Industrial Engineering and Innovation Management (2025)
Clausius Scientific Press, Canada

DOI: 10.23977/ieim.2025.080114
ISSN 2522-6924 Vol. 8 Num. 1

A Graph Embedding Algorithm Based on Reinforcement Learning for Solving Fuzzy Multi-Objective Flexible Job Shop Scheduling Problem

Weiyuan Wang[1,*], Fuqing Zhao[1]

[1]*School of Computer and Communication, Lanzhou University of Technology, Lanzhou, China*
**Corresponding author: wwy19396720686@163.com*

Keywords: Multi-objective flexible job-shop scheduling problem, Graph embedding, Fuzzy disjunctive graph, Reinforcement learning

Abstract: This study delves into the multi-objective flexible job-shop scheduling problem with fuzzy time (MOFFJSP), addressing complex and dynamic production environments, with the goal of minimizing the maximum completion time and total machine workload (TMW). A graph embedding algorithm based on reinforcement learning (GEARL) is proposed in this article, consisting of four modules: population initialization, fuzzy disjunctive graph, policy model, and solution set optimization processing. Diverse initial populations are constructed using different rules tailored to the problem. A fuzzy disjunctive graph is designed to transform individual data information into graph information. Graph embedding technology is utilized to extract individual feature information and generate corresponding optimization strategies through the policy model. Solution set optimization processing involves performing non-dominated sorting on the entire population to filter out advantageous individuals to guide subsequent optimization directions. Experimental results demonstrate that the GEARL algorithm exhibits significant advantages in solving the MOFFJSP.

1. Introduction

As manufacturing enterprises continue to expand, the urgent need to reduce machine workload while improving production efficiency has prompted traditional manufacturing enterprises to accelerate transformation and upgrading [1]. Compared to traditional workshop scheduling issues, the flexible job shop scheduling problem (FJSP) is more complex, requiring not only the rational arrangement of processing sequences but also the precise allocation of machine resources for each operation [2]. Therefore, FJSP is fundamentally classified as a classic NP-hard problem [3].

In recent years, researchers have conducted in-depth studies on the FJSP. For single-objective FJSP, existing literature [4-6] has delved into the subject, with goals typically focusing on minimizing the maximum completion time. However, as manufacturing enterprises continue to grow in scale, decision-makers need to balance multiple objectives. Consequently, the multi-objective flexible job shop scheduling problem (MOFJSP) has gradually become a research hotspot.

Zhang et al. [7] studied the distributed MOFJSP, considering both total completion time (TET) and total energy consumption (TEC), and proposed a super-heuristic algorithm based on Q-learning. Luo et al. [8] designed a knowledge-guided two-stage memetic algorithm to address multi-objective energy-saving FJSP, which also considers TET and TEC. Zhu et al. [9] proposed an improved memetic algorithm for distributed FJSP in production environments, aiming to optimize TET and TEC. Li et al. [10] focused on energy-saving FJSP under finite worker conditions and proposed a multi-objective evolutionary algorithm based on reference fuzzy correlation entropy. This article, when studying MOFJSP, considers both TET and total machine workload (TMW) as objectives and constructs the corresponding mathematical model.

In actual production workshop scheduling, numerous uncertainties such as raw material shortages, insufficient power supply, machine failures, lead to uncertainty in job processing times in FJSP [11]. However, most existing theoretical models typically assume fixed values for processing times, which significantly differ from real production environments. Therefore, to better adapt to uncertain environments, this article introduces fuzzy numbers into MOFJSP, expanding it to fuzzy MOFJSP. Chen et al. [12] proposed a multi-objective hybrid immune algorithm for solving fuzzy FJSP with variable processing speeds, considering fuzzy TET and fuzzy TEC. Abdel-Basset et al. [13] introduced a hybrid fuzzy flexible job shop scheduling algorithm (HFFSA) for fuzzy FJSP. Triangular fuzzy numbers (TFN) are introduced into MOFJSP in this article, as this model more accurately reflects real production situations.

In the past, evolutionary algorithms such as ant colony optimization (ACO) [14], artificial bee colony (ABC) [15], memetic algorithm (MA) [16], etc., have been widely adopted to find optimal solutions within a reasonable time frame. However, these algorithms often face challenges such as convergence difficulties, cumbersome parameter tuning, and computational complexity when dealing with increasingly complex models. With the rapid development of artificial intelligence technology, introducing reinforcement learning (RL), neural networks, etc., into scheduling problems can significantly improve solution accuracy. Hu et al. [17] combined mathematical programming methods, RL, and metaheuristic algorithms to propose a learning-oriented multi-objective artificial bee colony algorithm to solve dynamic FJSP. Akram et al. [18] proposed a multi-objective black widow spider algorithm to address multi-objective dynamic FJSP. Chen et al. [19] designed a reinforcement learning algorithm integrated with long short-term Memory (LSTM) networks to solve dynamic FJSP. Zhang et al. [20] optimized FJSP by introducing attention mechanisms and RL frameworks for intelligent agents. Park et al. [21] combined graph neural networks (GNN) and RL, using disjunctive graphs to extract common JSP indicators. Based on this, this article introduces GNN and RL to solve fuzzy FJSP.

In the research background mentioned above, this article proposes a fuzzy time-aware FJSP at the problem level, considering both TET and TMW as objectives. A graph embedding algorithm based on reinforcement learning (GEARL) is proposed to solve this problem. The rest of this article is organized as follows: The second part introduces the basic theory of triangular fuzzy numbers and defines the MOFFJSP model. The third part presents the GEARL. The fourth part discusses the experimental results. The fifth part concludes and outlines future work.

2. Definition of the problem

2.1. The fundamental theory of TFN

Fuzzy set theory is a generalization and extension of classical set theory, with its core being the introduction of membership functions to quantify the degree of an element's membership in a set. A fuzzy set $\tilde{A}$ is defined by formula (1). Here, x is any element in the fuzzy set $\tilde{A}$, $u_{\tilde{A}}(x)$ serves as the

membership function quantifying the possibility of x belonging to set $\tilde{A}$, and X is the fixed domain of the fuzzy variable.

$$\tilde{A} = \{x, u_{\tilde{A}}(x) | \forall x \in X\}, 0 \leq u_{\tilde{A}}(x) \leq 1 \tag{1}$$

Classical sets are deterministic sets, where $u_{\tilde{A}}(x) = 1$ implies that the element x completely belongs to the set. Formula (2) represents the classical set.

$$F = \{x, u_{\tilde{A}}(x) = 1 | \forall x \in X\} \tag{2}$$

$$f(x) = \begin{cases} 0, & x \leq t_1 \\ \dfrac{x - t_1}{t_2 - t_1}, & t_1 < x \leq t_2 \\ \dfrac{t_3 - x}{t_3 - t_2}, & t_2 < x < t_3 \\ 0, & x \geq t_3 \end{cases} \tag{3}$$

In addressing practical scheduling problems, due to the uncertainty of job processing times, TFN are introduced. As shown in Figure 1, t_1 represents the earliest processing time, t_2 represents the most likely processing time, t_3 represents the latest processing time. TFN are typically represented as a triplet $TFN = (t_1, t_2, t_3)$, and the triangular membership function is defined as shown in formula (3).

Given two triangular fuzzy numbers $\tilde{A} = (a_1, a_2, a_3)$ and $\tilde{B} = (b_1, b_2, b_3)$, three operations are introduced: addition, comparison, and maximum value selection.

(1) Addition operation

$$\tilde{A} + \tilde{B} = (a_1 + b_1, a_2 + b_2, a_3 + b_3) \tag{4}$$

(2) Comparison operation

Case 1: $y_1(\tilde{x}) = (x_1 + 2x_2 + x_3)/4$, If $y_1(\tilde{A}) > y_1(\tilde{B})$, then $\tilde{A} > \tilde{B}$, otherwise $\tilde{A} < \tilde{B}$.

Case 2: $y_2(\tilde{x}) = x_2$, when $y_1(\tilde{A}) = y_1(\tilde{B})$, If $y_2(\tilde{A}) > y_2(\tilde{B})$, then $\tilde{A} > \tilde{B}$, otherwise $\tilde{A} < \tilde{B}$.

Case 3: $y_3(\tilde{x}) = a_3 - a_1$, when $y_2(\tilde{A}) = y_2(\tilde{B})$, $y_3(\tilde{A}) > y_3(\tilde{B})$, then $\tilde{A} > \tilde{B}$, otherwise $\tilde{A} < \tilde{B}$.

(3) Maximum value selection operation

If $\tilde{A} > \tilde{B}$, then $\tilde{A} \vee \tilde{B} = \tilde{A}$; otherwise, $\tilde{A} \vee \tilde{B} = \tilde{B}$.

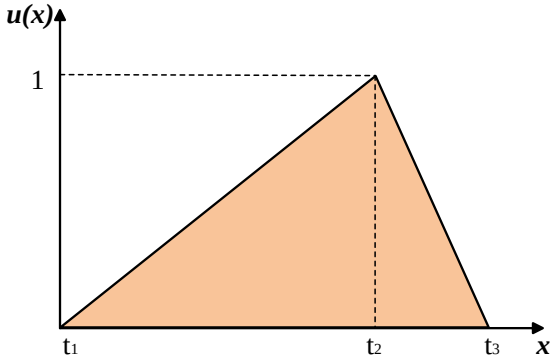

Figure 1: TFN

2.2. The notations of MOFFJSP

The notations of MOT2FJSP are described as follows.

Notations	Description
n	Total number of jobs.
m	Total number of machines.
i	Index of machine.
j, k	Index of job.
h_j	Total number of operations of job j.
l	Index of operation.
$\tilde{p}_{ijh}$	The fuzzy processing time of the h operation of job j on machine i.
$\tilde{s}_{jh}$	The fuzzy start time of the h operation of job j.
$\tilde{c}_{jh}$	The fuzzy completion time of the h operation of job j.
L	A sufficiently large positive number.
$\tilde{C}_{max}$	The fuzzy maximum completion time of a schedule.
W_i	The total workload of machine i.
J	Collection of jobs, $J = \{1,2,3,\ldots,n\}$.
M	Collection of machines, $M = \{1,2,3,\ldots,m\}$.
H_j	Collection of operations for job j, $H_j = \{1,2,3,\ldots,h_j\}$.
$x_{ijh} = \begin{cases} 1, \\ 0, \end{cases}$	If operation O_{jh} selects machine i. Otherwise.
$y_{ijhkl} = \begin{cases} 1, \\ 0, \end{cases}$	If O_{ijh} processed before O_{ikl}. Otherwise.

2.3. The MILP of MOFFJSP

The MILP (Mixed Integer Linear Programming) model is an abstract method, the construction of which is aimed at simplifying and handling complex problems [22]. The MILP description of MOFFJSP is given as follows:

$$\text{Objective function: } min(\tilde{C}_{max},\ W_i) \tag{5}$$

$$\text{s.t. } \tilde{s}_{jh} + x_{ijh} \times \tilde{p}_{ijh} \leq \tilde{c}_{jh}, (\forall i \in M, j \in J, h \in H_j) \tag{6}$$

$$\tilde{c}_{jh} \leq \tilde{s}_{j(h+1)}, (\forall j \in J, h \in H_{j-1}) \tag{7}$$

$$\tilde{c}_{jh_j} \leq \tilde{C}_{max}, (\forall j \in J) \tag{8}$$

$$\tilde{s}_{jh} + \tilde{p}_{ijh} \leq \tilde{s}_{kl} + L \times (1 - y_{ijhkl}), (\forall i \in M, j \in J, k \in J, h \in H_j, l \in H_k) \tag{9}$$

$$\tilde{c}_{jh} \leq \tilde{s}_{j(h+1)} + L \times (1 - y_{iklj(h+1)}), (\forall i \in M, j \in J, k \in J, h \in H_{j-1}, l \in H_k) \tag{10}$$

$$\sum_{i=1}^{m_{jh}} x_{ijh} = 1, (\forall h \in H_j, j \in J) \tag{11}$$

$$\sum_{j=1}^{n} \sum_{h=1}^{h_j} y_{ijhkl} = x_{ikl}, (\forall i \in M, k \in J, l \in H_k) \tag{12}$$

$$\sum_{k=1}^{n}\sum_{l=1}^{h_k} y_{ijhkl} = x_{ijh}, (\forall i \in M, j \in J, h \in H_k) \tag{13}$$

$$\tilde{s}_{jh} \geq 0, \tilde{c}_{jh} \geq 0, (\forall j \in J, h \in H_j) \tag{14}$$

$$W_i = \tilde{p}_{ijh} \times x_{ijh}, (\forall i \in M, j \in J, h \in H_j) \tag{15}$$

The optimization objective of MOFFJSP is defined by function (5). The sequence of operations for each job is determined according to formulas (6) and (7). Formula (8) sets an upper limit on the completion time for each job to ensure that the completion time of all jobs does not exceed the total completion time. Formulas (9) and (10) specify that only one operation is processed on the same machine at the same time. Formula (11) represents machine constraints, stating that a particular operation is processed by only one machine at the same time, and formulas (12) and (13) indicate the existence of cyclic operations on each machine. Formula (14) stipulates that all parameter variables must be positive. Formula (15) outlines the calculation method for machine workload.

3. Algorithm Section

3.1. The overall framework of the GEARL

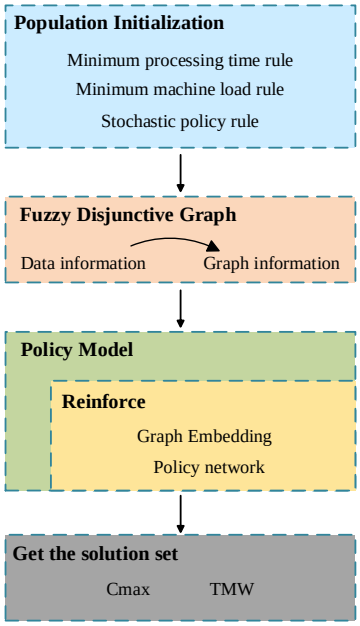

Figure 2: The overall framework of the GEARL

The overall structure of the algorithm is illustrated in Figure 2, consisting of four modules: population initialization, fuzzy disjunctive graph information transformation, strategy model construction, and solution set optimization processing. In the population initialization phase, three different rules are designed to enhance the diversity of the population and ensure the high quality of the initial population. The three initialization rules are as follows: the minimum processing time rule, the minimum machine load rule, and the random strategy rule. In the fuzzy disjunctive graph module, the data of each individual in the population is transformed into the form of a disjunctive graph, facilitating the training and learning of the subsequent strategy model. In the strategy model module, graph embedding technology is utilized to extract individual features, which are then input

into a policy network consisting of fully connected layers. This network outputs sequence adjustment strategies for each individual based on their features, such as the job sequence and machine sequence. By comparing the performance of new and old individuals in terms of maximum completion time and total machine load, the values of the corresponding reward functions are adjusted accordingly. The entire model iteratively updates using the Reinforce algorithm in reinforcement learning, ensuring that the policy model guides each individual to choose the optimal evolutionary path through continuous training. In the solution set optimization module, a non-dominated sorting is applied to the updated population to select the optimal solution set. In summary, the GEARL algorithm efficiently addresses MOFFJSP through these four closely collaborating modules.

3.2. Encoding and population initialization

In this paper, two one-dimensional vectors (operate sequence (OS) and a machine sequence (MS)) are used to represent a solution for MOFFJSP. As shown in Figure 3, OS lists the execution order of each operation in all jobs, while MS indicates the corresponding machines selected for each operation. The lengths of OS and MS are equal to the total number of operations in the problem being solved. By sorting based on the positional correspondence of elements in OS and MS, the maximum completion time of this solution can be calculated. Combining MS with its corresponding time information allows for the determination of the total workload of the machines. Ultimately, this solution is parsed and transformed into a specific fuzzy scheduling scheme.

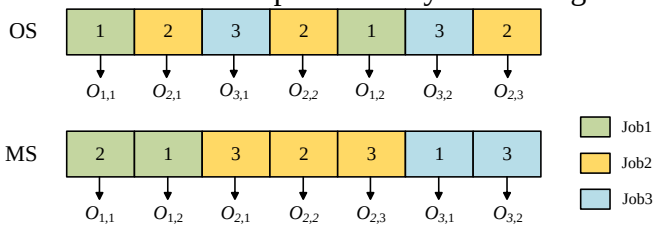

Figure 3: Representation of the solution

To improve the convergence efficiency of the algorithm and reduce the consumption of computing resources, three methods for initializing the population have been designed. The population is evenly divided into three parts, and then constructed according to the following rules in sequence:

(1) Minimum processing time rule: Firstly, OS is randomly initialized, and based on the principle of fuzzy number comparison, the machine with the smallest processing time among the currently available machines is assigned to each operation to minimize the objective of minimizing $\tilde{C}_{max}$.

(2) Minimum machine load rule: OS is randomly initialized, and based on the principle of fuzzy number comparison, the machine with the minimum machine load is selected for each operation allocation, aiming to reduce the total machine load.

(3) Random strategy rule: OS and MS are randomly initialized to increase the randomness and diversity of the population, thereby expanding the search space.

3.3. Fuzzy disjunctive graph

In the fuzzy disjunctive graph module, a fuzzy disjunctive graph was designed for a specific problem, which effectively transforms sequence information into intuitive graphical information [23]. A triplet $G = \{O, C, E\}$ is defined to represent the fuzzy disjunctive graph, as shown in Figure 4. Here, O represents a set of operation nodes, including start and end nodes as well as all intermediate nodes, each node accompanied by corresponding fuzzy time information. C is a set of directed arcs

connected by black arrows, explicitly showing the logical relationships between preceding and succeeding operations within the same job. E represents a set of undirected disjunctive arcs, where arcs of the same color connect operations that need to be performed on the same machine.

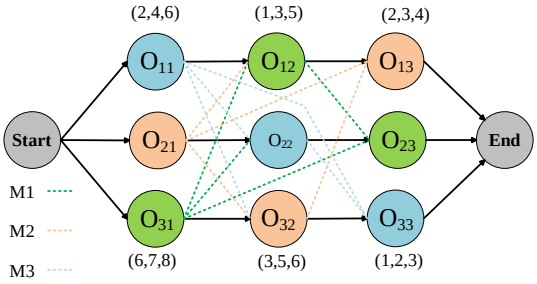

Figure 4: Fuzzy Disjunctive Graph of a 3 × 3 Instance

The process of transforming sequence information into graphical information in this module is as follows: initially, nodes are constructed based on OS, and fuzzy time is normalized by querying the fuzzy time of operations on the corresponding machines, with the normalized results stored as weight information in each node. Subsequently, to distinguish between different processing machines, different colours of dashed lines are used to represent machine information. Simultaneously, black arrows act as connecting arcs, clarifying the sequence of operations. For a complete solution, it is essential to determine the direction of disjunctive arcs and the start time of each operation. By employing an arc-adding strategy, nodes are selected from the arcs of available machines based on the known MS to establish connections. Ultimately, the information from OS and MS is transformed into a fuzzy disjunctive graph. This graph not only consolidates all intricate information but also facilitates subsequent graph embedding operations.

3.4. Policy model

The GEARL algorithm considers the entire population as a synergistic whole, where each individual in the population becomes a part of the reinforcement learning iterative chain. Through the policy model, the algorithm can precisely match the most suitable evolutionary strategy for each individual. Subsequently, the algorithm evaluates key metrics of individuals before and after evolution, including the change in $\tilde{C}_{max}$ value and machine workload, to calculate the cumulative reward function based on this evaluation. Therefore, the magnitude of the cumulative reward function becomes an intuitive scale to measure the quality of the population, aligning well with Markov decision processes. This not only demonstrates the effectiveness of the evolutionary strategy but also provides clear guidance for subsequent network updates.

To gain a deeper understanding of the problem characteristics and optimize the solution strategy, a variant of GNN [24] called graph isomorphic network (GIN) technology is introduced, which extracts vector information from the fuzzy disjunctive graph and generates feature vectors that comprehensively represent the entire graph through multiple processing steps. The detailed process is as follows: $G = (V, E)$ defines the graph, GIN performs k update iterations to compute p - dimensional embeddings for each node $v \in V$, and updates according to equation (16), where $h_v^{(k)}$ is the representation of node v at iteration k, $h_v^{(0)}$ is its original input feature, $MLP_{\theta_k}^{(k)}$ is a multi-layer perceptron with parameters θ_k, followed by batch normalization, ϵ is a learnable parameter, and $N(v)$ is the neighbourhoods of v.

The extracted graph embedding features $h_v^{(k)}$ are then inputted into the policy model, which is responsible for generating a probability distribution $P(a_t)$ to guide the transformation operations of

individuals. The setting of the reward function is closely related to changes in individual target values: if both target values decrease, the reward is 20 to encourage significant optimization; if any target value decreases, the reward is 10 to acknowledge partial optimization; if both target values increase, the reward is 0 as a penalty. The entire model iteratively updates through the Reinforce algorithm, where each population in each iteration represents one round. The improvement in cumulative reward value directly reflects the model's ability to more accurately select the optimal evolutionary strategy based on individual circumstances after training.

$$h_v^{(k)} = MLP_{\theta_k}^{(k)}\left(\left(1+\epsilon^{(k)}\right)\times h_v^{(k-1)} + \sum_{u\in N(v)} h_u^{(k-1)}\right) \tag{16}$$

The 6 strategies output by the policy network are designed based on MOFFJSP and are combined as follows:

OS:

(1): Priority crossover operation on OS, dividing the job set into two parts, exchanging and duplicating OS segments within subsets, and then integrating and restoring them.

(2): Operation exchange on OS, randomly selecting two segments in the OS sequence and swapping their positions.

(3): Operation mutation, randomly selecting a segment in OS and inserting it at any OS position.

MS:

(1): Crossover operation on MS, randomly selecting segments at the same position in two MS and exchanging them.

(2): Mutation operation on MS, randomly selecting a position in MS and replacing it with any machine from the corresponding optional machine set for the operation.

4. Experimental Results and Analysis

In this section, a detailed evaluation of the proposed algorithm's performance is conducted and the experimental results are presented. The experiments are based on testing a subset of standard sets for the fuzzy flexible job shop scheduling problem [25].

GEARL is implemented using the Python programming language. The experimental environment consists of a Windows 11 Pro 64-bit operating system with a 12th Gen Intel(R) Core (TM) i5-12500H CPU @ 3.10 GHz and NVIDIA GeForce RTX 3050 Laptop GPU. Each instance is run 10 times individually, with a termination condition of n*m seconds of CPU time to ensure fairness in the experiments. In subsequent numerical experiments, unless otherwise stated, the above experimental settings remain consistent. MOFFJSP is a multi-objective problem. Three metrics, Hypervolume, Inverted Generational Distance, and C Metric, are used to evaluate the performance of the GEARL algorithm, considering the completeness, convergence, diversity, and dominance relationships of the Pareto front.

4.1. Parameter Settings

The GEARL algorithm mainly includes three key parameters: population size $P \in \{80,90,100,110\}$, policy gradient discount rate $G \in \{0.8,0.9,1\}$, and learning rate $L \in \{2e^{-6}, 2e^{-5}, 2e^{-4}\}$. These parameters are determined based on existing literature and preliminary experimental results. According to orthogonal experimental design, there are a total of $4 \times 3 \times 3 = 36$ different parameter combinations. The optimal parameter combination for the GEARL algorithm is selected based on the 8 different instances in the test set. Each instance is terminated after n×m

seconds of CPU time, run independently 5 times, resulting in a total of 1440 outcomes ($36 \times 8 \times 5$).

Table 1: Presents the ANOVA results for the HV

Source	*Sum Sq.*	*d. f.*	*Mean Sq.*	*F-ratio*	*p-value*
P	0.00044	3	0.00015	14.52	**0.0003**
G	0.00039	2	0.00020	19.10	**0.0002**
L	0.00042	2	0.00021	20.68	**0.0001**
$P * G$	0.00011	6	0.00002	1.73	0.1964
$P * L$	0.00008	6	0.00001	1.30	0.3274
$G * L$	0.00014	4	0.00004	3.50	0.0410
Error	0.00012	12	0.00001		
Total	0.00171	35			

Table 2: Presents the ANOVA results for the IGD

Source	*Sum Sq.*	*d. f.*	*Mean Sq.*	*F-ratio*	*p-value*
P	54.336	3	18.1119	17.27	**0.0001**
G	32.479	2	16.2394	15.48	**0.0005**
L	37.695	2	18.8477	17.97	**0.0002**
$P * G$	11.894	6	1.9824	1.89	0.1639
$P * L$	14.836	6	2.4727	2.36	0.0972
$G * L$	15.385	4	3.8462	3.67	0.0357
Error	12.587	12	1.0489		
Total	179.212	35			

The experiment utilized analysis of variance (ANOVA) technique to assess the experimental results. Before applying ANOVA, three core assumptions were validated: the normality of the data, homogeneity of variance, and independence among observations. Through rigorous testing procedures, it was confirmed that the data collected in this experiment did not exhibit significant biases, meeting the prerequisites for analysis. Subsequently, ANOVA calculations were conducted for the metrics HV and IGD, and the results were summarized in Tables 1 and 2. By analyzing the data in Tables 1 and 2, it is evident that when the $p - value$ is less than the significance level of 0.05, the population size P, policy gradient discount rate G, and learning rate L significantly impact the performance of the GEARL algorithm. This discovery indicates that these three parameters are crucial factors determining the overall efficiency of the GEARL. Furthermore, by comparing the magnitudes of the F-ratio values, it was observed that the learning rate L stands out as the most prominent influencer among all these significant parameters on the performance of the GEARL.

As shown in Figures 5 and 6, main effect plots regarding each participant were constructed based on the HV and IGD metrics. From these plots, it is clearly observed that the performance of all metrics reaches an optimal state when the key parameters are set to $P = 100$, $G = 0.9$, and $L = 2e^{-5}$. This is because setting the population size too large increases the algorithm's time cost during the search process, thereby hindering overall performance. Conversely, a population size that is too small restricts diversity within the population, which is also detrimental to performance. Both excessively high or low discount rates and learning rates can impact the algorithm's performance. This is due to the high sensitivity of these parameters in the neural network model learning and optimization process using policy gradient methods. Improper parameter selection can slow down convergence to some extent and may lead to instability in the training process, thereby affecting the learning efficiency and final performance of the model within a specified time frame.

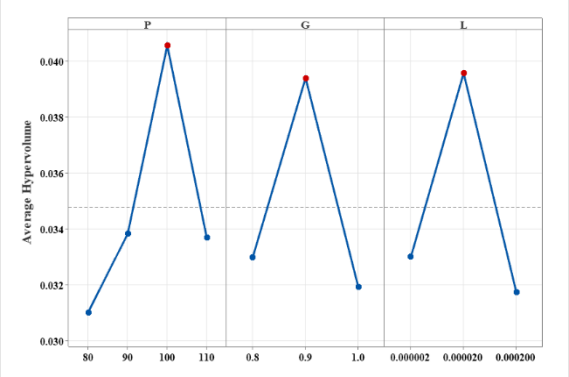

Figure 5: Main effect plot of HV

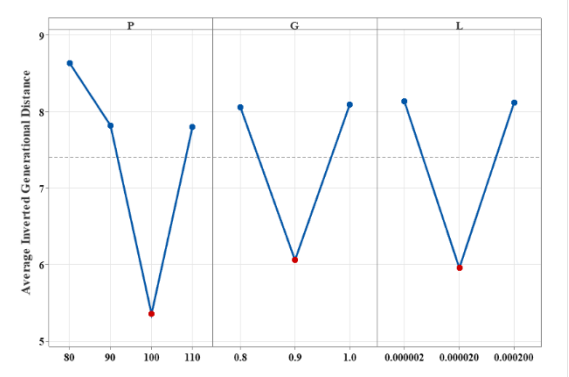

Figure 6: Main effect plot of IGD

4.2. Comparison with Other Algorithms

This experiment thoroughly validates the effectiveness and superiority of the GEARL algorithm in the MOFFSP problem by comparing it with three representative multi-objective problem-solving algorithms. The three comparative algorithms are NSGAIISDR [26], MOEA/D-URAW [27], and SSCEA [28]. To maintain fairness and consistency in the algorithm comparisons, unified standards were maintained in several key aspects, including the problem encoding and decoding scheme, the optimization objectives pursued, and the termination conditions of the experiments. In the experiment, the three comparative algorithms NSGAIISDR, MOEA/D-URAW, and SSCEA are denoted by $A1$, $A2$, and $A3$ respectively, while GEARL is denoted by A. $C1$ and $C2$ represent the domination situation of all solutions between A and $A1$, and similar comparisons for other algorithms.

Table 3: Presents the experimental results for HV

Problem	$A1$	$A2$	$A3$	A
Remanu01	0.0407	0.0413	0.0282	**0.0460**
Remanu02	0.0686	0.0453	0.0333	**0.0858**
Remanu03	0.0428	0.0298	0.0239	**0.0472**
Remanu04	0.1136	0.0613	0.0285	**0.1196**
Remanu05	0.0664	0.0275	0.0200	**0.1073**
Remanu06	0.0377	0.0191	0.0187	**0.0856**
Remanu07	0.0481	0.0269	0.0145	**0.1396**
Remanu08	0.0546	0.0337	0.0153	**0.2073**

Table 4: Presents the experimental results for IGD

Problem	$A1$	$A2$	$A3$	A
Remanu01	1.3821	1.2571	4.2321	**0.8634**
Remanu02	8.4384	21.291	35.167	**3.0996**
Remanu03	5.8900	13.447	20.545	**3.2076**
Remanu04	12.322	50.802	100.89	**5.9585**
Remanu05	72.682	177.01	175.61	**4.4567**
Remanu06	76.689	123.41	129.42	**10.833**
Remanu07	263.45	368.17	435.60	**90.682**
Remanu08	390.02	556.12	634.82	**136.78**

Table 5: Presents the experimental results for C (A, B)

Problem	$C1$	$C2$	$C3$	$C4$	$C5$	$C6$
Remanu01	**1.00**	0.00	**0.50**	0.00	**1.00**	0.00
Remanu02	**0.89**	0.00	**1.00**	0.00	**1.00**	0.00
Remanu03	**0.87**	0.20	**1.00**	0.00	**1.00**	0.00
Remanu04	**0.85**	0.00	**1.00**	0.00	**1.00**	0.00
Remanu05	**1.00**	0.00	**1.00**	0.00	**1.00**	0.00
Remanu06	**1.00**	0.00	**1.00**	0.00	**1.00**	0.00
Remanu07	**1.00**	0.00	**1.00**	0.00	**1.00**	0.00
Remanu08	**1.00**	0.00	**1.00**	0.00	**1.00**	0.00

Tables 3-5 provides a detailed display of the performance of the GEARL algorithm and the comparative algorithms across different metrics, where each cell represents the average value of the corresponding problem metric, with the best results highlighted in bold. From the Tables 3-5, it is observed that the GEARL algorithm demonstrates significant advantages on most test sets. While the algorithm shows relatively shorter running times when handling small-scale problems (attributed to the limitation of problem size), it does not significantly stand out in terms of metric performance. However, as the problem scale gradually increases, the superiority of the GEARL algorithm becomes more apparent. This is mainly due to its longer running time, allowing the policy model to be thoroughly trained, enabling precise selection of the best evolutionary strategy for each individual. This strategy selection mechanism greatly accelerates the optimization process of the population. In contrast, comparative algorithms require a significant amount of time to search for solutions in the domain. Therefore, when faced with complex and large-scale problems, the GEARL algorithm demonstrates more prominent performance with its efficient strategy selection and optimization mechanism.

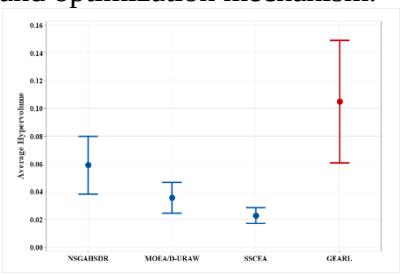

Figure 7: Interval diagram of HV

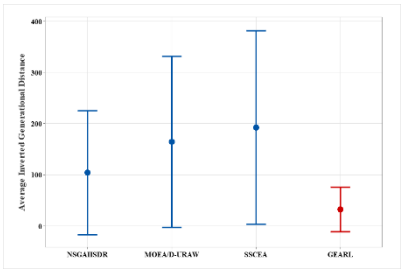

Figure 8: Interval diagram of IGD

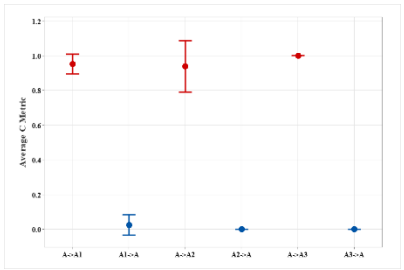

Figure 9: Interval diagram of C Metric

Figures 7-9 visually present the experimental results of various comparative algorithms, specifically displaying interval diagrams within a 95% confidence interval for three different performance metrics. It is evident from the diagrams that in the interval representations of each metric, the GEARL algorithm exhibits the best mean performance, surpassing the compared algorithms in all performance metrics. The GEARL algorithm demonstrates stronger robustness in terms of convergence and result distribution. Combining the experimental data and graphical analysis mentioned above, the GEARL algorithm, for each population individual, can accurately provide guidance tailored to current evolutionary needs through its unique policy model, thereby more rapidly and stably guiding the population to converge to the optimal solution set.

Figure 10 presents a comparison of the Pareto fronts obtained by various algorithms for different problem instances. The purpose of this graph is to visually showcase the final optimization results of different algorithms for each problem. Through the macroscopic display of the Pareto front graph, the significant effectiveness of the GEARL algorithm in solving MOFFJSP is observed. The

solution set positions of the GEARL algorithm are superior, closer to the coordinate origin and the horizontal and vertical axes, indicating excellent performance in reducing both the maximum completion time and total machine workload. Compared to other algorithms, the GEARL algorithm can explore and find a higher-quality Pareto optimal solution set within the same computational time, validating its efficiency and practicality in solving the MOFFJSP.

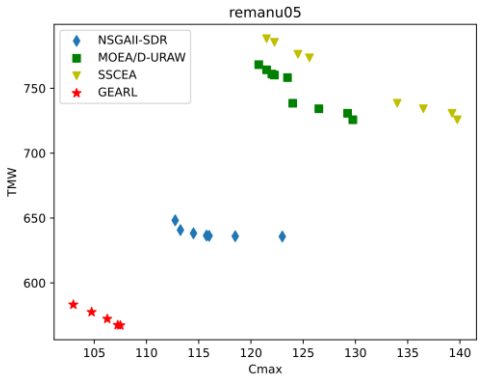

Figure 10: Pareto front diagram

5. Conclusions and Future work

In this research, GEARL was introduced to tackle the MOFFJSP, aiming to optimize total completion time and machine total load. The algorithm, structured around population initialization, fuzzy disjunctive graph, policy model, and solution set optimization modules, effectively guides population evolution. By utilizing graph information to represent individual data and selecting tailored evolutionary strategies for each, GEARL demonstrates significant advantages in addressing the MOFFJSP.

Future endeavours will involve integrating cutting-edge algorithms such as state-of-the-art large-scale models to further enhance and refine the existing model. The goal is to optimize a broader range of objectives simultaneously, thereby developing a production scheduling model that better aligns with real-world requirements

References

[1] SHEN L, DAUZ èRE-P èR èS S, MAECKER S. Energy cost efficient scheduling in flexible job-shop manufacturing systems [J]. European Journal of Operational Research, 2023, 310(3): 992-1016.

[2] SONG W, CHEN X, LI Q, et al. Flexible Job-Shop Scheduling via Graph Neural Network and Deep Reinforcement Learning [J]. IEEE Transactions on Industrial Informatics, 2023, 19(2): 1600-1610.

[3] GAO K, CAO Z, ZHANG L, et al. A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems [J]. IEEE/CAA Journal of Automatica Sinica, 2019, 6(4): 904-916.

[4] FAN J, SHEN W, GAO L, et al. A hybrid Jaya algorithm for solving flexible job shop scheduling problem considering multiple critical paths [J]. Journal of Manufacturing Systems, 2021, 60: 298-311.

[5] CHEN R, YANG B, LI S, et al. A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem [J]. Computers & Industrial Engineering, 2020, 149.

[6] SUN L, LIN L, GEN M, et al. A Hybrid Cooperative Coevolution Algorithm for Fuzzy Flexible Job Shop Scheduling [J]. IEEE Transactions on Fuzzy Systems, 2019, 27(5): 1008-1022.

[7] ZHANG Z-Q, WU F-C, QIAN B, et al. A Q-learning-based hyper-heuristic evolutionary algorithm for the distributed flexible job-shop scheduling problem with crane transportation [J]. Expert Systems with Applications, 2023, 234.

[8] LUO C, GONG W, LU C. Knowledge-driven two-stage memetic algorithm for energy-efficient flexible job shop scheduling with machine breakdowns [J]. Expert Systems with Applications, 2024, 235.

[9] ZHU N, GONG G, LU D, et al. An effective reformative memetic algorithm for distributed flexible job-shop

scheduling problem with order cancellation [J]. Expert Systems with Applications, 2024, 237.
[10] LI W, HE L, CAO Y. Many-Objective Evolutionary Algorithm With Reference Point-Based Fuzzy Correlation Entropy for Energy-Efficient Job Shop Scheduling With Limited Workers [J]. IEEE Transactions on Cybernetics, 2022, 52(10): 10721-10734.
[11] GAO D, WANG G-G, PEDRYCZ W. Solving Fuzzy Job-Shop Scheduling Problem Using DE Algorithm Improved by a Selection Mechanism [J]. IEEE Transactions on Fuzzy Systems, 2020, 28(12): 3265-3275.
[12] CHEN X-L, LI J-Q, DU Y. A hybrid evolutionary immune algorithm for fuzzy flexible job shop scheduling problem with variable processing speeds [J]. Expert Systems with Applications, 2023, 233.
[13] ABDEL-BASSET M, MOHAMED R, EL-SHAHAT D, et al. An efficient hybrid optimization method for Fuzzy Flexible Job-Shop Scheduling Problem: Steady-state performance and analysis [J]. Engineering Applications of Artificial Intelligence, 2023, 123.
[14] SHAO W, SHAO Z, PI D. An Ant Colony Optimization Behavior-Based MOEA/D for Distributed Heterogeneous Hybrid Flow Shop Scheduling Problem Under Nonidentical Time-of-Use Electricity Tariffs [J]. IEEE Transactions on Automation Science and Engineering, 2022, 19(4): 3379-3394.
[15] ZHAO F, WANG Z, WANG L. A Reinforcement Learning Driven Artificial Bee Colony Algorithm for Distributed Heterogeneous No-Wait Flowshop Scheduling Problem With Sequence-Dependent Setup Times [J]. IEEE Transactions on Automation Science and Engineering, 2023, 20(4): 2305-2320.
[16] ZHU K, GONG G, PENG N, et al. Dynamic distributed flexible job-shop scheduling problem considering operation inspection [J]. Expert Systems with Applications, 2023, 224.
[17] HU Y, ZHANG L, ZHANG Z, et al. Matheuristic and learning-oriented multi-objective artificial bee colony algorithm for energy-aware flexible assembly job shop scheduling problem [J]. Engineering Applications of Artificial Intelligence, 2024, 133.
[18] AKRAM K, BHUTTA M U, BUTT S I, et al. A Pareto-optimality based black widow spider algorithm for energy efficient flexible job shop scheduling problem considering new job insertion [J]. Applied Soft Computing, 2024, 164.
[19] SI J, LI X, GAO L, et al. An efficient and adaptive design of reinforcement learning environment to solve job shop scheduling problem with soft actor-critic algorithm [J]. International Journal of Production Research, 2024, 62(23): 8260-8275.
[20] ZHANG W, ZHAO F, LI Y, et al. A novel collaborative agent reinforcement learning framework based on an attention mechanism and disjunctive graph embedding for flexible job shop scheduling problem [J]. Journal of Manufacturing Systems, 2024, 74: 329-345.
[21] PARK J, CHUN J, KIM S H, et al. Learning to schedule job-shop problems: representation and policy learning using graph neural network and reinforcement learning [J]. International Journal of Production Research, 2021, 59(11): 3360-3377.
[22] ZHAO F, XU Z, BAO H, et al. A cooperative whale optimization algorithm for energy-efficient scheduling of the distributed blocking flow-shop with sequence-dependent setup time [J]. Computers & Industrial Engineering, 2023, 178.
[23] BŁAŻEWICZ J, PESCH E, STERNA M. The disjunctive graph machine representation of the job shop scheduling problem [J]. European Journal of Operational Research, 2000, 127(2): 3173-3185.
[24] WU Z, PAN S, CHEN F, et al. A Comprehensive Survey on Graph Neural Networks [J]. IEEE Transactions on Neural Networks and Learning Systems, 2021, 32(1): 4-24.
[25] GAO K Z, SUGANTHAN P N, PAN Q K, et al. An effective discrete harmony search algorithm for flexible job shop scheduling problem with fuzzy processing time [J]. International Journal of Production Research, 2015, 53(19): 5896-5911.
[26] TIAN Y, CHENG R, ZHANG X, et al. A Strengthened Dominance Relation Considering Convergence and Diversity for Evolutionary Many-Objective Optimization [J]. IEEE Transactions on Evolutionary Computation, 2019, 23(2): 331-345.
[27] FARIAS L R C, ARAúJO A F R. Many-Objective Evolutionary Algorithm Based On Decomposition With Random And Adaptive Weights [J]. IEEE International Conference on Systems, Man and Cybernetics (SMC), 2019.
[28] LIU G, PEI Z, LIU N, et al. Subspace segmentation based co-evolutionary algorithm for balancing convergence and diversity in many-objective optimization [J]. Swarm and Evolutionary Computation, 2023, 83.