\tilde{T}_i$ represents the fuzzy completion time of job i, while W_k denotes the load on machine k. Eqs. (3), (4), and (5) are three objective functions, Eq. (3), is the maximum fuzzy completion time of all operations, Eq. (4) is the total machine load and Eq.(5) is the maximum machine load.

$$\tilde{T_{ij}} - \tilde{T_{i(j+1)}} \ge \tilde{P_{i(j+1)k}}, \quad 1 \le i \le n, 1 \le j < h_i, 1 \le k \le m \tag{6}$$

$$(\tilde{T_{i1j1}} - \tilde{P_{i1j1k}} - \tilde{T_{i2j2}})\tilde{X}_{i1j1k}\tilde{X}_{i2j2k} \ge 0, \quad 1 \le i1, i2 \le n, 1 \le j1 \le h_{i1}, 1 \le j2 \le h_{i2}, 1 \le k \le m \tag{7}$$

$$W_k = \sum_{i=1}^{n} \sum_{j=1}^{h_i} \tilde{P_{ijk}} \tilde{X}_{ijk}, \tag{8}$$

$$1 \le i \le n, 1 \le j < h_j, 1 \le k \le mT_i = T_{ih_i}, 1 \le i \le n \tag{9}$$

$\tilde{T_{ij}}$ represents the fuzzy completion time of the j-th process of job i. $\tilde{P_{ijk}}$ is the fuzzy processing time of the j-th operation of Job i on Machine k. h_i represents the total number of operations for job i, and X_{ijk} represents a binary variable that takes on values of 0 or 1. When the j-th operation of job i is processed on machine k, X is equal to 1; otherwise, it is equal to 0. Eq. (6) indicates that the subsequent operation of a job cannot start until the previous operation of the same job is completed. Eq. (7) indicates that no machine can process more than one job at any given time. Eq. (8) represents the total workload of machine k, and Eq. (9) represents the fuzzy completion time of job i.

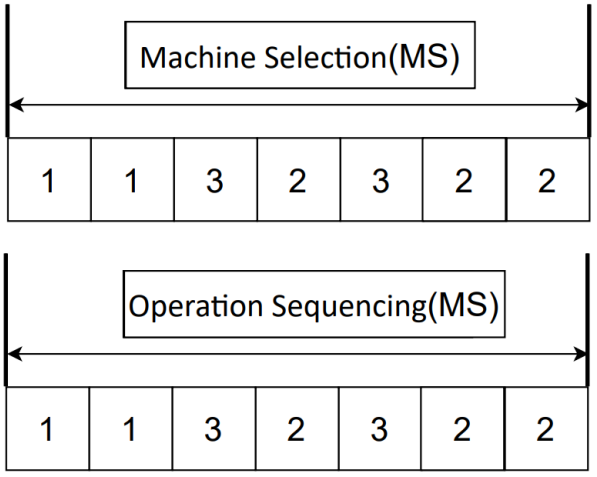


FIGURE 1. Encoding representation.

IV. PROPOSED ALGORITHM

A. IMPROVEMENT STRATEGY

For the solution of the MOFFJSP, previous research has mostly used local search strategies to improve the convergence of the algorithms. However, after several rounds of local search, the algorithm will concentrate the population around a few points, greatly reducing the diversity of the generated non-dominated solution set. Some algorithms have proposed crowding strategies to address this issue, but the results are not satisfactory. MOEA/D uses weight vectors and Tchybecheff function, which produce solutions that are distributed in the weight vector space, while simultaneously balancing convergence and diversity. In order to improve the convergence and distribution of the Pareto-optimal set and enhance the performance of the algorithm, this paper proposes an improved MOEA/D (IMOEA/D) algorithm based on the previous work. Firstly, three strategies based on local minimum processing time, global minimum machine load, and random chromosome generation are used to generate the initial population. The non-dominated sorting principle is employed to optimize the initial population and ensure its quality and randomness. Then, three evolutionary strategies are selected with emphasis on the early, middle, and late stages of iteration, and the probability of each strategy being selected is dynamically adjusted by calculating their evolution rates, to adapt to the different requirements of evolutionary strategies during different evolutionary periods. Finally, four variable neighborhood search strategies combined with local search strategies were designed to further enhance the search capability of the algorithm.

B. ENCODING AND DECODING

The MOFFJSP involves the selection of multiple jobs, multiple operations, and multiple machines, which belongs to a typical discrete problem. The chromosome adopts a double-layer encoding form, consisting of the machine selection part (MS) and the operation sequencing part (OS), and the chromosome length is equal to the total number of machining operations. The MS part of the chromosome represents the selection of the processing machine for each operation, and each bit corresponds to a specific operation. The OS part of the chromosome represents the processing order of the jobs, with each number indicating a specific job and the frequency of appearance of each number in the chromosome representing the sequence of processing for each job.

Fig.1 displays an encoded solution. The first digit represents the processing of operation O_{31} on machine 1, and the second digit represents the processing of operation O_{11} on machine 1. Therefore, the processing sequence of all operations is $O_{31} \rightarrow O_{11} \rightarrow O_{21} \rightarrow O_{12} \rightarrow O_{22} \rightarrow O_{23} \rightarrow O_{32}$. This encoding method ensures the feasibility of the chromosome solutions generated in subsequent operations, and has no requirements on the length of the workpiece's process and the number of workpieces, thus avoiding subsequent complex corrective operations and achieving simplicity and flexibility.

C. INITIALIZATION STRATEGY

Population initialization is one of the crucial steps in an algorithm, and the quality of the initial solutions can significantly affect the speed and quality of the genetic algorithm's solution.

We propose to use three strategies: local minimum processing time strategy, global minimum workload strategy, and random strategy.

(1)The local minimum processing time strategy selects the machine with the minimum processing time from the candidate set for each operation to generate chromosomes, aiming to reduce the fuzzy completion time.

(2)The global minimum workload strategy selects the available machine with the minimum workload from the candidate set for each operation, aiming to reduce the total workload of the machines.

(3)The random strategy has a simple rule and can ensure that the initial population has a high degree of diversity.

After applying the local minimum processing time strategy and the global minimum workload strategy, two populations P_1 and P_2 with size N are generated respectively, and then these two populations are merged. The merged population is sorted using the fast nondominated sorting algorithm and dominated solutions are eliminated. If the population size after the operation is greater than N, the top N individuals are selected as the initial population. If it is less than N, the random strategy is used to add individuals to make up the population.

D. CROSSOVER AND MUTATION

To enhance the search capability and avoid premature convergence of the IMOEA/D algorithm, we adopt three chromosome crossing-over strategies during the crossover process.

a. Selecting a random individual from the population as a reference individual, and then generate new individuals by recombining this individual with two other randomly selected individuals. This approach has the advantages of a higher global search capability, good global convergence performance, and a reduced likelihood of getting stuck in local optima. However, it has the disadvantage of slower convergence speed.

b. Selecting the best-performing individual in the contemporary population as a reference individual, and recombining it with a stochastic differential vector to generate new individuals is a strategy that exhibits a noteworthy ability to optimize locally, leading to rapid convergence rates. Additionally, it facilitates the effective inheritance of the most successful individuals from each generation. However, the strategy's ability to explore the global search space is relatively limited, rendering the algorithm vulnerable to becoming trapped in local optima.

c. Selecting a random individual from the population as a reference individual, then generating a fixed vector by combining this individual with the best-performing individual from the current population. This vector is then recombined with two vectors generated by randomly selected individuals to create new individuals. This strategy has the advantage of effectively balancing global search capability and local optimization ability, but has a relatively weaker robustness.

Based on the above analysis, although different evolutionary strategies have differences in search ability and search range, their ways of generating new individuals are basically the same, which is to produce new candidate individuals through a linear combination of a base individual and differential vectors. Due to the commonality in structure and evolutionary patterns among different mutation strategies, they can complement and cooperate with each other to improve the search performance.

In the early stages of evolution, strategy a is more likely to be selected, as the population has sufficient evolutionary momentum and is in a fast-evolving phase that emphasizes convergence, but with a slight lack of diversity. Strategy a enhances diversity under the condition of rapid convergence. In the middle stage of evolution, both mutation rate a and strategy c are more likely to be selected, but the probability of strategy c being selected is higher. At this stage, the evolutionary momentum of the population is weak and it is in a slow evolution phase. It is difficult to converge to the ideal Pareto front. Strategy a and c provide evolutionary momentum and enhance its convergence to prevent the population from getting stuck in a stagnant state. In the later stages of evolution, strategy b is more likely to be selected. At this time, the population gradually converges to the Pareto front, and the algorithm enters the stage of local tuning. It is important to focus on improving the algorithm's diversity and its ability in local search, optimization, and replacement, in order to make the distribution of solutions more uniform when converging to the Pareto front. Therefore, the probabilities of the three strategies need to be dynamically adjusted with the evolution generation. The evolution probabilities of the three strategies are P_1, P_2, P_3, and initially $P_1 = P_2 = P_3 = 1/3$. During each generation of the evolutionary process, the success rate of using each evolutionary strategy to update the parent population is calculated. This involves counting the number of individuals in the current population that have used each of the three strategies (n1, n2, n3), as well as the number of individuals that were successfully updated using each strategy (m1, m2, m3). The probabilities of selecting each strategy (p1, p2, p3) are then dynamically adjusted according to the following formula:

$$P_1 = \frac{m_1/n_1}{m_1/n_1 + m_2/n_2 + m_3/n_3} \tag{10}$$

$$P_2 = \frac{m_2/n_2}{m_1/n_1 + m_2/n_2 + m_3/n_3} \tag{11}$$

$$P_3 = \frac{m_3/n_3}{m_1/n_1 + m_2/n_2 + m_3/n_3} \tag{12}$$

After each generation, the algorithm can automatically adjust the mutation strategy that is suitable for the current stage based on the success rate of evolution. For the mutation strategy that is suitable for this stage, we will increase the proportion of its implementation in the algorithm, while for the mutation strategy that is not suitable for this stage, we will reduce its proportion in the algorithm. By adopting a dynamic population evolution model, we are able to adjust the probabilities of different strategies in real-time based on their impact on the current population evolution, thus avoiding the stagnation of population evolution and effectively improving the convergence and evolution efficiency of the algorithm. In addition, since the three evolution mutation strategies participate simultaneously and cooperate with each other, this model can ensure that the algorithm has diversity.

Mutation operations can play a role in expanding randomness and increasing the search ability of the algorithm. In the initial stage of evolution, a small mutation probability is needed, and as many good genes as possible should be preserved. In the later stages, it is necessary to increase the mutation probability appropriately to generate gene diversity and avoid premature convergence. Therefore, dynamic and

adaptive mutation probability is used in the iterative process. The formula for calculating the mutation probability is as follows:

$$P_N = P_0 + \rho(\frac{GEN}{MAX_{GEN}}) \tag{13}$$

Eq. (13) P_0 is the initial mutation probability, ρ is the maximum rate of change, GEN is the current evolutionary generation, and MAX_{GEN} is the total evolutionary generation.

E. VARIABLE NEIGHBORHOOD SEARCH

We have chosen to embed the variable neighborhood algorithm in MOEA/D to enhance the local search capability of the algorithm. A total of four key-path-based neighborhood search structures have been designed as follows:

N1 (minimum processing time): Select any operation and choose the machine with the shortest processing time for the current operation from its available machine set.

N2 (Insertion): Randomly select two positions of operations and insert the later operation to the position before the earlier operation.

N3 (Swap): Randomly select two positions of operations and swap the positions of these two operations in the sequence of operations.

N4 (Permutation): Select two positions randomly in the feasible sequence of operations, and reverse the order of operations in between.

During the variable neighborhood search, the aforementioned operations are applied in sequence. If the new solution generated by the neighborhood structure N1 is better than the current solution, the new solution is accepted and replaces the current solution, and then the N1 neighborhood structure is continued to be used. If the current solution cannot be improved, the algorithm moves to the next neighborhood structure for searching, and this process continues until the stopping iteration is reached.

F. ALGORITHM DESCRIPTIONS

The improved decomposition-based multi-objective evolutionary algorithm (IMOEA/D) is proposed by integrating the MOEA/D algorithm with population initialization, crossover and mutation strategies, and variable neighborhood search discussed in the previous section. The specific steps are as follows:

Step 1: Set the algorithm parameters, including population size and maximum iteration number.

Step 2: Initialize the weight vectors uniformly distributed in the solution space. Initialize neighborhood size and reference points based on the weight vectors.

Step 3: Generate the initial population using the initialization strategy and non-dominated sorting.

Step 4: Check if the algorithm termination criteria are met. If yes, proceed to Step 8. If not, proceed to Step 5.

Step 5: For each individual in the current population, select a crossover strategy and perform mutation operation based on the evolutionary generation to generate new solutions. Calculate the fitness of the new solutions and update the reference points.

Step 6: Perform a variable neighborhood search strategy, calculate the fitness of the neighborhood solutions, and replace the original chromosome with a superior solution.

Step 7: Evaluate the updated individuals, update the neighborhood and reference points using non-dominated sorting, then return to Step 4.

Step 8: The algorithm terminates, and the non-dominated solution set is outputted.

G. COMPLEXITY ANALYSIS

Suppose there are L total operations for given job, and the algorithm has a maximum iteration number of MAX_{GEN}, a population size of N, and a neighborhood size of T. The iteration process mainly consists of four parts:

The first part involves calculating the fitness of the new solutions generated through crossover and mutation and updating the reference points, with a time complexity of $O(MAX_{GEN}N + MAX_{GEN}NZ)$, where Z is the number of reference points.

The second part involves updating the population with a time complexity of $O(MAX_{GEN}NL)$.

The third part involves variable neighborhood search with a time complexity of $O(MAX_{GEN}LK)$, where K is the maximum iteration number for variable neighborhood search.

The fourth part involves updating the non-dominated solution set. In each iteration of the first generation, dominated solutions are removed from the solution set, with a time complexity of $O(MAX_{GEN}E_pN)$, where E_p is the size of the non-dominated solution set.

Based on this, the overall time complexity of the algorithm can be determined to be $T(n) = MAX_GEN(N + NZ + NL + LK + E_pN)$, indicating that the main parameters affecting algorithm performance are the population size and total number of operations.

V. EXPERIMENTAL RESULTS AND COMPARISONS

A. EXPERIMENTAL DESIGN

The test dataset used in this study was obtained from [17] and [18], comprising a total of 10 questions. The job processing range was between 10-20, the machine range was between 6-20, and the total number of operations ranged from 50-255.

The performance of the algorithm was evaluated using the ratio and distance metrics proposed in [19] and [20].

(1)The ratio metric ξ_a is defined as the proportion of non-dominated solutions provided by the non-dominated solution set S_a obtained by algorithm a, in the entire reference set S^*, as below:

$$\xi_a = \frac{|S_r \cap S^*|}{|S_r|} \tag{14}$$

Here, S^* represents the non-dominated solution set obtained by merging the non-dominated solution sets of all algorithms. Therefore, the larger the value of ξ_a, the more non-dominated solutions the algorithm can obtain.

TABLE 1. Parameter level.

parameters	values			
	1	2	3	4
T	5	10	15	20
N_P	50	100	150	200
MAX_{GEN}	50	100	150	200

(2) Distance metric IGD_a: refers to the distance between the elements in the non-dominated solution set S_a obtained by algorithm a and the reference set S^*. It can be calculated using the following formula:

$$IGD_a = \frac{1}{|S^*|} \sum_{Y \in S^*} min\{d_{xy} | X \in S_a\} \tag{15}$$

where

$$d_{xy} = [(F_1'(X) - F_1'(Y))^2 + \ldots + (F_D'(X) - F_D'(Y))^2]^(1/2) \tag{16}$$

where D represents the number of targets. To account for the disparate magnitudes of maximum completion time, maximum machine load, and total load, we normalize F_z into F_z' prior to calculation. The normalization formula is as follows

$$F_z = \frac{F_z - F_{zmin}}{F_{zmax} - F_{zmin}} \tag{17}$$

where F_{zmin} and F_{zmax} represent the maximum and minimum values of the z-th objective among all non-dominated solutions in the reference set S^*, respectively. It is evident that the smaller the value of IGD_a, the better the algorithm's performance.

B. PARAMETERS SETTING

The key parameters of IMOEA/D include population size N_P, number of generations MAX_{GEN}, and neighborhood size T. In this section, an experimental design method is used to set these three parameters, by constructing a three-factor four-level orthogonal table $L_{16}(4^3)$. (1)

Table.(2) exhibits the outcomes derived from performing 10 independent iterations of algorithms M1 and M6 for each parameter configuration, with X serving as the performance evaluation metric for data acquisition. The values presented in table.(2) correspond to the average results obtained from the aforementioned runs. Table.(3) presents the response values of the various parameters, indicating that the range of parameter MAX_{GEN} is the largest among all, thereby exerting the most significant impact on algorithm performance, followed by N_P, while parameter T has the least influence. Based on a comprehensive analysis of the outcomes, the optimal parameter configuration for IMOEA/D is determined to be: population size $N_P = 200$, evolution generation $MAX_{GEN} = 200$, and neighborhood size $T = 15$.

TABLE 2. Orthogonal tables and distance indices.

parameter combination	level			IGD_a
	N_P	MAX_{GEN}	T	
1	1	1	1	0.0687
2	1	2	2	0.0568
3	1	3	3	0.0473
4	1	4	4	0.0494
5	2	1	2	0.0729
6	2	2	1	0.0798
7	2	3	4	0.0547
8	2	4	3	0.0493
9	3	1	3	0.0603
10	3	2	4	0.0504
11	3	3	1	0.0607
12	3	4	2	0.0503
13	4	1	4	0.0549
14	4	2	3	0.0479
15	4	3	2	0.0486
16	4	4	1	0.0472

TABLE 3. The response value of each parameter.

level	N_P	MAX_{GEN}	T
1	0.0556	0.0642	0.0641
2	0.0642	0.0587	0.0571
3	0.0554	0.0528	**0.0512**
4	**0.0496**	**0.0491**	0.0523
range	0.0146	0.0151	0.0129
grade	2	1	3

C. COMPARISON AND DISCUSSION

In order to further substantiate the convergence and distribution performance of the improved MOEA/D, the following algorithms were employed as comparative benchmarks: an evolutionary taboo search algorithm (EATS) proposed in [21], and an improved hybrid particle swarm optimization algorithm (IH-PSO) proposed in [22]. The effectiveness of the proposed algorithm was verified through simulation experiments. Reference [21] focuses on the multi-objective fuzzy flexible job shop problem, with a main emphasis on the satisfaction level of due dates, while [22] addresses the multi-objective flexible job shop scheduling problem. Both algorithms' optimization objectives were modified to align with the optimization goals of this paper, resulting in an improved evolutionary taboo search algorithm (EATS) and a hybrid particle swarm optimization algorithm (IH-PSO) that share the same optimization objectives as the proposed algorithm in this study. All algorithms were set with the following parameters for comparison: population size was 200 and the maximum number of iterations was 200. Each comparative algorithm was independently run 30 times on the test set to obtain the average IGD_a and ξ. Tables (4) and (5) show the average values of x and y for the 10 groups obtained by the three algorithms. The best performing metric is indicated in bold and italic.

According to the results shown in (4) and (5), the IMOEA/D algorithm achieved the best performance in all test cases. The average values of the three algorithms for this

TABLE 4. ξ values obtained by IMOEAD/D, EATS and IH-PSO.

instance	IMOEA/D	EATS	IH-PSO
M1(50×6)	**0.4583**	0.3750	0.1667
M2(50×10)	**0.4063**	0.3125	0.2813
M3(100×8)	**0.4655**	0.2931	0.2413
M4(150×8)	**0.5263**	0.2456	0.2281
M5(100×10)	**0.3809**	0.3473	0.2787
M6(120×15)	**0.5128**	0.1025	0.3846
M7(150×10)	**0.4285**	0.3137	0.2577
M8(200×10)	**0.5517**	0.2413	0.2069
M9(240×10)	**0.4957**	0.1453	0.3590
M10(255×20)	**0.4193**	0.1935	0.3871

TABLE 5. *IGD* values obtained by IMOEAD/D, EATS and IH-PSO.

instance	IMOEA/D	EATS	IH-PSO
M1	**0.0061**	0.0103	0.0121
M2	**0.0039**	0.0045	0.0046
M3	**0.0042**	0.0168	0.0210
M4	**0.0033**	0.0051	0.0138
M5	0.0079	**0.0072**	0.0159
M6	**0.0017**	0.0045	0.0142
M7	**0.0096**	0.0149	0.0164
M8	0.0099	0.0115	**0.0089**
M9	**0.0034**	0.0132	0.0060
M10	**0.0066**	0.0107	0.0092

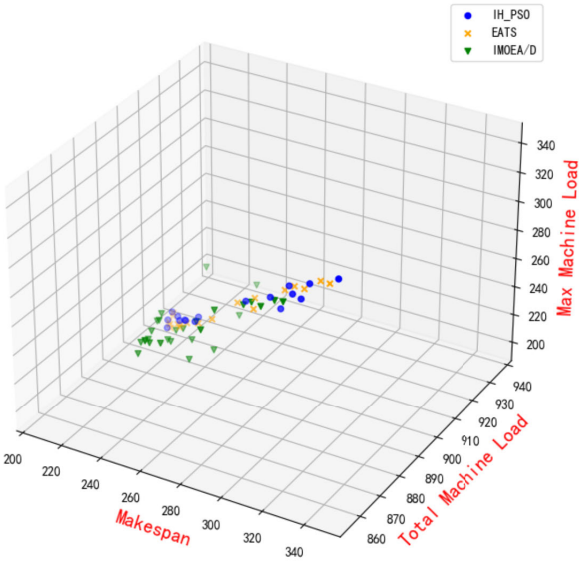


FIGURE 2. Solution space of instance M3.

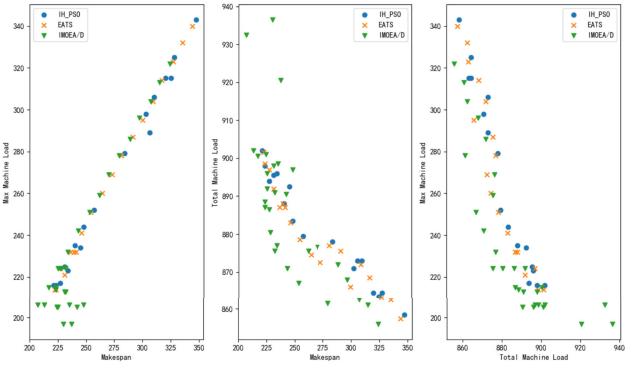


FIGURE 3. Three views of solution space of instance M3.

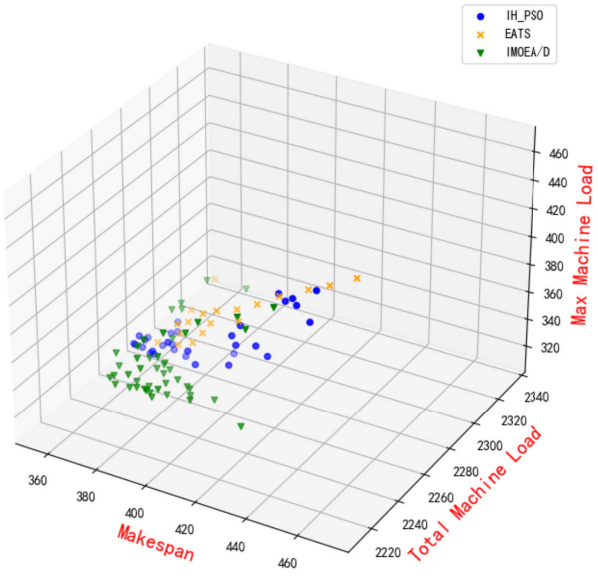


FIGURE 4. Solution space of instance M9.

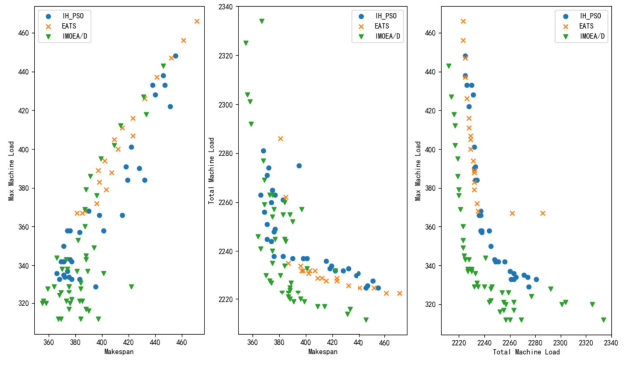


FIGURE 5. Three views of solution space of instance M9.

metric in the 10 groups of test cases were 0.4645, 0.2569, and 0.2791, indicating that the IMOEA/D algorithm has a certain advantage in terms of distribution compared to the other two algorithms. Further comparison between IMOEA/D algorithm and EATS algorithm shows that the number of non-dominated solutions obtained by both algorithms is close when the problem size is small (such as problem instances M1, M2, and M3). However, as the problem size increases (such as problem instances M9 and M10), IMOEA/D algorithm has a significant advantage over EATS algorithm in terms of the number of non-dominated solutions obtained. Compared with the IH-PSO algorithm, the IMOEA/D algorithm has a significant advantage in small-scale examples (such as M1, M3, M4). It can be seen that regardless of small or large-scale examples, IMOEA/D has good performance

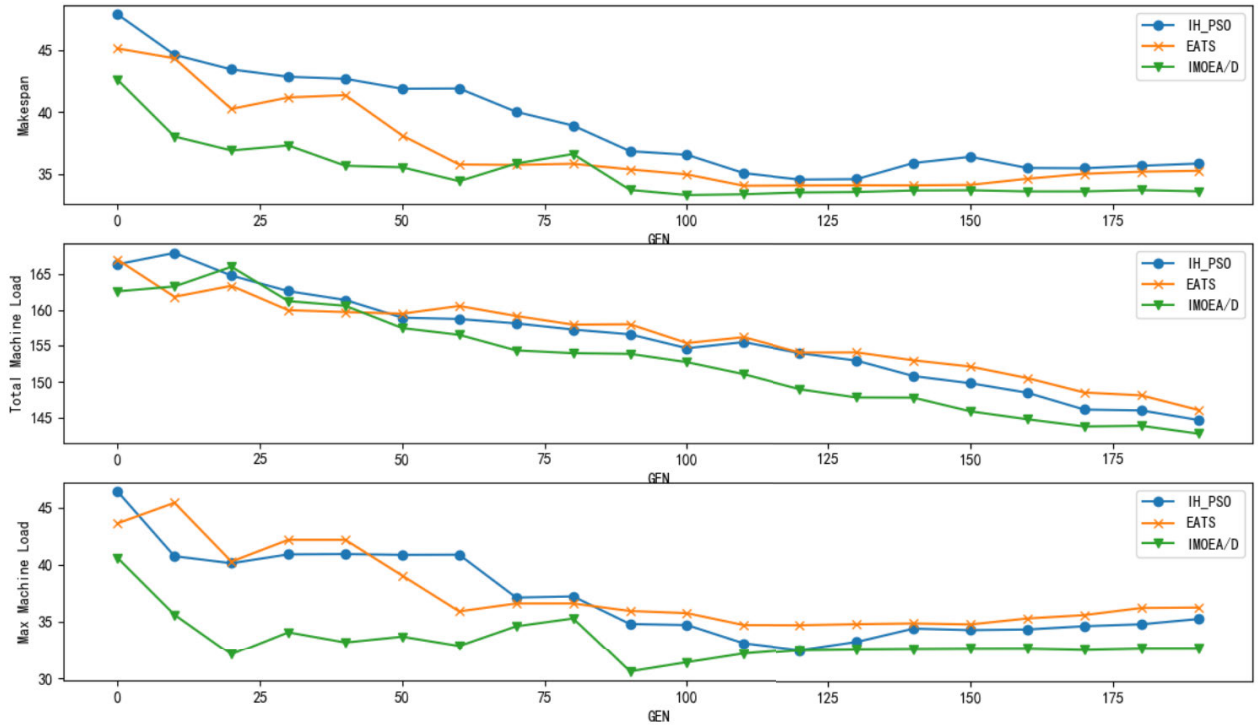


FIGURE 6. Convergence curve of instance 2.

in the distribution of non-dominated solutions. The superior performance of IMOEA/D in terms of distribution diversity may be attributed to several factors. Firstly, the algorithm utilizes multiple strategies in the crossover process to update the population, ensuring individual diversity and balancing global and local search capabilities. Secondly, the evolution of the population is adjusted based on the success rate of the evolution, resulting in improved population diversity. Finally, the algorithm employs a neighborhood search strategy to further enhance local search capabilities and improve the uniformity of the distribution of the non-dominated solution set.

The results in Table (5) show that IMOEA/D algorithm obtained the best performance in 8 instances, while EATS algorithm obtained the best performance in one test case, and IH-PSO algorithm obtained the best performance in one instance as well. Compared with the EATS algorithm, IMOEA/D algorithm significantly outperformed it in 4 instances (M1, M3, M6, and M9) and showed no significant difference in the other 6 instances. Compared with the IH-PSO algorithm, IMOEA/D algorithm significantly outperformed it in 5 instances (M1, M2, M4, M5, and M6) and showed no significant difference in the other 5 instances. Moreover, for instances 9 and 10, IMOEA/D algorithm obtained the smallest indicators, indicating that IMOEA/D has good convergence performance in solving large-scale problems. For small-scale problems, such as instances 1, 2, and 4, IMOEA/D also has certain advantages. Although IMOEA/D did not achieve the best performance in instances 5 and 8, there was no significant difference compared to the other two algorithms. The average value of the indicator for these three algorithms in all 10 instances were 0.0056, 0.0098, and 0.0122, respectively, which further indicates that IMOEA/D is competitive in terms of convergence.

There are several possible reasons why IMOEA/D performs well in terms of convergence. Firstly, the multiple crossover strategies used during the evolutionary process, particularly strategies b and c, ensure a high inheritance rate for the optimal individuals, thereby preserving the superior genetic information within the population and maintaining the quality of the solution set. Secondly, the four neighborhood search strategies utilized by the algorithm provide a means for further exploring the search space and discovering better solutions. The use of weight vectors for single-objective evaluation and greedy search also enhances the quality of the non-dominated solution set in multi-objective problems. Lastly, the use of an excellent initial population can increase the efficiency of the algorithm and contribute to improving the final solution set.

As shown in Figures (2), (3), (4) and (5), the distribution and three-dimensional views of the obtained non-dominated solutions by the three algorithms are presented for test instances M3 and M9, respectively. For case M3, IMOEA/D obtains a non-dominated solution set with good distribution and convergence. The non-dominated solution set obtained by EATS exhibits acceptable distribution and convergence, while IH-PSO performs relatively poorly in convergence but has good distribution of the obtained non-dominated

solution set. EATS algorithm has a slight advantage over IH-PSO in both distribution and convergence of the obtained non-dominated solution set. For case M9, IMOEA/D has a certain advantage in both distribution and quality of the obtained non-dominated solution set compared to the other two algorithms. The non-dominated solution set obtained by EATS has good distribution but relatively poor convergence, while IH-PSO exhibits acceptable distribution and convergence in the obtained non-dominated solution set. Overall, the three optimization objectives interact with each other, and IMOEA/D algorithm has advantages in both distribution and convergence of the obtained solutions.

To further compare the performance of the algorithms, Figure (6) shows the convergence curves of the three algorithms for Instance 2.

The convergence curves of makespan, total machine load, and maximum machine load with respect to the evolution generations for case 2 are presented in Figure (6), which further demonstrates the competitiveness of IMOEA/D algorithm in terms of convergence performance. IMOEA/D algorithm converges faster than the other two algorithms in terms of maximum machine load and completion time, with significant advantages in completion time and maximum machine load. The completion time converges to the optimal solution and becomes stable around 100 generations, while the maximum load converges to the optimal solution around 120 generations. It can be seen from the above figure that the initial values of IMOEA/D are lower than those of the other two algorithms, indicating that the population initialization strategy proposed in this paper can effectively improve the quality of the initial population and has a certain effect on improving the convergence of the algorithm. In summary, the IMOEA/D algorithm demonstrates good performance on the job shop scheduling problem studied in this paper.

VI. CONCLUSION AND FUTURE WORKS

This study addresses the MOFFJSP by formulating a mathematical model that optimizes for maximum completion time, maximum machine load, and total machine load. To solve this problem, an improved decomposition-based multi-objective evolutionary algorithm is proposed. The algorithm proposed in this study employs a dual-level encoding scheme. It generates the initial population using a minimum processing time and workload strategy, as well as a non-dominated solution priority selection mechanism. To ensure diversity in the search process, three strategies are introduced in the crossover operation. Moreover, a variable neighborhood search algorithm is integrated into the search process, which comprises four distinct neighborhood structures. The experimental results of algorithm simulation testing on 10 standard datasets demonstrate the effectiveness of IMOEA/D in solving the MOFFJSP proposed in this paper.

The future research directions are as follows: (1) Extend the scheduling problem addressed in this paper to more complex production environments, such as dynamic production workshops and distributed production workshops. (2) Combine the improved decomposition-based multi-objective evolutionary algorithm with other heuristic algorithms to leverage their respective strengths and achieve better algorithm performance. (3) Consider adding intelligent agents in the algorithm and using multi-agent search to obtain better scheduling solutions.

ACKNOWLEDGMENT

The authors would like to thank the editors and reviewers for their helpful comments and suggestions to improve the quality of this article.

REFERENCES

[1] K. G. Devi, R. S. Mishra, and A. K. Madan, "A dynamic adaptive firefly algorithm for flexible job shop scheduling," *Intell. Autom. Soft Comput.*, vol. 31, no. 1, pp. 429–448, 2022.

[2] X. Wen, K. Wang, and H. Li, "A two-stage solution method based on NSGA-II for green multi-objective integrated process planning and scheduling in a battery packaging machinery workshop," *Swarm Evol. Comput.*, vol. 61, Mar. 2021, Art. no. 100820.

[3] A. Baykasoğlu, F. S. Madenoğlu, and A. Hamzadayı, "Greedy randomized adaptive search for dynamic flexible job-shop scheduling," *J. Manuf. Syst.*, vol. 56, pp. 425–451, Jul. 2020.

[4] K. Geng, C. Ye, Z. H. Dai, and L. Liu, "Bi-objective re-entrant hybrid flow shop scheduling considering energy consumption cost under time-of-use electricity tariffs," *Complexity*, vol. 2020, pp. 1–17, Feb. 2020.

[5] X. Jiang, Z. Tian, W. Liu, Y. Suo, K. Chen, X. Xu, and Z. Li, "Energy-efficient scheduling of flexible job shops with complex processes: A case study for the aerospace industry complex components in China," *J. Ind. Inf. Integr.*, vol. 27, May 2022, Art. no. 100293.

[6] K. Gao, F. Yang, M. Zhou, Q. Pan, and P. N. Suganthan, "Flexible job-shop rescheduling for new job insertion by using discrete Jaya algorithm," *IEEE Trans. Cybern.*, vol. 49, no. 5, pp. 1944–1955, May 2019, doi: 10.1109/TCYB.2018.2817240.

[7] A. Gnanavelbabu, R. H. Caldeira, and T. Vaidyanathan, "A simulation-based modified backtracking search algorithm for multi-objective stochastic flexible job shop scheduling problem with worker flexibility," *Appl. Soft Comput.*, vol. 113, Dec. 2021, Art. no. 107960.

[8] Z. Zhu and X. Zhou, "A multi-objective multi-micro-swarm leadership hierarchy-based optimizer for uncertain flexible job shop scheduling problem with job precedence constraints," *Expert Syst. Appl.*, vol. 182, Nov. 2021, Art. no. 115214.

[9] D. Lei, "Co-evolutionary genetic algorithm for fuzzy flexible job shop scheduling," *Appl. Soft Comput.*, vol. 12, no. 8, pp. 2237–2245, Aug. 2012.

[10] B. Liu, Y. Fan, and Y. Liu, "A fast estimation of distribution algorithm for dynamic fuzzy flexible job-shop scheduling problem," *Comput. Ind. Eng.*, vol. 87, pp. 193–201, Sep. 2015.

[11] 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, Oct. 2022, Art. no. 117380.

[12] D. Yang, X. Zhou, Z. Yang, Q. Jiang, and W. Feng, "Multi-objective optimization model for flexible job shop scheduling problem considering transportation constraints: A comparative study," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2020, pp. 1–8, doi: 10.1109/CEC48606.2020.9185653.

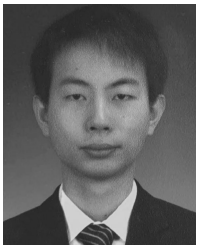
[13] E.-D. Jiang, L. Wang, and Z.-P. Peng, "Solving energy-efficient distributed job shop scheduling via multi-objective evolutionary algorithm with decomposition," *Swarm Evol. Comput.*, vol. 58, Nov. 2020, Art. no. 100745.

[14] B. Zhou and X. Liao, "Particle filter and levy flight-based decomposed multi-objective evolution hybridized particle swarm for flexible job shop greening scheduling with crane transportation," *Appl. Soft Comput.*, vol. 91, Jun. 2020, Art. no. 106217.

[15] L. Xixing and L. Yi, "Approach of solving dual resource constrained multi-objective flexible job shop scheduling problem based on MOEA/D," *Int. J. Online Eng. (iJOE)*, vol. 14, no. 7, p. 75, Jul. 2018.

[16] K. Z. Gao, P. N. Suganthan, Q. K. Pan, T. J. Chua, C. S. Chong, and T. X. Cai, "An improved artificial bee colony algorithm for flexible job-shop scheduling problem with fuzzy processing time," *Expert Syst. Appl.*, vol. 65, pp. 52–67, Dec. 2016.

[17] F. Jolai, H. Asefi, and M. Rabiee, "Bi-objective simulated annealing approaches for no-wait two-stage flexible flow shop scheduling problem," *Scientia Iranica*, vol. 20, pp. 861–872, Jun. 2013.
[18] J. J. Palacios, I. González-Rodríguez, C. R. Vela, and J. Puente, "Coevolutionary makespan optimisation through different ranking methods for the fuzzy flexible job shop," *Fuzzy Sets Syst.*, vol. 278, pp. 81–97, Nov. 2015.
[19] D. Lei, "Pareto archive particle swarm optimization for multi-objective fuzzy job shop scheduling problems," *Int. J. Adv. Manuf. Technol.*, vol. 37, pp. 157–165, Apr. 2008.
[20] J. Knowles and D. Corne, "On metrics for comparing nondominated sets," in *Proc. Congr. Evol. Comput. (CEC)*, May 2002, pp. 711–716, doi: 10.1109/CEC.2002.1007013.
[21] C. R. Vela, S. Afsar, J. J. Palacios, I. González-Rodríguez, and J. Puente, "Evolutionary Tabu search for flexible due-date satisfaction in fuzzy job shop scheduling," *Comput. Oper. Res.*, vol. 119, Jul. 2020, Art. no. 104931.
[22] Y. Zhang, H. Zhu, and D. Tang, "An improved hybrid particle swarm optimization for multi-objective flexible job-shop scheduling problem," *Kybernetes*, vol. 49, no. 12, pp. 2873–2892, Dec. 2019.



ZHENGGANG LIU was born in Jilin, China, in 1993. He received the B.S. degree in information engineering from Tianjin University, in 2016. He is currently pursuing the M.S. degree in electronic information with Beijing Information Science and Technology University. His research interests include system optimization theory and manufacturing informatization.

XU LIANG received the Graduate degree from the Department of Computer Science, Sichuan University, Chengdu, China, in 1997, the M.S. degree in traffic engineering from the Dalian Railway Institute, in 2001, and the Ph.D. degree in mechanical engineering from Dalian Jiaotong University, in 2005. She is currently a Professor and a Doctoral Supervisor with Beijing Information Science and Technology University. Her research interests include system optimization theory and applications, and manufacturing informatization.

LINGYAN HOU received the Graduate degree from the Department of Automation, Hunan University, Changsha, China, in 1985, and the M.S. degree in signal aircraft instrumentation and testing from Beihang University, in 1991. She is currently a Professor with Beijing Information Science and Technology University. Her research interests include pattern recognition, intelligent storage, and logistics information processing.

DALI YANG received the Graduate degree from the Department of Mathematics, Inner Mongolia University, China, in 1984, the M.S. degree in mathematics from Tsinghua University, in 1995, and the Ph.D. degree, in 2003. He is currently an Associate Professor with Beijing Information Science and Technology University. His research interests include speech recognition and image recognition.

QIANG TONG received the Graduate degree from the Department of Automation, Northeastern University, China, in 2003, the M.S. degree in computer science from Tsinghua University, in 2007, and the Ph.D. degree, in 2012. He is currently a Lecturer with Beijing Information Science and Technology University. His research interests include computer vision and pattern recognition.