FRE-Based Sparrow Search Algorithm for Green Flexible Job Shop Scheduling

Ziming Xue 1049043007@QQ.COM
School of Mechanical and Automotive Engineering, Shanghai University of Engineering Science, Shanghai 201620, China

Jun Zhou[*] ZHOUJUN@SUES.EDN.CN
School of Mechanical and Automotive Engineering, Shanghai University of Engineering Science, Shanghai 201620, China
[*]*Corresponding author*

Editors: Nianyin Zeng, Ram Bilas Pachori and Dongshu Wang

Abstract

The modern manufacturing is facing the challenge of energy saving and emission reduction. This study addresses the Multi-objective Green Flexible Job-shop Scheduling Problem (MGFJSP) with three objectives makespan, machine workload and carbon emissions, a Fuzzy Relative Entropy (FRE)-based improved Sparrow Search Algorithm (FISSA) is proposed. FISSA begins with special initialize methods to ensure a uniform distribution in solution space. Next, a logarithmic spiral is introduced in scroungers to enhance global search capability. Additionally, an insertion strategy is implemented to reduce machine idle time and carbon emissions. Finally, a FRE coefficient is introduced, where solutions are evaluated by comparing them with the ideal point, diversity is quantified, and selection is guided. Experimental results confirm that FISSA outperforms other multi-objective algorithms, significantly minimizing processing time and carbon emissions, demonstrate superior robustness and convergence.

Keywords: sparrow search algorithm; flexible job-shop scheduling; fuzzy relative entropy; low-carbon manufacturing; multi-objective optimization

1. Introduction

The degradation of the ecological environment and the increase in energy consumption have become global concerns. The manufacturing industry, as the primary energy consumer, is facing significant economic pressure. Low-carbon manufacturing, as a sustainable production method, can effectively fulfill the energy conservation and emission reduction requirements of enterprises. In recent years, considering the influence of energy consumption in the production scheduling process has become a vital aspect of intelligent scheduling research (Zhu et al., 2023).

Flexible Job-shop Scheduling Problem (FJSP) has been proven to be an NP-hard (Ding and Gu, 2020) problem. In recent years, FJSP has evolved towards green, low-carbon, and multi-objective optimization, leading to significant research achievements. Shi et al. (2024) considered total machine energy consumption in FJSP and proposed a multi-objective evolutionary algo-rithm with decomposition (MOEA/D) algorithm based on reinforcement learning, incorporating deep Q-learning to enhance its solving performance. Zhu et al. (2020) introduced a novel carbon emission model that considers both machine emissions and workers' learning effects, solving it using a Memetic Algorithm (MA). Lei et al. (2018) proposed an improved Artificial Bee Colony (ABC) algorithm

© 2025 Z. Xue & J. Zhou.

to address the low-carbon FJSP with variable processing speed constraints, minimizing energy consumption and completion time. Introducing low-carbon objectives results in a Multi-objective Optimization Problem (MOP), and the fitness evaluation mechanism is crucial in optimizing algorithm performance (Ma et al., 2023). The existing Pareto-based algorithms face challenges including difficulties in solution se-lection and performance degradation with an increasing number of objectives (Deng et al., 2022). This study introduces a fuzzy relative entropy (FRE) coefficient as a criterion to address these challenges.

Swarm Intelligence Algorithm (SIA), a common intelligence optimization method, has been widely applied in shop scheduling (Del Gallo et al., 2023). Xu et al. (2024) integrated Quantum Particle Swarm Optimization (QPSO) with Variable Neighborhood Search (VNS), employing chaotic encoding and nine neighborhood structures to enhance search capability, demonstrating superior stability in solving FJSP. (Long et al., 2022) proposed an improved self-learning ABC, which dynamically adjusts the update dimensions of ABC through reinforcement learning, resulting in faster convergence and better solution quality for FJSP. Kong et al. (2022) improved Grey Wolf Optimization (GWO) by incorporating chaotic mapping and designing a discrete update operator with an adaptive convergence factor. Experiments validate its improved accuracy in FJSP applications. Another novel SIA optimization algorithm, the Sparrow Search Algorithm (SSA), was proposed by Xue and Shen (2020) in 2020, based on the foraging and anti-predatory behaviors of sparrow populations. Compared to other algorithms, SSA exhibits strong competitiveness in search accuracy, convergence speed, and stability, and has been applied to various fields. Ma et al. (2024) proposed an enhanced multi-strategy sparrow search algorithm that incorporates the Levy flight strategy to optimize the trajectory of a six-axis industrial robot, ad-dressing issues of low efficiency and high vibration. In order to solve JSP Li et al. (2024) introduced genetic operators into the SSA to enhance its search capability and incorporated simulated annealing to avoid local optima. The results demonstrate that the improved SSA exhibits better performance. Zhang et al. (2024) addressed the issue of declining search capability in SSA during the later stages of iterations by employing chaotic cube mapping to initialize the population and using the firefly algorithm to perturb the individuals. This approach was ultimately applied to UAV inspection path planning. Du et al. (2023) incorporated reverse learning and tent mapping into SSA, improving the quality of the initial solutions and enhancing its local search performance. Ultimately, the modified SSA was combined with 2D Otsu to improve the accuracy of image seg-mentation. It is evident that SSA tends to fall into local optima, and currently, there is only a few of research focused on workshop scheduling, and our study fills the gap in FJSP.

The SSA has a fast convergence speed and high solution accuracy and performs well when facing multi-objective high dimensional space problems. This study formulates a low-carbon manufacturing oriented FJSP, considering three objectives: makespan, machine load, and carbon emissions. The structure of this work is organized as follows: Section 2 presents problems and notations; Section 3 details FISSA improvements; Section 4 applies FISSA to MGFJSP benchmarks; Section 5 concludes with key findings and contributions.

2. The MILP of Multi-Objective Green Flexible Job Shop Scheduling

2.1. Problem Description

The Multi-objective Green Flexible Job-shop Scheduling Problem (MGFJSP) can be described as follows: There are n jobs $J = \{J_1, J_2, \cdots, J_i, \cdots J_n\}$ can be processed on m machines $M =$

$\{M_1, M_2, \cdots, M_k, \cdots M_m\}$. Each job has its own operations $O_i = \{O_{i1}, O_{i2}, \cdots, O_{ij}, \cdots O_{in_i}\}$, and the same operation can be processed on different machines. For the i-th job with j-th operation O_{ij}, there exists a machine set $\Phi_{ij} = \left\{\varphi_{ij1}, \varphi_{ij2}, \cdots, \varphi_{ijl}, \cdots, \varphi_{ijm_{ij}}\right\}$, allowing the operation processed on any machine within the set Φ_{ij}, where $\varphi_{ijl} \in M$. Due to variation production environments on each machine, the power consumption and processing time for the same operation can differ depending on the machine.

Thus, MGFJSP needs to determine the processing machine and the sequence of operations, by arranging the order of operations and selecting machines, a scheduling solution that meets the optimization objectives will be generated. Additionally, the following constraints are considered: All machines are initially in an idle state and waiting for processing; once the processing begins, it cannot be interrupted; each operation can only be processed once and cannot be repeated; each machine can only process one operation at a time; and there is no priority between different jobs.

2.2. Objcectives and Constraints

The optimization objectives of the MGFJSP model include the maximum processing time C_{max}, as shown in Equation (1); the maximum workload W_{max}, as shown in Equation (2); and the whole carbon emissions CE, as shown in Equation (3), which are determined by the load and idle power consumption of all machines, and CEF represents the carbon emission factor. The calculation of machine idle time is given in Equation (4), S_{ijk}, C_{ijk} and W_{ijk} represent the start time, completion time and processing time of O_{ij} on machine k. pw_{ijk} and pi_k denote the load and idle power of machine k.

$$C_m = \min\left\{\max\left\{C_{ijk}\right\}\right\}, \forall i \in J, \forall j \in O_i, \forall k \in M \tag{1}$$

$$W_{max} = \min\left\{\max\left\{\sum_{i=1}^{n}\sum_{j=1}^{n_i} W_{ijk}X_{ijk}\right\}\right\}, \forall k \in M \tag{2}$$

$$CE = \min\left\{CEF\left(\sum_{i=1}^{n}\sum_{j=1}^{n_i}\sum_{k=1}^{m} W_{ijk}X_{ijk}pw_{ijk} + \sum_{k=1}^{m} I_k pi_k\right)\right\} \tag{3}$$

$$I_k = \sum_{k=1}^{m}\left(\max\left\{C_{ijk}\right\} - \sum_{i=1}^{j}\sum_{j=1}^{n_j}\left(C_{ijk} - S_{ijk}\right)X_{ijk}\right) \tag{4}$$

$$\sum_{k=1}^{m} X_{ijk} = 1, \forall i, j, k \tag{5}$$

$$S_{ijk} \geq C_{i(j-1)k} \tag{6}$$

$$X_{ijk}S_{ijk} \geq X_{i\prime j\prime k}C_{i\prime j\prime k} \tag{7}$$

$$\left(S_{ijk} - C_{i'j'k}\right)Y_{iji'j'} \geq 0 \tag{8}$$

$$X_{ijk} \in \{0, 1\}, \forall i, j, k \tag{9}$$

$$Y_{iji'j'} \in \{0, 1\}, \forall i, j, i', j' \tag{10}$$

Equation (5) stipulates that each operation can only be processed once on one machine; Equation (6) requires that each operation must be executed in a specific sequence; Equation (7) restricts that each machine can be processed only one operation at any time; and Equation (8) defines the sequential relationship between adjacent operations on the same machine. Equations (9) and (10) define the range of state variables. When operation O_{ij} is processed on machine k, the state variable X_{ijk} is 1, otherwise, it is 0; when operation O_{ij} is processed before operation $O_{i\prime j\prime}$, the state variable $Y_{iji'j'}$ is 1, otherwise it is 0.

3. FISSA for MGFJSP

3.1. Improvement of SSA

SSA stimulates the foraging behavior of sparrows and their strategies for evading predators. In SSA, the sparrow population is divided into two subgroups: producers and scroungers. Producers are responsible for finding food, while scroungers observe and forage by following the producers. Additionally, some sparrows in the population act as alert reconnaissance agents, detecting potential threats and triggering an anti-predation response when danger is perceived.

Producers, as individuals with higher fitness, have priority access to food and relocate to safer areas when sensing danger. The behavior can be described as Equation (11). Scroungers compete with producers for food, with their update rule defined in Equation (12). Furthermore, both producers and scroungers have a 10%-20% probability of triggering an anti-predation response, as shown in Equation (13).

$$x_{ij}^{t+1} = \begin{cases} x_{ij}^{t} \cdot \exp\left(\frac{-i}{\alpha \cdot \text{iter}_{\text{max}}}\right) & r < ST \\ x_{ij}^{t} + Q & r \geq ST \end{cases} \tag{11}$$

$$x_{ij}^{t+1} = \begin{cases} Q \cdot \exp\left(\frac{xw_j^t - x_{ij}^t}{i^2}\right) & i > \frac{n}{2} \\ xp_j^{t+1} + \frac{1}{D}\sum_{j=1}^{D} \text{rand}\{-1,1\} \cdot \left(\left|x_{ij}^t - xp_j^{t+1}\right|\right) & \text{other} \end{cases} \tag{12}$$

$$x_{ij}^{t+1} = \begin{cases} xb_j^t + Q \cdot \left|x_{ij}^t - xb_j^t\right| & \text{other} \\ x_{ij}^t + k \cdot \left(\frac{\left|x_{ij}^t - xw_j^t\right|}{(f_i - f_w) + \varepsilon}\right) & f_i = f_g \end{cases} \tag{13}$$

The variable x_{ij} represents the value of the j-th parameter for the i-th sparrow. xp_j^t and xw_j^t denote the j-th parameter values of the best and worst fitness individuals in the population, respectively. t represents the current iteration number, while $iter_{max}$ denotes the maximum number of iterations. Both α and r are uniformly distributed random numbers with a $U(0,1)$ parameter, whereas Q follows a standard normal distribution. The variable r represents the sparrow's alertness value, and $ST \in (0.5, 1]$ denotes the safety threshold. Moreover, f_i, f_w, and f_g represent the fitness values of the i-th, the worst, and the best individual in the population, respectively.

In SSA, scroungers are individuals with lower fitness in the population and will move to the positions of producers. Although this strategy quickly reduces the variance between the encodings of producers and scroungers, leading to rapid convergence, it also increases the risk of getting trapped in local optima. To address this issue, the logarithmic spiral is applied to the movement process of scroungers, as shown in Equation (14). The parameter l, whose value increases with the number of iterations, is defined in Equation (15)

$$x_{ij}^{t+1} = \begin{cases} \text{random}(0,1)\left|x_{\max}^{t} - x_{\min}^{t}\right| + x_{\min}^{t} & i > \frac{N}{2} \\ x_{p}^{t} + \left|x_{p}^{t} - x_{ij}^{t}\right| \times a \times e^{bl} \times \cos(2\pi l) & \text{other} \end{cases} \tag{14}$$

$$l = \ random\left(0\ ,\ 2 \times \left(1 - \frac{Iter}{Iter_{max}}\right)\right) \tag{15}$$

3.2. Coding and Initialization

The scheduling solution is represented by using two parts of vector: the MS vector $[\sigma_1, \sigma_2, \cdots, \sigma_p, \cdots \sigma_D]$ indicates the machine assignment for each operation, in contrast, the OS vector $[\lambda_1, \lambda_2, \cdots, \lambda_p, \cdots \lambda_D]$ represents the processing sequence of operations. Specifically, σ_p denotes that operation O_p is assigned to the σ_p-th machine $\varphi_{p\sigma_p}$ in its available machine set Φ_p. Meanwhile, λ_p indicates that operation $O_{\lambda_p j\prime}$ is scheduled in the p-th position, where $j\prime$ is the number of times λ_p appears among the first p elements of the OS vector, $D = \sum_{i=1}^{n} n_i$. Figure 1(a) shows a scheduling solution, where $\sigma_4 = 4$ indicates that O_{22} is assigned to the forth machine in its available machine set Φ_{22}. The processing sequence of this solution is $O_{21} \rightarrow O_{11} \rightarrow O_{22} \rightarrow O_{31} \rightarrow O_{12} \rightarrow O_{32} \rightarrow O_{23}$.

Since the MGFJSP problem is a discrete scheduling optimization problem, while SSA generates decimal-coded values during iterations, it is necessary to perform a decimal-to-integer mapping. In the MS vector, mapping is conducted by using Equations (16) and (17), where β represents the upper and lower bounds of decimal genes. $D(p)$ and $I(p)$ denote the decimal and integer forms of the p-th gene, respectively. The OS vector's decimal values are mapped to integers based on their relative magnitudes. Conversely, a random sequence within $\{-\beta,\ \beta\}$ is generated and mapped to integers according to the encoding scheme. Figure 1(b) illustrates the mapping process. Additionally, The GLR machine selection method is adopted in this study to initialize the populations (Zhang et al., 2011).

$$I\left(p\right) = \ round\left(\frac{1}{2\beta}\left(D\left(p\right) + \beta\right)\left(r\left(p\right) - 1\right) + 1\right) \tag{16}$$

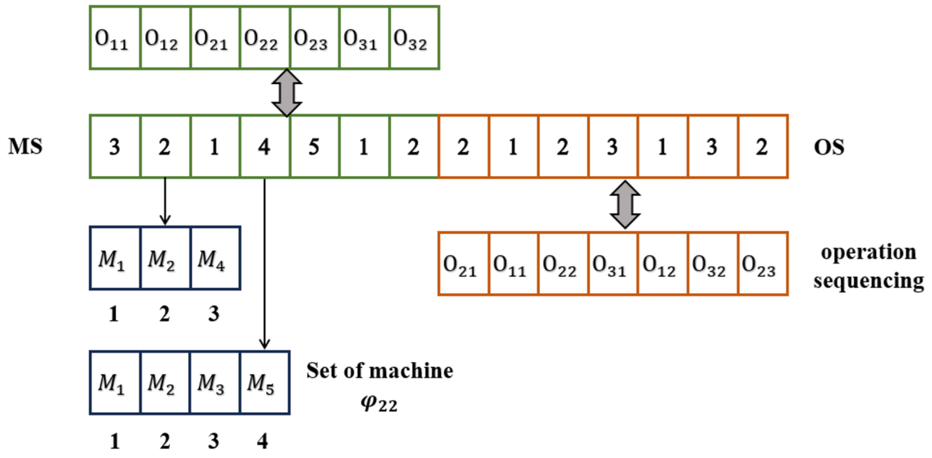
$$D\left(p\right) = \ \frac{2\beta}{r\left(p\right) - 1}\left(I\left(p\right) - 1\right) - \beta \tag{17}$$

3.3. Insertion strategy

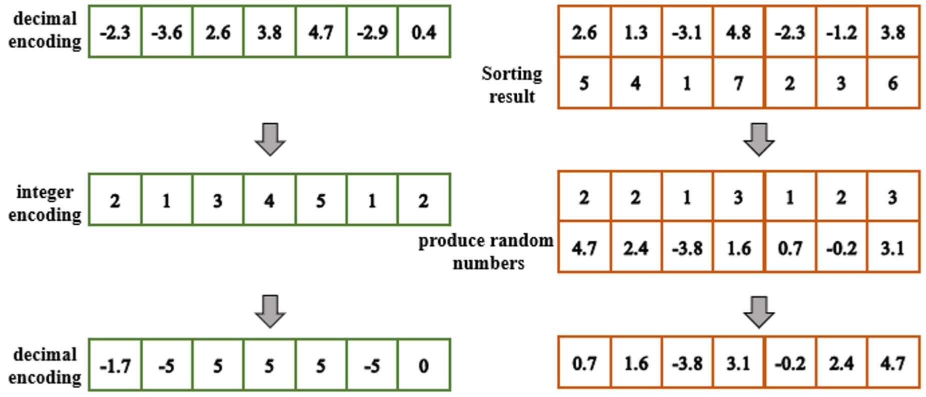
Despite there are conflicts among the optimization objectives of MGFJSP, reducing machine idle time has a positive impact for three objectives. This study proposes an Insertion strategy to reduce both machine idle time and energy consumption.

Specifically, for each machine in a scheduling solution, traverse its assigned operations set $\Omega_k = \{O_{k1}, O_{k2}, \cdots O_{kp}, \cdots O_{kn_k}\}$ and idle time set $idle_k = \{I_{k1},\ I_{k2}, \cdots I_{kl}, \cdots I_{kI}\}$, check each operations in Ω_k from back to front. If there exists an operation $O_{kp} = O_{ij} \in \Omega_k$ and an idle interval $I_{kl} \in idle_k$ such that the processing time $W_{ijk} < I_{kl}$, and the inserted position satisfies the $S_{ijk} \geq C_{i(j-1)k\prime}$, then O_{kp} is inserted into the corresponding idle interval, and then update the OS vector accordingly. This process is recursively applied until all operations have been evaluated.

This approach not only reduces machine idle time but also lowers machine workload, thereby decreasing total energy consumption. As shown in Figure 2, first only operations O_{32} can be inserted into the preceding idle time slots, but after that O_{33} can be inserted either because of the satisfaction of the operation sequential constrains. This process reduces the makespan from 12 to10.



(a) MSOS vector



(b) Mapping process

Figure 1: Coding process.

3.4. Population select strategy

This study proposes an FRE-based ranking method to quantify the quality of solutions. Fuzzy relative entropy originates from fuzzy set theory. Unlike classical sets, a type-1 fuzzy set F on a given domain X is defined as Equation (18), which F is mapped by the fuzzy membership function $\mu_F(x)$ to the range of [0,1] on domain X. $\mu_F(x)$ represents the fuzzy membership degree to element x in F, reflecting its fuzziness within the set F.

$$F : X \to [0, 1], \quad x \mapsto \mu_F(x) \tag{18}$$

To compare the uncertainty differences between two fuzzy sets, FRE must be calculated. Given two fuzzy sets A and B with membership functions $\mu_A(x)$ and $\mu_B(x)$, their corresponding information entropy can be computed as show in Equations (19). $E(A)$ represents the information entropy of fuzzy set A, while K is a normalization index, and x_i denotes an element in the domain U. Information entropy measures the uniformity of element distribution within the fuzzy set. Subsequently,

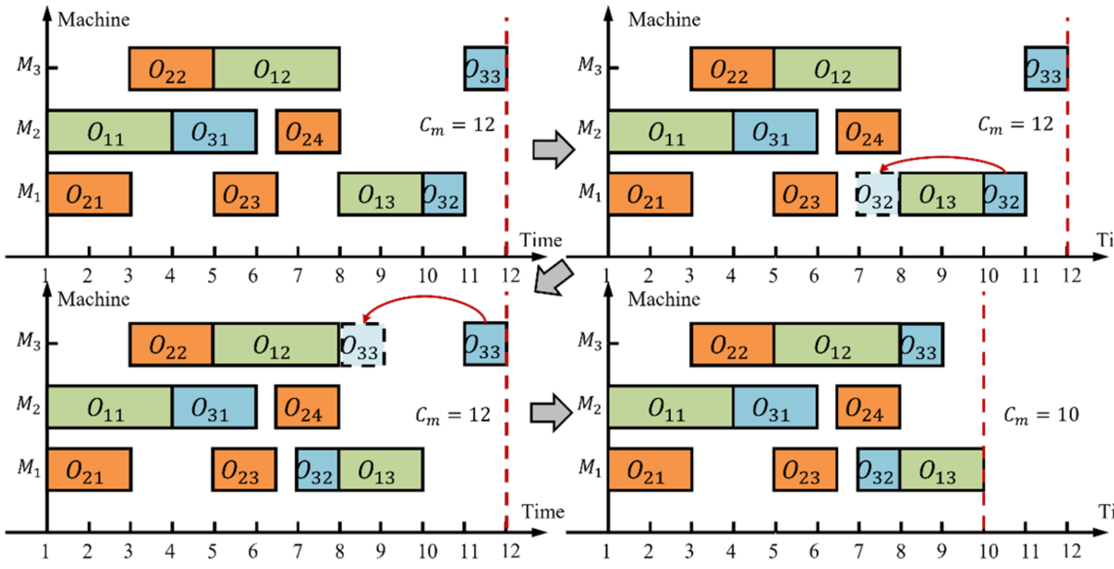


Figure 2: Insertion process.

the relative entropy between A and B is calculated using Equations (20).

$$E_A = -K \sum_{x_i \in U} \{\mu_A(x_i) ln(\mu_A(x_i)) + [1 - \mu_A(x_i)]\, ln(1 - \mu_A(x_i))\} \tag{19}$$

$$E_A(B) = -K \sum_{x_i \in U} \{\mu_A(x_i)\, ln(\mu_B(x_i)) + [1 - \mu_A(x_i)]\, ln(1 - \mu_B(x_i))\} \tag{20}$$

Let $E_A(B)$ be the relative entropy, it quantifies the difference of fuzzy set A compared to B. Equations (21) and (22) define the Fuzzy Relative Entropy $E(A, B)$ and the Fuzzy Relative Entropy Coefficient $R(A, B)$. $R(A, B)$ effectively reflects the differences between fuzzy sets, a higher value indicates greater similarity between A and B (Guo and Xin, 2005). Satisfying $0 \leq R(A, B) \leq 1$, and $R(A, B) = 1$ if $A = B$.

$$E(A, B) = E_A(B) + E_B(A) \tag{21}$$

$$R(A, B) = \frac{E_A + E_B}{k \times E(A, B)} \tag{22}$$

As demonstrated above, it needs to express each solution as a fuzzy set and establish the corresponding mapping. Assuming that the values of each solution across M objectives belong to the domain U, the membership function for solution x_i is defined by Equation (23). Which, $f_j(x_p)$ and $f_j(x_w)$ represent the maximum and minimum values of the j-th objective within the solution set. The information entropy of the fuzzy set is calculated by using Equation (19), where $K = \frac{1}{M \times ln2}$. In the iterative process, the ideal point x_{ideal} can be expressed as $f(x_{ideal}) = \{f_1(x_p), \cdots, f_j(x_p) \cdots, f_M(x_p)\}$. Compare any solution x_i in the population with x_{ideal}, calculate its FRE coefficient according to Equation (22), then ranked and filtered population based on the magnitude of the FRE coefficient. The larger $R(x_i, x_{ideal})$ indicates that x_i is closer to the ideal point.

$$\mu_j(x_i) = \begin{cases} 0.999 & , f_j(x_i) \le f_j(x_p) \\ \frac{f_j(x_w)-f_j(x_i)}{f_j(x_w)-f_j(x_p)} & , f_j(x_p) < f_j(x_i) \le f_j(x_w) \\ 1\times 10^{-3} & , f_j(x_i) > f_j(x_p) \end{cases} \tag{23}$$

3.5. Algorithm framework

Step 1: Initialize relevant parameters such as population size, maximum iterations, and antipredation probability. Generate the initial population using the GLR method.

Step 2: Calculate the fitness of individuals and map the integer encoding to decimal encoding.

Step 3: Update the population positions based on Equations (11), (14), and (13).

Step 4: Map the decimal encoding back to integer encoding, compute the FRE coefficients for individuals, and sort the population accordingly.

Step 5: Perform insertion operations to the producer individuals for the next iteration.

Step 6: Check whether the maximum iteration is reached. If so, terminate the algorithm; otherwise, return to Step 2.

Step 7: Output the final set of non-dominated solutions.

3.6. Computational complexity analysis

The computational complexity of FISSA consists of three components. First, in Step 1, the population initialization for a population of size N has a complexity of $O(N\sum_{i=1}^{n} n_i)$, and the same complexity applies to the encoding and mapping process. In Step 3, position updates among individuals require $O(N\sum_{i=1}^{n} n_i)$ operations, and the FRE calculations in Step 4 have the same order of complexity. In Step 5, the insertion strategy is applied only to the producer individuals. In the worst case, this step incurs a complexity of $O(N_d\sum_{k=1}^{m} {n_k}^2)$. In summary, the overall computational complexity of FISSA can be expressed as $max(O\left(N\sum_{i=1}^{n} n_i\right), O(N_d\sum_{k=1}^{m} {n_k}^2))$.

This indicates that the FISSA's complexity is primarily influenced by the population size N, the number of operations per job n_i, the number of producer individuals N_d, and the number of operations assigned to each machine n_k.

4. Experiment and Result Analysis

In this section, the proposed algorithm is tested. The experimental environment is a 64-bit Windows 11 operating system, Intel (R) Core (TM) i7-12700h2.70 GHz processor and 32.00GB memory. The programming environment is Python 3.10.

4.1. Parameter settings

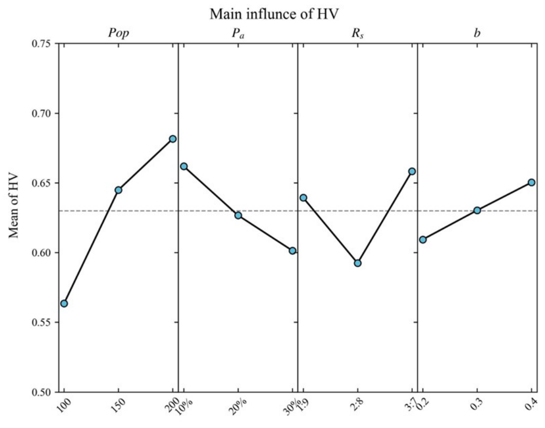
This experiment, 15 FJSP instances are used, including 10 MK instances and 5 Kacem instances. The load power of each operation on the machines follows a uniform distribution within the range [10, 20], while the idle power consumption follows a uniform distribution within the range [2, 10]. This study introduces two evaluation metrics: HV (Hyper-Volume), IGD (Inverted Generational Distance).

FISSA has four initial parameters: Population size Pop; Ratio of producers to scroungers R_s; Anti-predation probability P_a; Spiral growth rate b. The Taguchi method [17] is employed to determine the optimal initial parameters for the algorithm. Each parameter is divided into four levels, The values at different levels are as follows: $Pop = 100,\ 150,\ 200$; $R_s = 1:9,\ 2:8,\ 3:7$; $P_a = 10\%,\ 20\%, 30\%$; $b =\ 0.2,\ 0.3,\ 0.4$. FISSA is executed 20 times on the MK10 instance using different initialization parameters. The average values of the HV and IGD metrics are recorded, and the signal-to-noise ratio (S/N) for each level is calculated. The final results are presented in Table 1, with the highest values highlighted in bold. Based on a comprehensive analysis, the optimal parameter configuration is determined as follows: $Pop = 200$; $R_s = 2:8$; $P_a = 10\%$; $b = 0.2$.

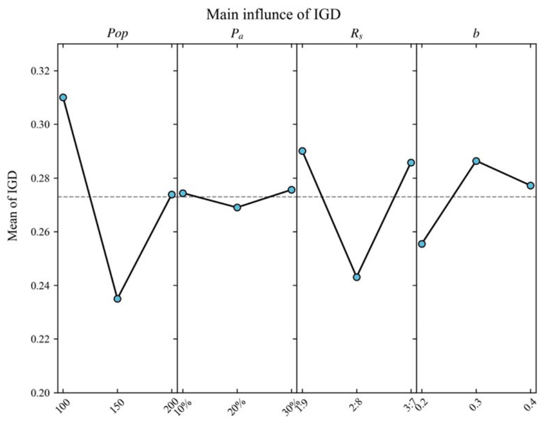
Table 1: S/N ratio for each level.

Parameters	Level 1	Level 2	Level 3
Pop	0.2407	0.2037	**0.3121**
P_a	**0.2895**	0.2470	0.2200
R_s	0.2359	**0.2687**	0.2520
b	**0.3040**	0.2081	0.2444

Figure 3 illustrates the effects of different parameter settings on two metrics. The vertical axis represents the average value of each indicator over 20 runs under each parameter level, while the dashed line represents the average the indexes at all parameter levels. As shown in the figure, it can be seen from the figure that the change of population number has the greatest influence on HV, and the change of R_s and b has the greatest influence on IGD, in contrast, P_a has the minor impact on IGD. Overall, the impact trends of each parameter on different levels are consistent with the initial parameter settings of FISSA, further validating the rationality of its parameter selection.



(a) Main influence of HV



(b) Main influence of IGD

Figure 3: Main influence of metrics.

4.2. Comparsion experiment

The superiority of FISSA is demonstrated through a comparative experiment with three algorithms: two diversity-based Multi-objective Optimization Evolution Algorithm NSGA-II, NSGA-III; and a decomposition-based MOEAs, MOEA/D (Zhang and Li, 2007). The initial parameters for all algorithms were set according to their optimal configurations from the original studies, maintaining the same population size as FISSA. The maximum number of iterations was set to 600, and each algorithm was independently executed 20 times. Table 2 presents the test results and average values of each algorithm.

Table 2: Comparison experiment result.

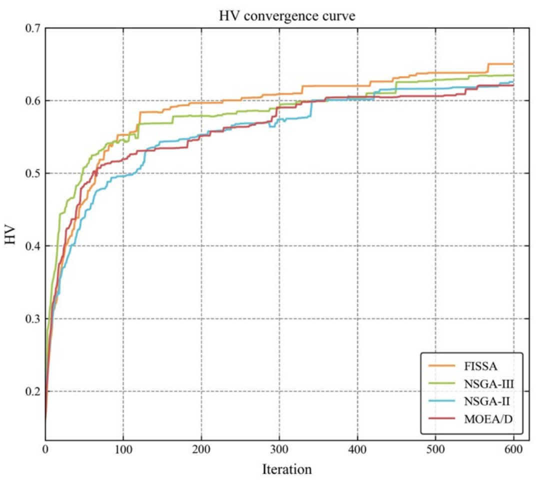
Instance	FISSA		NSGA-II		NSGA-III		MOEA/D	
	HV	IGD	HV	IGD	HV	IGD	HV	IGD
MK01	**0.7266**	**0.2511**	0.6950	0.3164	0.6852	0.3214	0.7234	0.2645
MK02	0.6986	**0.2537**	0.6941	0.2709	**0.7007**	0.2938	0.6601	0.2641
MK03	**0.6432**	0.2833	0.6584	0.2960	0.5226	**0.2710**	0.5584	0.3229
MK04	**0.6254**	**0.2623**	0.6163	0.2718	0.6134	0.2841	0.5654	0.2952
MK05	**0.6364**	**0.2436**	0.5575	0.3157	0.5416	0.2684	0.5369	0.2550
MK06	**0.6796**	**0.2668**	0.6030	0.2880	0.6267	0.3049	0.6085	0.2820
MK07	**0.6904**	0.2837	0.5714	0.3193	0.5451	0.3014	0.6319	**0.2647**
MK08	**0.6246**	**0.2679**	0.5081	0.3096	0.5797	0.2812	0.5813	0.2907
MK09	**0.6405**	**0.2753**	0.5625	0.2849	0.5713	0.3237	0.5409	0.3027
MK10	**0.6730**	**0.2516**	0.6271	0.2902	0.6322	0.3119	0.5914	0.2782
Kacem01	0.7034	0.2921	0.7080	0.2915	**0.7098**	0.3205	0.6985	**0.2515**
Kacem02	**0.6794**	**0.2544**	0.6292	0.3175	0.6461	0.2831	0.6515	0.2996
Kacem03	**0.6528**	**0.2743**	0.6429	0.2991	0.5837	0.2842	0.5854	0.2802
Kacem04	**0.6949**	**0.2546**	0.6620	0.2661	0.6304	0.2993	0.6204	0.3047
Kacem05	**0.7134**	0.2942	0.6327	0.3043	0.6869	**0.2859**	0.6457	0.3261
Average	**0.6721**	**0.2673**	0.6245	0.2961	0.6184	0.2957	0.6133	0.2855

Among the 15 benchmark instances, FISSA achieved the best HV in 13 cases This performance is primarily attributed to the efficient exploration of the solution space by scroungers and the application of the insertion strategy, which significantly enhance algorithm convergence. Although FISSA did not achieve the best HV in MK02 and Kacem01, its results remained very close to the optimal values. Notably, MOEA/D ,NSGA-II and NSGA-III both GA-based algorithms, they do not involve a decimal-to-integer mapping process, thereby avoiding precision loss during iterations is an issue that FISSA also encounters.

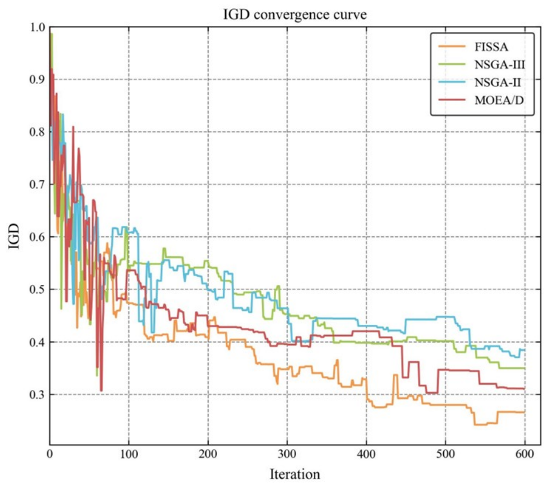
For the IGD metric, FISSA obtained the best values in 11 instances, by using the FRE coefficient as the ranking criterion, the algorithm effectively measures the similarity between solutions and the ideal point, allowing for a more precise approximation of the Pareto front and ultimately improving the overall performance of FISSA. Overall, FISSA increases HV by 7.9% and decreases IGD by 9.4% on average compared with other algorithms.

Figure 4 illustrates the convergence curves of each algorithm on the MK09 benchmark instance. As shown in Figure 4(a), FISSA exhibits a slower HV convergence rate than NSGA-II and MOEA/D during the first 100 iterations. However, after that, it demonstrates a significantly faster convergence trend and eventually surpasses the other algorithms, achieving the best HV value. In Figure 5(b), the IGD values of all algorithms fluctuate considerably in the early stages. After the 120 iterations, the IGD values begin to decline steadily and gradually stabilize. Notably, FISSA shows a marked advantage in the later stages, significantly outperforming the other algorithms in IGD, indicating superior solution distribution capability.

Figure 5 presents the Pareto front obtained by each algorithm for the MK09 instance. A solution set closer to the origin (lower-left corner) indicates a higher-quality solution set. Observations



(*a*) HV convergence curve



(*b*) IGD convergence curve

Figure 4: Convergence curve on MK09.

show that FISSA's solution set is closer to the origin compared to other algorithms, demonstrating both higher solution quality and better diversity. Additionally, Figure 6 provides the Gantt chart for the MK09 scheduling instance, which achieves minimal carbon emissions. The corresponding makespan, carbon emissions, and machine workload are $F = 420, 439.99, 325$.

5. Conclusion

This study explores the flexible job-shop scheduling problem under the green manufacturing principles. A multi-objective optimization model is formulated to minimize makespan, machine workload, and carbon emissions. To solve this problem, an improved Sparrow Search Algorithm based on FRE is proposed. The algorithm enhances the local search capability of scroungers using a logarithmic spiral, effectively reduces machine load and carbon emissions through an insertion strategy. Additionally, the introduction of the FRE coefficient further enhances population diversity. Comparative experiments demonstrate that FISSA outperforms other algorithms across several performance metrics. The Pareto front visualization indicate that FISSA exhibits strong robustness and a better Pareto front, proving its effectiveness in solving the MGFJSP.

A more sustainable scheduling approach is provided for FJSP, optimizing machine assignment and significantly reducing energy consumption and machine workload while improving production efficiency. Future research could explore unexpected disruptions in production, such as machine failures and urgent orders, while integrate machine learning techniques to achieve more precise scheduling solutions.

References

M. Del Gallo, G. Mazzuto, F. E. Ciarapica, and M. Bevilacqua. Artificial intelligence to solve production scheduling problems in real industrial settings: Systematic literature review. *Electronics*, 12(23):4732, 2023. doi: 10.3390/electronics12234732.

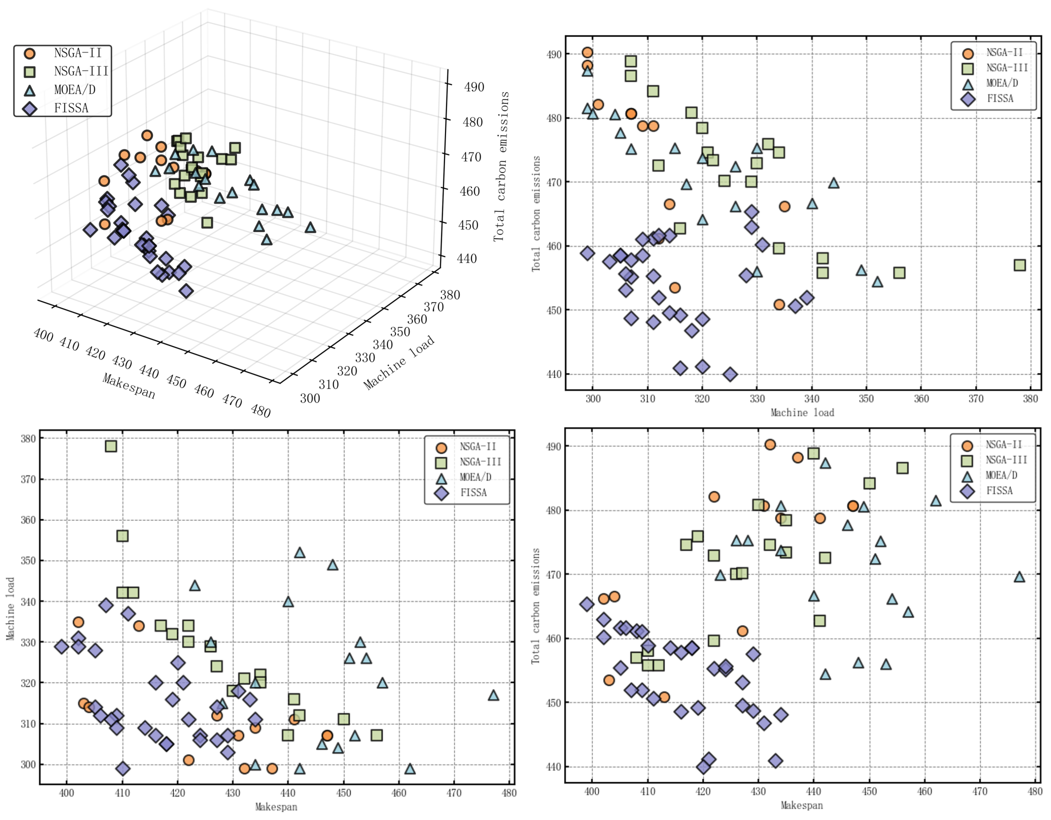


Figure 5: Pareto front of MK09.

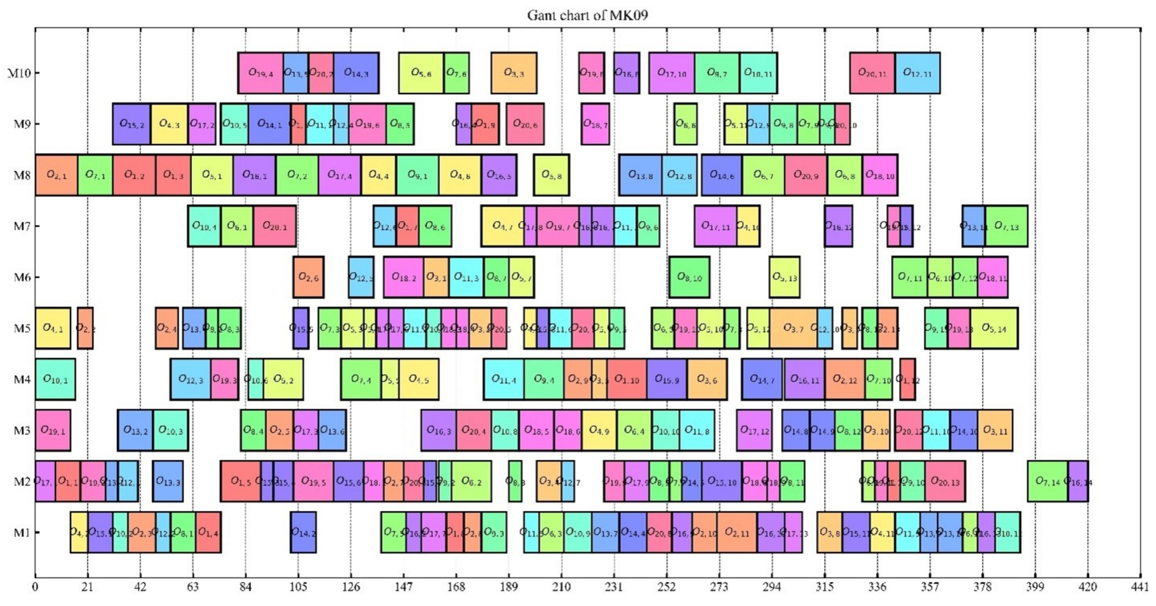


Figure 6: Gant chart of MK09.

Q. Deng, Q. Kang, L. Zhang, M. Zhou, and J. An. Objective space-based population generation to accelerate evolutionary algorithms for large-scale many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 27:326–340, 2022. doi: 10.1109/TEVC.2022.3166815.

H. Ding and X. Gu. Improved particle swarm optimization algorithm based on novel encoding and decoding schemes for flexible job shop scheduling problem. *Computers & Operations Research*, 121:104951, 2020. doi: 10.1016/j.cor.2020.104951.

Y. Du, H. Yuan, K. Jia, and F. Li. Research on threshold segmentation method of two-dimensional otsu image based on improved sparrow search algorithm. *IEEE Access*, 11:70459–70469, 2023. doi: 10.1109/ACCESS.2023.3293191.

X. Z. Guo and X. L. Xin. Partial entropy and relative entropy of fuzzy sets. *Fuzzy System & Mathematics*, 19:97–102, 2005.

X. Kong, Y. Yao, W. Yang, Z. Yang, and J. Su. Solving the flexible job shop scheduling problem using a discrete improved grey wolf optimization algorithm. *Machines*, 10(11):1100, 2022. doi: 10.3390/machines10111100.

D. Lei, M. Li, and L. Wang. A two-phase meta-heuristic for multiobjective flexible job shop scheduling problem with total energy consumption threshold. *IEEE Transactions on Cybernetics*, 49: 1097–1109, 2018. doi: 10.1109/TCYB.2018.2796119.

Z. Li, C. Zhao, G. Zhang, D. Zhu, and L. Cui. Multi-strategy improved sparrow search algorithm for job shop scheduling problem. *Cluster Computing*, 27:4605–4619, 2024. doi: 10.1007/s10586-023-04200-w.

X. Long, J. Zhang, X. Qi, W. Xu, T. Jin, and K. Zhou. A self-learning artificial bee colony algorithm based on reinforcement learning for a flexible job-shop scheduling problem. *Concurrency and Computation: Practice and Experience*, 34:e6658, 2022. doi: 10.1002/cpe.6658.

F. Ma, W. Sun, Z. Jiang, S. Suo, X. Wang, and Y. Liu. Industrial robot trajectory optimization based on improved sparrow search algorithm. *Machines*, 12(7):490, 2024. doi: 10.3390/machines12070490.

H. Ma, Y. Zhang, S. Sun, T. Liu, and Y. Shan. A comprehensive survey on nsga-ii for multi-objective optimization and applications. *Artificial Intelligence Review*, 56:15217–15270, 2023. doi: 10.1007/s10462-023-10526-z.

J. Shi, W. Liu, and J. Yang. An enhanced multi-objective evolutionary algorithm with reinforcement learning for energy-efficient scheduling in the flexible job shop. *Processes*, 12(9):1976, 2024. doi: 10.3390/pr12091976.

Y. Xu, M. Zhang, M. Yang, and D. Wang. Hybrid quantum particle swarm optimization and variable neighborhood search for flexible job-shop scheduling problem. *Journal of Manufacturing Systems*, 73:334–348, 2024. doi: 10.1016/j.jmsy.2024.02.007.

J. Xue and B. Shen. A novel swarm intelligence optimization approach: sparrow search algorithm. *Systems Science & Control Engineering*, 8:22–34, 2020. doi: 10.1080/21642583.2019.1708830.

G. Zhang, L. Gao, and Y. Shi. An effective genetic algorithm for the flexible job-shop scheduling problem. *Expert Systems with Applications*, 38:3563–3573, 2011. doi: 10.1016/j.eswa.2010.08.145.

J. Zhang, X. Zhu, and J. Li. Intelligent path planning with an improved sparrow search algorithm for workshop uav inspection. *Sensors*, 24(4):1104, 2024. doi: 10.3390/s24041104.

Q. Zhang and H. Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11:712–731, 2007. doi: 10.1109/TEVC.2007.892759.

H. Zhu, Q. Deng, L. Zhang, X. Hu, and W. Lin. Low carbon flexible job shop scheduling problem considering worker learning using a memetic algorithm. *Optimization and Engineering*, 21:1691–1716, 2020. doi: 10.1007/s11081-020-09494-y.

T. Zhu, X. Liu, X. Wang, and H. He. Technical development and prospect for collaborative reduction of pollution and carbon emissions from iron and steel industry in china. *Engineering*, 31:37–49, 2023. doi: 10.1016/j.eng.2023.02.014.