Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.045826

ARTICLE

An Improved Harris Hawk Optimization Algorithm for Flexible Job Shop Scheduling Problem

Zhaolin Lv[1], Yuexia Zhao[2], Hongyue Kang[3,*], Zhenyu Gao[3] and Yuhang Qin[4]

[1]College of Systems Engineering, National University of Defense Technology, Changsha, 410073, China

[2]College of Information Management, Nanjing Agricultural University, Nanjing, 210031, China

[3]Department of Intelligent Supply Chain, JD Logistics, Beijing, 100076, China

[4]College of Intelligent Science and Technology, National University of Defense Technology, Changsha, 410073, China

*Corresponding Author: Hongyue Kang. Email: kanghongyue1@jd.com

Received: 08 September 2023 Accepted: 04 December 2023 Published: 27 February 2024

ABSTRACT

Flexible job shop scheduling problem (FJSP) is the core decision-making problem of intelligent manufacturing production management. The Harris hawk optimization (HHO) algorithm, as a typical metaheuristic algorithm, has been widely employed to solve scheduling problems. However, HHO suffers from premature convergence when solving NP-hard problems. Therefore, this paper proposes an improved HHO algorithm (GNHHO) to solve the FJSP. GNHHO introduces an elitism strategy, a chaotic mechanism, a nonlinear escaping energy update strategy, and a Gaussian random walk strategy to prevent premature convergence. A flexible job shop scheduling model is constructed, and the static and dynamic FJSP is investigated to minimize the makespan. This paper chooses a two-segment encoding mode based on the job and the machine of the FJSP. To verify the effectiveness of GNHHO, this study tests it in 23 benchmark functions, 10 standard job shop scheduling problems (JSPs), and 5 standard FJSPs. Besides, this study collects data from an agricultural company and uses the GNHHO algorithm to optimize the company's FJSP. The optimized scheduling scheme demonstrates significant improvements in makespan, with an advancement of 28.16% for static scheduling and 35.63% for dynamic scheduling. Moreover, it achieves an average increase of 21.50% in the on-time order delivery rate. The results demonstrate that the performance of the GNHHO algorithm in solving FJSP is superior to some existing algorithms.

KEYWORDS

Flexible job shop scheduling; improved Harris hawk optimization algorithm (GNHHO); premature convergence; maximum completion time (makespan)

1 Introduction

Consumers' demand patterns are changing due to the rapid development of globalization and a shift in customer consumption consciousness. The demand of the whole market has gradually evolved from the previous state of few varieties and large quantities to a state where customers require small and medium-sized batches, more variety, high product quality, and short delivery time requirements [1]. There is an urgent need for manufacturing enterprises to achieve high efficiency, high

quality, and flexibility and enhance enterprise competitiveness with low-cost manufacturing. With the advancement of technology, information technology has brought new convenience to the development of companies [2,3]. Intelligent manufacturing is the primary trend and key to the development of the manufacturing industry [4]. Job shop scheduling problem (JSP) is a core decision-making problem of intelligent manufacturing production management. The flexible job shop scheduling problem (FJSP) is an extension of the classical JSP which comprises two sub-problems: machine assignment and operation sequencing. Machine assignment involves selecting a machine from a candidate set for each operation, while operation sequencing deals with scheduling all operations on all machines to create satisfactory schedules. The FJSP is a classical NP-hard problem whose application fields are extremely broad [5]. The scheduling scheme optimization aims to reduce the maximum completion time (makespan) [6] or reduce the production cost. An efficient scheduling scheme can shorten the production cycle, reduce production costs, and facilitate adaptation to complex markets [7].

With the improvement of intelligence level, companies need to change their traditional ways of operating and implement smart manufacturing, including intelligent production processes and decision-making [8]. Therefore, it is essential to use intelligent algorithms to solve the FJSP. Some studies have utilized intelligent algorithms to solve the FJSP and its variants. Zhang et al. [9] solved the reconfigurable distributed flowshop group scheduling problem using an iterated F-Race algorithm. Experimental results demonstrate significant advancement in solution accuracy and efficiency compared to other heuristics. Lu et al. [10] tackled the energy-efficient scheduling of distributed flow shops with heterogeneous factories, considering permutation and hybrid flow shops. A hybrid multi-objective optimization algorithm is proposed and it outperforms six advanced benchmark algorithms in a real-world example. The Harris hawk optimization (HHO) [11] algorithm is a novel bionic swarm intelligence optimization algorithm inspired by hawks' predatory behavior. Due to its fewer parameters, high efficiency, and ease of implementation, it has captured the attention of many researchers and has been used in several fields [12]. Several literatures have used the HHO algorithm to solve the JSP. Liu [13] proposed an enhanced HHO algorithm (IHHO) for the shortcomings of HHO, which has blind search during the global search. Experimental results for the JSP showed that IHHO outperformed other compared algorithms. Choo et al. [14] proposed an integrated HHO (EN-HHO) to solve the multi-objective flexible job shop scheduling problem (MOFJSP). Jouhari et al. [15] used the modified HHO (MHHO) to solve the problem of unrelated parallel machine scheduling. Amer et al. [16] proposed an improved elite learning HHO (ELHHO) to address the multi-objective scheduling problem, and the experimental results demonstrated that the proposed ELHHO method could yield better results than other algorithms. Attiya et al. [17] proposed an improved HHO algorithm based on simulated annealing (SA) called HHOSA for job scheduling in a cloud environment. Experimental results showed that the proposed HHOSA can significantly reduce the makespan of job scheduling. However, these researchers do not apply the HHO algorithm to solve the classical FJSP and do not consider its application in the real manufacturing process. Furthermore, the current research on the HHO algorithm has found that it has the same dilemma as other intelligent algorithms, that is, it will appear a premature convergence problem when solving some more complex issues [18–20]. Additionally, since global exploration is only conducted in the first half of the iteration, it is challenging to balance the capabilities of global exploration and local exploitation using the escaping energy of prey. Table 1 compares related work on job shop scheduling to highlight the differences between our work and the existing research.

In this paper, we propose an improved Harris hawk optimization algorithm (GNHHO) to solve the FJSP, which introduces an elitism strategy, a chaotic mechanism, a nonlinear escaping energy update strategy, and a Gaussian random walk strategy to prevent premature convergence. Specifically,

GNHHO prevents the algorithm from premature convergence by replacing the optimal solution, adding a perturbation to the parameter r, proposing a nonlinear method to update the escaping energy E, and using a Gaussian random walk strategy to generate new individuals. Furthermore, this paper utilizes the tent mapping mechanism to add perturbation for the parameter r and introduces a cosine function to adjust the step size of the Gaussian random walk. Our objective is to minimize the makespan and we choose a two-segment encoding mode based on the job and the machine of the FJSP. By comparing the proposed GNHHO algorithm with the HHO algorithm on 23 functions, we verify the effectiveness of the GNHHO algorithm. This study then uses the GNHHO algorithm to solve the 10 standard JSPs and 5 Brandimarte FJSP standard examples. The results demonstrate that the performance of the GNHHO algorithm in solving JSP and FJSP is superior to some existing algorithms. We then collect data from an agricultural company and use the GNHHO algorithm to optimize the company's job shop scheduling whose objective is to minimize makespan. The company has the characteristics of multiple varieties, small batches, and flexible working. The production job shop of the company is a flexible job shop, which is in line with the FJSP that we aim to solve through the proposed method. In addition, the actual data of the company can be used to deal with some special cases of scheduling problems, and the change of order delivery rate can be further analyzed. The final results are expressed in the form of a Gantt chart [21]. By comparing the scheduling schemes before and after optimization, the optimized scheduling scheme could advance the makespan by 28.16% and 35.63% for static and dynamic scheduling, respectively, with an average increase of 21.50% in the order delivery rate on time.

Table 1: Comparison of related works on job shop scheduling

Reference	Algorithm	Application field	Objective
Liu (2021) [13]	IHHO	JSP	Min makespan
Choo et al. (2022) [14]	EN-HHO	MOFJSP	Min makespan
Jouhari et al. (2020) [15]	MHHO	Parallel machine scheduling	Min makespan
Attiya et al. (2020) [17]	HHOSA	JSP	Min makespan
Our work	GNHHO	FJSP	Min makespan

The main contributions of this paper are as follows:

(1) This paper addresses a real-world enterprise problem where static scheduling orders have low delivery rates and dynamic scheduling cannot respond to the change of customer demands in time.

(2) This paper proposes a GNHHO algorithm that introduces an elitism strategy, a chaotic mechanism, a nonlinear escaping energy update strategy, and a Gaussian random walk strategy to prevent premature convergence.

(3) The experimental results indicate that our proposed GNHHO algorithm is better than some existing algorithms. It can advance the makespan by 28.16% and 35.63% for static and dynamic scheduling, respectively while also increasing the order delivery rate by 21.50%.

The paper is organized as follows. Section 2 summarizes the research methodology. Section 3 demonstrates an improved HHO algorithm for FJSP. Section 4 shows the experiment and analysis. The paper is concluded in Section 5.

2 Research Methodology

This section describes the constructed flexible job shop scheduling model. Section 2.1 introduces the FJSP. Section 2.2 provides a detailed description of the model.

2.1 Problem Description

For a job shop in production job shops, there are N jobs $\{J_1, J_2, \cdots J_i, \cdots, J_N\}$ to be processed, and there are M machines in a production workshop. Job i contains a_i operations. The data information we can learn is the time of each production operation. We need to choose different machines for these operations and determine the start time of the processing to achieve the optimal goal, which means that it minimizes makespan. For the FJSP, some basic assumptions should be satisfied:

(1) Before the operation behind the same jobs starts, all the previous operations have been processed.

(2) There is no specific requirement for the processing sequence of different jobs.

The meanings of notations to be used in this paper are shown in Table 2.

Table 2: Meaning of the notation

Notation	Meaning
i	Job index
m	Machine index
j	Operation index
a_i	No. of operations of the job i
N	No. of jobs
M	No. of machines in a job shop
T_{max}	The maximum makespan of all jobs
J_n	The nth scheduling job
P_{ijm}	The processing time of the operation j of the job i by machine m
S_{ij}	The start time of the operation j of the job i
T_{ij}	The end time of the operation j of the job i
X_{ijm} $Y_{i(j+1)hlm}$	0-1 variable
Z	A large positive number

2.2 Modeling

This study selects the model's objective function to minimize the makespan, ensuring that a job shop can complete its task as quickly as possible. The objective function is defined by Eq. (1).

$$T = \min\left(\max_{1\le i\le N} T_{iai}\right) = \min T_{\max} \tag{1}$$

where T is the corresponding time when the processing of job i is completed. The constraints are as follows:

s.t.

$$T_{ia_i} \le T_{max} \tag{2}$$

$$T_{ij} \geq S_{ij} + P_{ijm} \times X_{ijm} \tag{3}$$

$$S_{ij} + P_{ijm} \leq S_{hl} + Z\left(1 - y_{ijhlm}\right) \tag{4}$$

$$T_{ij} \leq S_{i(j+1)} + Z\left(1 - y_{hli(j+1)m}\right) \tag{5}$$

$$\sum_{i=1}^{N} \sum_{j=1}^{a_i} Y_{ijhlm} = X_{hlm} \tag{6}$$

$$\sum_{m=1}^{M} X_{ijm} = 1 \tag{7}$$

$$S_{ij+1} \geq T_{ij} \tag{8}$$

$$S_{ij} \geq 0 \tag{9}$$

$$T_{ij} \geq 0 \tag{10}$$

Eq. (2) represents the makespan of any job must be less than or equal to the makespan of all jobs. Eq. (3) is the constraint on processing operation. Specifically, X_{ijm} is a 0–1 variable that is 1 if operation j of job i is processed on machine m, and 0 otherwise. Eqs. (4) and (5) symbolize when processing jobs, only one operation can be performed simultaneously on the same equipment. Specifically, $Y_{i(j+1)hlm}$ is a 0–1 variable that is 1 if the operation j of job i precedes the operation l of job h on machine m. Eq. (6) limits that an operation can only be processed once by a machine. Eq. (7) represents an operation that can only choose one machine to process. Eq. (8) represents the sequence constraint of the operations. Eqs. (9) and (10) indicate that the parameter variable must be no less than 0.

3 Improved HHO Algorithm for Flexible Job Shop Scheduling Problem

This section describes the improved HHO algorithm for the FJSP. The HHO algorithm is presented in Section 3.1, and an improved algorithm is proposed in Section 3.2. The proposed GNHHO algorithm is designed to solve the FJSP in Section 3.3.

3.1 HHO Algorithm

Harris hawks can switch various chasing modes according to the dynamic changes of the capture environment they face and the escape mode of the prey [22]. Based on this, the HHO algorithm is developed by mathematically simulating this behavior and the raiding method.

1. Exploration stage

In the algorithm, the hawk's position is determined by two different methods based on the locations of the other hawks and the rabbit at the time of the kill, with equal probability. It can be defined by Eq. (11):

$$X\left(t+1\right) = \begin{cases} X_{rand}\left(t\right) - r_1 \left| X_{rand}\left(t\right) - 2r_2 X\left(t\right) \right|, & q \geq 0.5 \\ \left(X_{rabbit}\left(t\right) - X_m\left(\mathrm{t}\right)\right) - r_3\left(lb + r_4\left(ub - lb\right)\right), & q < 0.5 \end{cases} \tag{11}$$

where $X(t+1)$ is the position of the hawk after the t iterations and $X(t)$ is the location of the hawk at the moment t. $X_{rabbit}(t)$ represents the location of the rabbit, and r_1, r_2, r_3, r_4 and q are random numbers whose ranges are (0, 1). The variables' range is determined by the values of lb and ub, and $X_{rand}(t)$ is the randomly chosen hawk position. $X_m\left(t\right) = 1/Q \sum_{i=1}^{Q} X_i\left(t\right)$, where X_m is the average position of the current hawk group and X_i represents the position of each hawk. Q denotes the total number of hawks.

2. The transition between exploration and exploitation

In the process of running away, the rabbit's energy will be significantly reduced. We call it the escaping energy of the prey E. It can be defined by Eq. (12):

$$E = 2E_0\left(1 - \frac{t}{T}\right) \tag{12}$$

where E_0 is the initial value of energy, which is a random number, and the range is $(-1, 1)$. T is the maximum number of iterations. The exploration stage is carried out at $|E| \geq 1$.

3. Exploitation stage

At this stage, the Harris hawk has concluded the exploration stage and has detected the location of the target prey (rabbit). The HHO lists four possible scenarios depending on how the prey escapes and the hawk hunts. Assuming that r is the probability of determining whether the prey can escape, $r < 0.5$ means the rabbit can avoid the hawk's capture. Otherwise, the prey cannot escape. When $|E| \geq 0.5$ and $r \geq 0.5$, the Harris hawks only need to surround the rabbit to capture it softly. When $|E| < 0.5$ but $r \geq 0.5$, the Harris hawks directly adopt a hard besiege raid on the rabbit. When $|E| \geq 0.5$ but $r < 0.5$, the Harris hawks need to employ a wittier soft surrounding with progressive quick dives. When $|E| < 0.5$ and $r < 0.5$, the Harris hawks will take the hard besiege with a progressive rapid dive strategy.

3.2 Improved HHO Algorithm

As an intelligent algorithm, the HHO algorithm requires fewer parameters than other algorithms and possesses strong global exploration capabilities. However, it may also encounter the problem of premature convergence when dealing with more complex issues. Therefore, this paper takes some measures to improve the HHO algorithm to solve the problem of premature convergence.

1. Introducing an Elitism Strategy

In order to improve the global search ability of the algorithm and avoid the decrease of population diversity in the later stages of iteration, an elitism strategy (elitist preservation strategy) [23] is introduced. Considering enhanced communication between the optimal and suboptimal solutions during the iteration, we choose two optimal positions to replace the optimal solution which can better guide the hawks to prey.

$$X_{rabbit} = \sum_{j=\alpha,\beta} \frac{f\left(X_{jbest}\left(t\right)\right)}{\sum_{z=\alpha,\beta} f\left(X_{zbest}\left(t\right)\right)} X\left(t\right) \tag{13}$$

where X_{jbest} and $f(X_{jbest})$ mean the dominant individual and its fitness value.

2. Introducing a Chaotic Mechanism

The tent mapping system [24] has the characteristics of randomness and regularity which makes the algorithm easy to jump out of local optima, so it is often used to increase the perturbation of the algorithm or generate the initial population. Fig. 1 depicts the scatter plot and the sequence distribution histogram of the tent mapping, in which the maximum number of iterations is 500. From Fig. 1, it can be seen that the tent mapping is random and evenly distributed. Therefore, this paper uses this tent mapping mechanism to add perturbation to the parameter r in HHO to make its change more volatile.

The formula for calculating r is as follows:

$$r_{i+1} = \begin{cases} r_i/0.6, & x_i < 0.6 \\ (5/2) \times (1 - r_i), & x_i \geq 0.6 \end{cases} \tag{14}$$

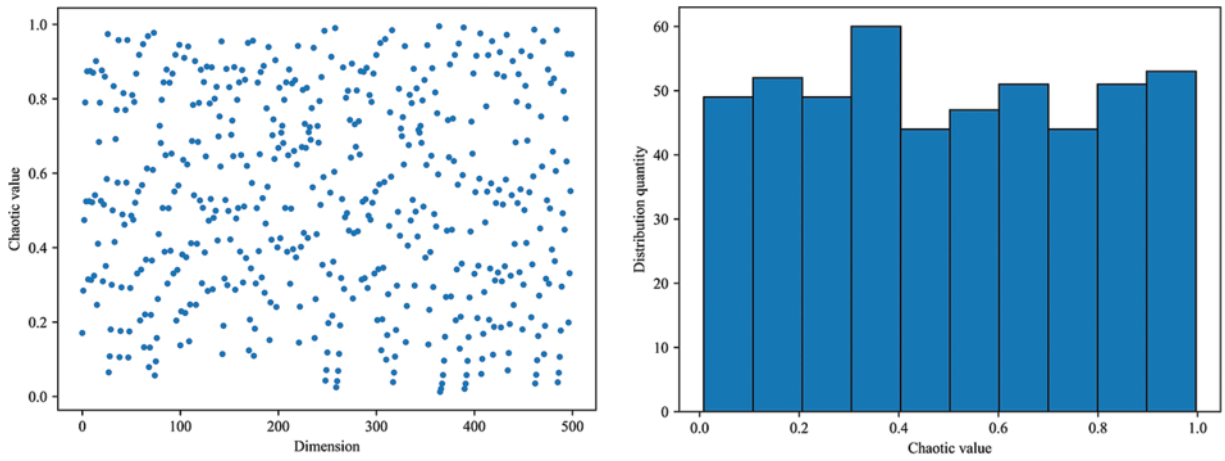


Figure 1: The scatter plot and the histogram of the tent mapping

3. Introducing a Nonlinear Escaping Energy Update Strategy

In the HHO algorithm, the transition of the algorithm from the exploration stage to the exploitation stage is controlled by the E. However, the update of E changes linearly, which means that if the algorithm enters the second half of the iteration, it will only perform local development and search. At this point, the algorithm easily falls into the local optimization. Therefore, a nonlinear method is proposed to update the escaping energy E.

$$E1 = 2 \times E\left(1 - \frac{t}{T}\right) \times \sin\left(\left(3k + \frac{1}{4}\right)\pi \frac{t}{T}\right) \tag{15}$$

where k is related to the number of decreasing cycles of the escaping energy. In our study, it is determined by grid search, which is searched in the scope of [0,15]. Experimental results demonstrate that the algorithm achieves optimal exploration and exploitation performance when $k = 5$. And when the value of k was set to 5, it generated the most optimal solutions and exhibited superior convergence speed when solving various benchmark functions. Therefore, it is concluded that the optimal value for the parameter k would be 5. To provide a clearer visualization of the effect of the proposed nonlinear escaping energy update strategy, Fig. 2 displays the relationship between the escaping energy and the number of iterations for HHO and GNHHO. Fig. 2 shows that compared with the linear variation of the escaping energy E, the periodic trend of $E1$ towards 0 reflects that the hawk is approaching and capturing the prey according to the probability. Therefore, this enables multiple rounds of global and local optimization searches.

4. Introducing a Gaussian Random Walk Strategy

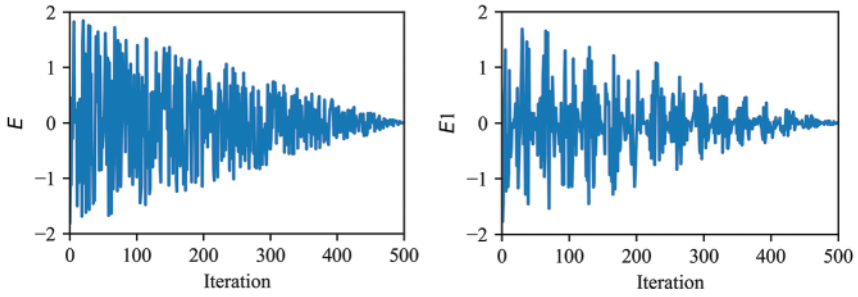
During successive iterations of the algorithm, we consider that if the computation finds that the mean value of the dominant population has not changed twice, and this happens continuously, then we determine that the algorithm is stalled at this iteration. To overcome the issue of falling into stagnation and improve exploration abilities, we utilize the Gaussian random walk strategy [25] to generate new individuals. The random walk strategy introduces a probabilistic model that imitates the random movement of organisms in nature. It has been widely utilized in the design and enhancement of several optimization algorithms. It can efficiently address the challenge of falling into local optima

by injecting randomness and diversifying the search process.

$$X(t+1) = Gaussian(X(t), \sigma) \tag{16}$$

$$\sigma = \cos\left(\pi/2 \times (t/T)^2\right) \times (X(t) - X^*(t)) \tag{17}$$

where $X^*(t)$ is a randomly selected individual from the dominant population, and a cosine function continuously adjusts the random step size. In this way, it can be realized that relatively large disturbances can be generated in the population in the early stage, and the disturbances in the later stage will be reduced.



(a) Escaping energy for HHO (b) Escaping energy for GNHHO

Figure 2: Comparison of escaping energy for HHO and GNHHO

In general, we adopt four strategies which introduce an elitism strategy, a chaotic mechanism, a nonlinear escaping energy update strategy, and a Gaussian random walk strategy to improve the HHO algorithm, and the pseudo-code of the improved HHO algorithm is displayed in the Algorithm 1.

3.3 GNHHO for the Flexible Job Shop Scheduling Problem

3.3.1 Encoding and Decoding of Algorithm

The original HHO algorithm was initially developed for tackling successional optimization problems, while the FJSP represents a discrete optimization problem. Therefore, it becomes crucial to establish a transitional mechanism between the HHO and the FJSP. In this paper, we introduce a two-segment encoding and decoding mechanism to solve this problem. The coding strategy adopted in this paper is a two-segment coding method that considers the operation and machine selection. It represents a set of feasible solutions with $X = [X_g|X_j]$, where the vector X_g represents the selection method of the machine and the vector X_j represents the processing sequence of the operation. The position vector of the Harris hawks in the HHO algorithm is expressed by a $2l$-dimensional vector of real numbers, $X = \{x_1, x_2, \cdots, x_l, x_{l+1}, \cdots, x_{2l}\}$, where l is the number of all operations in the FJSP to be solved, and the element X of the vector is the real number between $[-N, N]$. The two-segment encoding method is shown in Table 3.

Algorithm 1: Pseudo-code of GNHHO

Input: The population size Q and maximum number of iterations T
Output: The location of the best solution and its fitness value
Initialize the population;
Let iter = 0;
While iter $< T$ **do**

(Continued)

Algorithm 1 (continued)

```
    if The algorithm stagnates then
        Update hawk position by Eq. (16);
    else
        Calculate the fitness function and locate the rabbit by Eq. (13);
    end
    for each hawk do
        Update E1 by Eq. (15) and calculate the modified r by Eq. (14);
        if |E1| ≥ 1 then
            In the exploration stage, update the hawk position by Eq. (11);
        end
        if |E1| < 1 then
            if r ≥ 0.5 and |E1| ≥ 0.5 then
                Update the hawk position by soft besiege;
            else
                if r ≥ 0.5 and |E1| < 0.5 then
                    Update the hawk position by hard besiege;
                else
                    if r < 0.5 and |E1| ≥ 0.5 then
                        Update the hawk position by soft besiege with rapid dives;
                    else
                        Update the hawk position by hard besiege with progressive rapid dives;
                    end
                end
            end
        end
    end
end
Return the best global solution founded;
```

Table 3 Meaning of two-segment encoding mode

Machine selection X_g						Operation sequence X_j					
2	1	2	3	1	3	1	2	3	2	1	3
M_2	M_1	M_2	M_3	M_1	M_3	O_{11}	O_{21}	O_{31}	O_{22}	O_{12}	O_{32}

Suppose a job shop needs to process 3 jobs, each job has 2 operations, and there are 3 optional machines, then the total length of the position vector is 12, and the value range is between [−3, 3]. The first step is based on the selection of the machine, assuming that a set of codes that exist at this time is [2 1 2 3 1 3], then this corresponds to the operation sequence and is selected for processing on the corresponding machine. At this time, this group of codes means that the machine selection order is machine 2, machine 1 ⋯, and machine 3 for the operation sequence $\{O_{11}\ O_{12}\ O_{21}\ O_{22}\ O_{31}\ O_{32}\}$. Then, the encoding method is to encode at the operation level. For example, it can be assumed that a code is [1 2 3 2 1 3] based on the operation. The number represents the serial number of the job to be processed, and the number of digits in this group of codes represents the number of all operations in all the jobs.

The sequence of the numbers represents the order in which the machine operations these operations, and the length of the code represents the total number of operations after the decomposition of the order. Taking this code as an example, it is 1, 2, 3, and means that there are 3 jobs, of which there are 6 numbers, representing 6 operations that need to be processed. That is, the processing sequence of the operation is operation 1 for job 1, operation 1 for job 2 ⋯, and operation 2 for job 3. This is a feasible set of operation scheduling schemes.

Then, we need to convert the given position vector X to the machine selection (MS) vector X_g and the operation sequence (OS) vector X_j. We adopted the method of [26] and used Eq. (18) to convert individual position vector X into corresponding numbers, which could be selected and assigned by machine sets. It can be defined by Eq. (18):

$$y_j = round\left(\frac{1}{2N}\left(x_j + N\right)\left(s_j - 1\right) + 1\right), 1 \le j \le l \tag{18}$$

where $round(x)$ is the function that sets x to the nearest integer and y_j represents the serial number of the machine assigned by operation j in the machine set of available machines for operation j and. Then, x_j is the vector value of j and s_j represents the number of available machines for operation j.

In the conversion of the process sequence vector, first place the elements of the position vector X_j in the process order $\{O_{i1}\ O_{i2}, \cdots, O_{ij}\}$. Then sort in descending order by the size of the element value, and replace the number of each operation with the job number it belongs to. Fig. 3 is an example of converting a position vector X_j into an operation sequence vector.

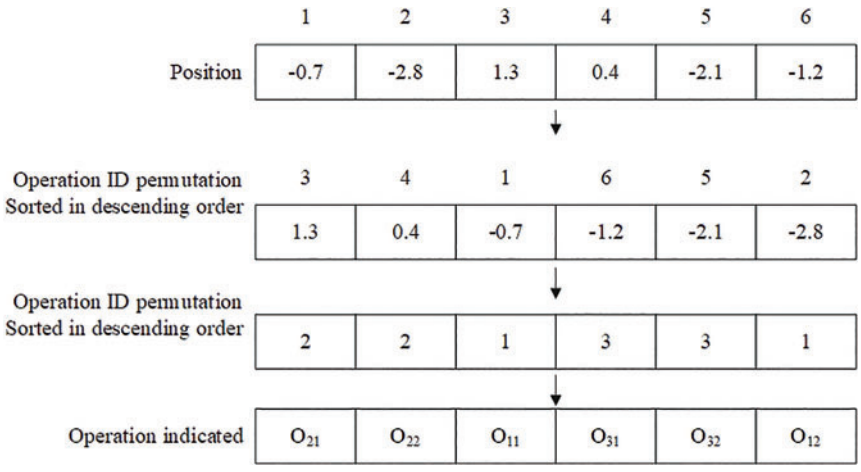


Figure 3: The conversion from the position vector to the operation sequence vector

The decoding part can be understood as the inverse operation of encoding. The inversion of Eq. (18) can be used to calculate the transformation of position vectors to scheduling schemes. It means that the real value is converted into an integer value, and then calculate the objective function value for decoding. When the individual position vector is converted into a scheduling scheme, the Ranked Order Value (ROV) rule [27] is used to rank operations. First, each individual is sorted according to the ascending order of the position vector values. In this way, the ROV value is determined, and the sequence of operations is obtained according to the corresponding ROV value. Table 4 shows the correspondence between individual location information and operation sequence.

Table 4: The relationship between location information and operation sequence

Operation number	1	1	2	2	3	3
Position	−0.7	−2.8	1.3	0.4	−2.1	−1.2
ROV	4	1	6	5	2	3
OS	2	1	3	3	1	2

3.3.2 Population Initialization

The effectiveness of the initial swarm directly impacts the optimization result of the metaheuristic algorithm. To enhance the algorithm's capability, specific strategies can be employed to improve the original swarm. In this paper, a two-segment coding mechanism is applied, which involves separate swarm initialization for the machine selection and operation sequence segments. For the initialization of machine selection, two different machine selection rules are adopted, global selection [28] and random selection. Global selection can reduce the makespan and improve the load balance of machines. The specific method is to set an array of the same length as the number of machines, and each value of the array corresponds to the processing time of the corresponding machine. The proportion of using the two machine rules is 0.7 and 0.3. Next, randomly select a job, starting from the first operation, and add the processing time of the selected machine for the current operation to the corresponding position in the array. Then select the machine with the shortest processing time as the processing machine and keep the time at the corresponding position in the array. Repeat similar operations until the machine selection for all operations is complete. The purpose of random selection is to maintain the diversity of the initial solutions of the population. For the initialization of the operation sequence, a set of operation sequence schemes is generated randomly. Finally, these schemes are then combined with the machine selection segment to produce various scheduling schemes.

3.3.3 Local Search Strategy

In this paper, when a scheduling scheme is obtained, we can use the neighborhood structure as a local search strategy to seek a better one close to the optimal one to prevent local optima [29]. This study performs the local search in the following four ways:

(1) For the neighborhood structure N1: randomly select two different operations in the existing scheduling scheme to exchange.

(2) For the neighborhood structure N2: randomly select two adjacent operations and exchange their sequences. Fig. 4 illustrates the neighborhood structure of N2.

(3) For the neighborhood structure N3: randomly select two different operations in the preferred scheduling plan and then invert all the operations contained between these two operations.

(4) For the neighborhood structure N4: randomly select a contiguous sequence of operations as a block, similarly select another contiguous sequence of operations as a block, and then switch the operations of these two blocks.

The specific strategy is to use the first way to obtain a new solution. The objective function value is calculated after the exchange, and compared with the objective function values before the exchange, leaving the solution with the smaller objective function value and updating it. After that, continue to implement the first way, and set a critical value as the basis for judgment if there is no better solution when more than the critical value, the next local search way is performed.

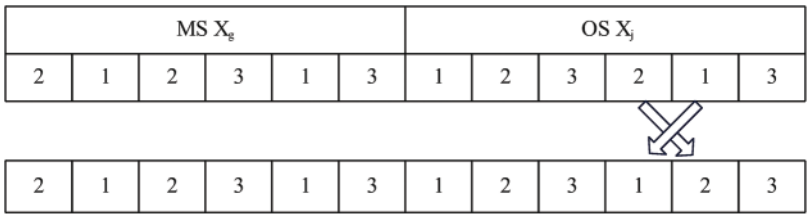


Figure 4: The neighborhood structure of N2

3.3.4 Algorithm Design for the Flexible Job Shop Scheduling Problem

After describing the algorithm's encoding and decoding approach and the local search strategy for the FJSP, we explained in detail how to use the proposed GNHHO algorithm to solve FJSP. In the GNHHO algorithm for the FJSP, the positions of the hawks denote the common candidate solutions, excluding the global optimum. The location of the prey represents the global optimal solution, which is the best scheduling scheme for FJSP. By employing the GNHHO algorithm, each candidate solution obtained represents a specific scheduling scheme. The objective is to obtain the optimal solution, which corresponds to the best flexible job shop scheduling scheme, by optimizing these candidate solutions using the GNHHO algorithm. The specific steps for solving FJSP using the proposed GNHHO algorithm are as follows:

Step 1. Set the parameters, including the number of jobs N, the number of machines M, and the maximum number of iterations T, and then generate a set of initial feasible solutions using the global selection and random selection.

Step 2. Generate the initial location of hawks and calculate the fitness value. Then decide the current best scheduling scheme by introduced elitism strategy.

Step 3. The value of the proposed nonlinear escaping energy $E1$ and introduced modified r is calculated, and the nonlinear escaping energy update strategy proposed by GNHHO is utilized to regulate global exploration and local exploitation.

Step 4. Select the processing sequence of the optimal operation obtained in the exploration process, and then search for a better solution (processing sequence) by four local search strategies.

Step 5. Assess the algorithm's stagnation and employ the introduced Gaussian random walk strategy to update the population.

Step 6. Determine whether the algorithm reaches the maximum number of iterations T. If it does, output the optimal scheduling scheme. Otherwise, repeat steps 2–5.

The flowchart of the GNHHO algorithm is presented in Fig. 5.

4 Experiment and Analysis

In this section, we use a set of benchmark datasets to evaluate the performance of the GNHHO. After that, we use GNHHO to solve the FJSP of company A. Some famous FJSP benchmarks are employed to verify the effectiveness of GNHHO in Section 4.1. Section 4.2 utilizes the GNHHO algorithm to address the FJSP of company A.

The experiments were run on a Windows 10 Ultimate 64-bit operating system PC and performed with MATLAB R2017a. To ensure the rationality and credibility of the experimental results, the

population size Q of 30 is uniformly adopted for all algorithms, while the maximum number of iterations T is set to 200. Other parameter settings of the algorithm are shown in Table 5.

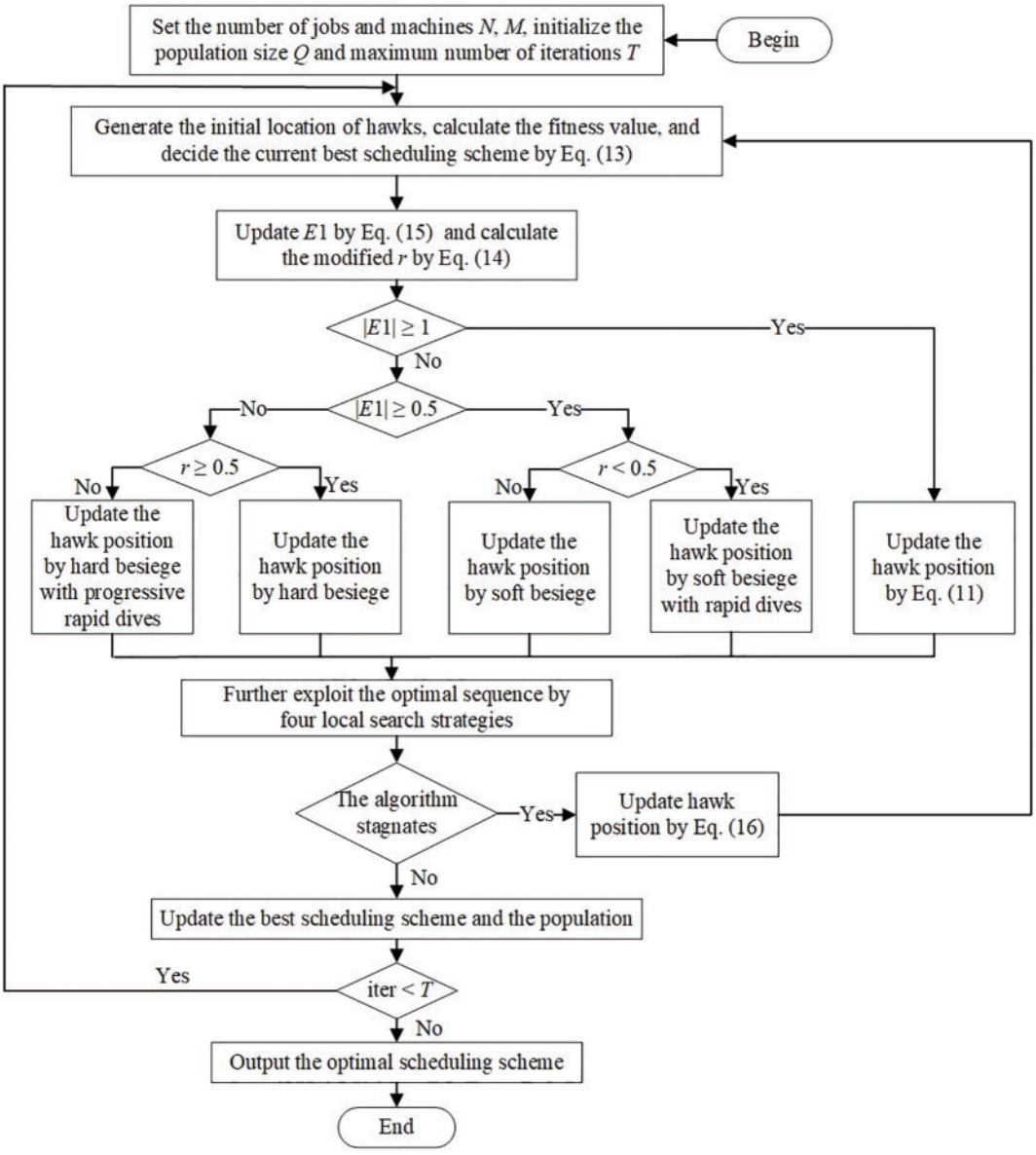


Figure 5: Flowchart of the GNHHO algorithm for FJSP

Table 5: Parameter settings

Algorithm	Parameter	Value
HHO	*No. of dimension*	30

(Continued)

Table 5 (continued)

Algorithm	Parameter	Value
GA	*genuine_cross*	0.1
	genuine_change	0.01
	No. of offspring per pair of parents	1
PSO	c_1	1.2
	c_2	1.2
	w	0.9
GWO	*Convergence parameter a*	[0,2]
WOA	*Convergence parameter a*	[0,2]
GNHHO	k	5
	No. of dimension	30

4.1 Tests of GNHHO Algorithm for Solving Scheduling Problems

This study uses HHO [30], GA [31], PSO [32], GWO [33], WOA [34] and GNHHO to solve the 10 examples from the standard test library JSPlib [35] and the 5 cases from the Brandimarte FJSP standard examples [36]. The parameter settings of each algorithm are shown in Table 5. The other experimental parameters Q and T are the same as in the previous experiment settings. We ran each algorithm 20 times independently. The results are shown in Tables 6 and 7, which list the example names, the problem sizes (N and M), the best-known solutions (BKS), the best results (Best), and the average results (Ave) obtained by existing algorithms and our proposed algorithm.

Based on the data presented in Tables 6 and 7, it can be observed that the GNHHO algorithm yields results closer to the actual value than other algorithms in 15 randomly selected computational examples. From Tables 6 and 7, we can see that the best and mean values of GNHHHO are better than those of other algorithms, which makes its optimization performance rank first. The value of the fitness in Fig. 6 represents the makespan of the experimental examples. From Fig. 6, we can see that the fitness of GNHHO is comparatively lower than other algorithms and has a faster convergence speed, indicating that GNHHO can quickly achieve a lower makespan. The reason is that the guidance of elite individuals can enhance the likelihood of GNHHO discovering superior candidate solutions. The nonlinear escaping energy updating strategy can effectively address the issue of transitioning into local search during the latter half of iterations. And the introduced Gaussian random walk strategy can mitigate the stagnation. The introduced four strategies can make the algorithm jump out of the local optimum and increase the ability of exploration and exploitation, which makes the algorithm faster to obtain better solutions. Also, the initial fitness value of GNHHO is 58, which is lower than other algorithms, showing that GNHHO is able to generate better initial solutions. This can be attributed to the incorporation of global selection and random selection methods for initialization in GNHHO, making it possible to find better initial solutions. However, the benchmark algorithms in our study have similar shortcomings, such as low convergence accuracy, falling into local optimization easily, and slow convergence speed leading to poor experimental results.

Table 6: Test results for the standard JSP examples

Example	*N*	*M*	BKS	Best						Ave					
				HHO	GA	PSO	GWO	WOA	GNHHO	HHO	GA	PSO	GWO	WOA	GNHHO
FT06	6	6	55	55	55	55	55	55	55	56.55	55.7	55.4	57.35	57.1	55
FT10	10	10	930	1023	1019	1016	1038	1042	964	1068.9	1034.2	1078.6	1061.8	1065.5	973.45
LA01	10	5	666	666	666	666	666	666	666	673.25	678.3	672.5	673.7	672.4	669.4
LA05	10	5	593	593	593	597	593	593	593	598.23	599.34	599.2	594.1	593.8	595.61
LA06	15	5	926	926	931	926	926	926	926	930.5	935.6	929.4	929.6	936.2	928.3
LA10	15	5	958	958	958	960	958	958	958	963.21	960.51	963.7	959.8	962.1	959.86
LA16	10	10	945	1029	1037	954	978	992	947	1032.34	1042.39	978.2	1026.4	1007.3	949.65
LA21	15	10	1046	1285	1299	1280	1269	1289	1136	1316.59	1325.63	1294	1296.3	1321.5	1174.6
LA25	15	10	977	1103	1112	1095	1008	1110	977	1249.58	1227.48	1203.8	1235.2	1239.7	986.21
LA36	15	15	1268	1546	1587	1481	1493	1491	1329	1576.8	1592.5	1389.7	1521.7	1519.6	1364.7

Table 7: Results for brandimarte examples

Example	N	M	Best						Ave					
			=HHO	GA	PSO	GWO	WOA	GNHHO	HHO	GA	PSO	GWO	WOA	GNHHO
MK01	10	6	41	42	41	42	42	40	43.1	44.6	43.3	45.2	45.7	42.4
MK04	15	8	57	55	63	68	66	49	59.2	57.6	68.2	71.5	69.1	53.1
MK07	20	5	171	168	157	184	154	151	181.6	174.5	168.9	193.7	158.3	157.9
MK09	20	10	328	332	331	381	368	315	343.8	347.1	365.2	420.4	389.0	329.1
MK10	20	15	259	247	262	337	253	231	271.4	258.2	274.1	355.6	267.9	249.8

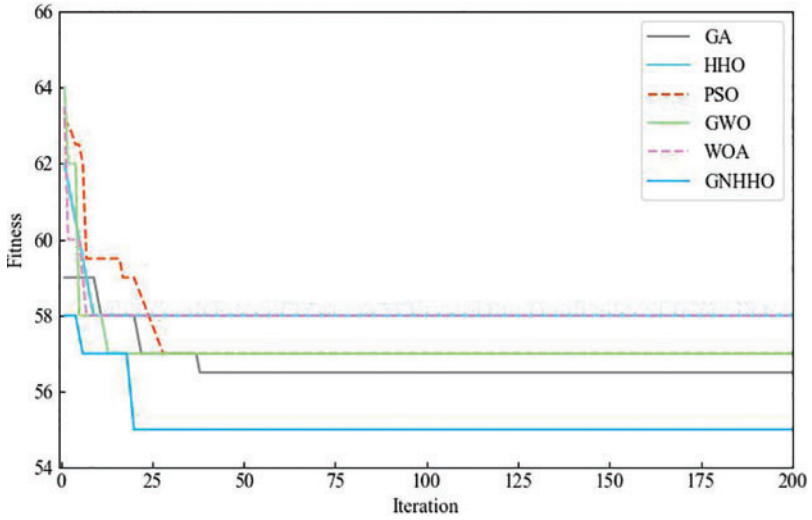


Figure 6: The convergence curves of FT06

Fig. 7 compares the makespan of HHO and GNHHO, taking MK04 as an example. It can be seen from Fig. 7 that GNHHO can find better solutions faster than HHO. In specific, the algorithm falls into a local optimum at the 6th and 13th iterations but is able to reach a lower makespan at the 12th and 16th. It demonstrates that the GNHHO algorithm can jump out of local optima compared with HHO because of the update strategy we introduced in HHO.

Fig. 8 shows the Gantt charts of the scheduling results solved by HHO and GNHHO. The horizontal coordinates of the Gantt chart represent time, while the vertical coordinates represent the machine. From Fig. 8, it is clear that GNHHO can produce better results than HHO. The makespan for this example has been reduced from 57 to 49. Therefore, it is clear that the GNHHO algorithm outperforms the HHO algorithm in handling the FJSP. This can reflect the better performance of the GNHHO algorithm in solving FJSP problems.

4.2 Solving Flexible Job Shop Scheduling Problems for Company A

4.2.1 Problem Analysis of Company A

Company *A* has the characteristics of multiple varieties, small batches, and flexible working. It can be found that the order on-time delivery rate of company *A* is at a relatively low level. Table 8 and

Fig. 9 show the order delivery rate of company *A* for each quarter of 2019–2021, where 19-1 is for the first quarter of 2019, and the other meanings are similar.

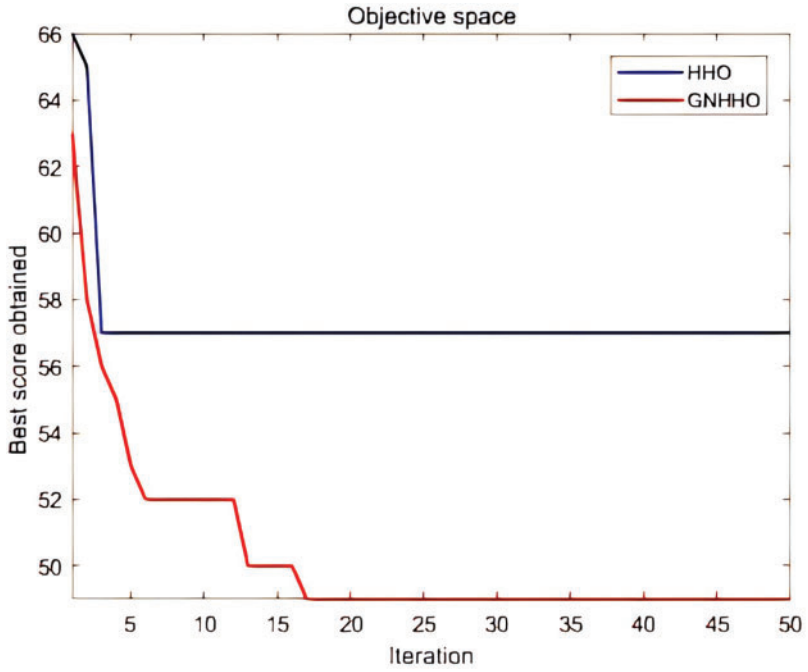
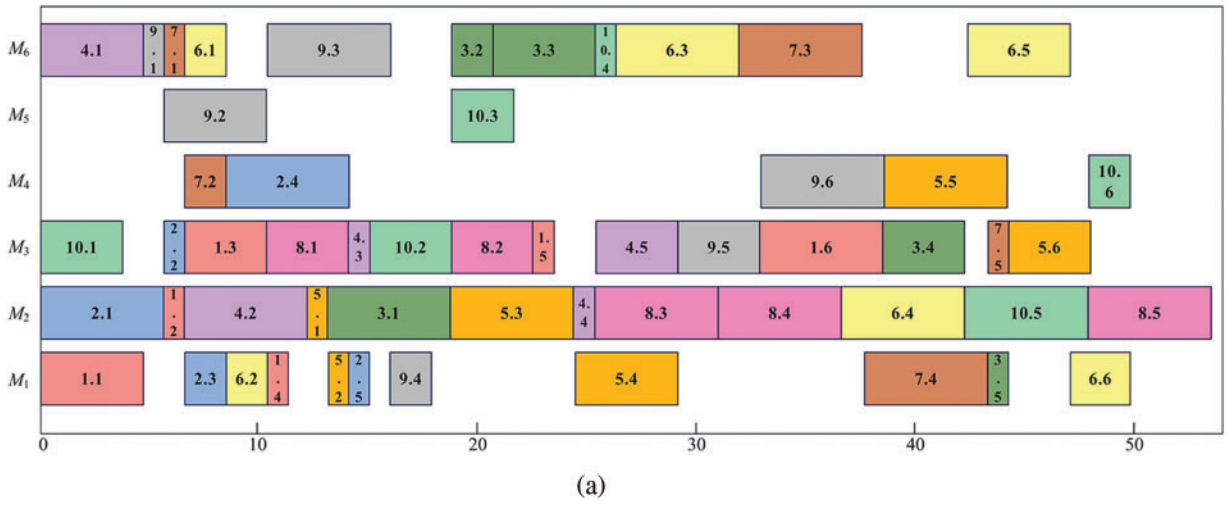
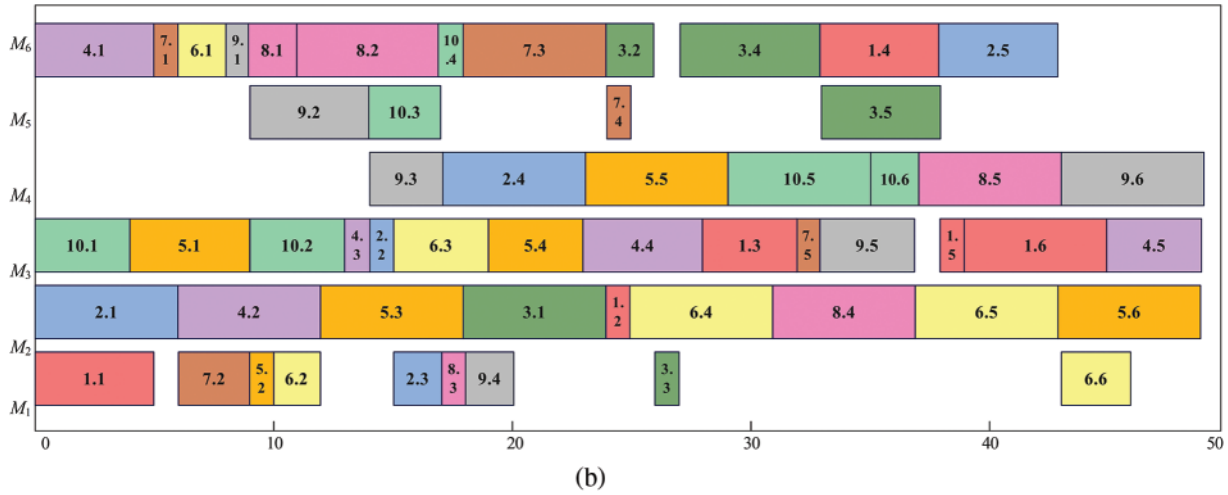


Figure 7: Comparison of HHO and GNHHO results



(a)

Figure 8: (Continued)



(b)

Figure 8: Flexible job shop scheduling problem for HHO and GNHHO

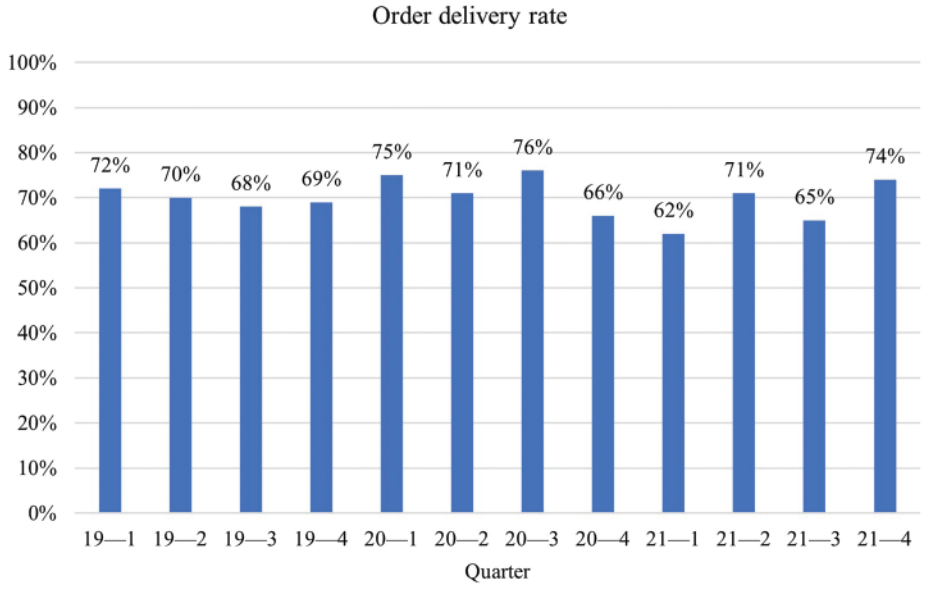


Figure 9: Order delivery rate of company *A* in 2019–2021 Quarterly

Table 8: Quarterly order delivery rate of company *A* in 2019–2021

Quarter	19-1	19-2	19-3	19-4	20-1	20-2	20-3	20-4	21-1	21-2	21-3	21-4	Average
Delivery rate	72%	70%	68%	69%	75%	71%	76%	66%	62%	71%	65%	74%	69.92%

From the data information shown in Table 8 and Fig. 9, it can be found that, on average, only 69.92% of the orders can be delivered on time. The highest on-time submission rate for orders is only 76%. This will reduce profits and significantly reduce customer satisfaction due to delayed orders. It can be revealed that the scheduling plan of company *A* is developed using manual preparation based

on the staff's experience, resulting in the long makespan, which ultimately led to low order delivery rates. Therefore, this paper uses the GNHHO algorithm to solve the FJSP of company *A*.

4.2.2 Static Scheduling for Company A

In one order, a total of 11 jobs need to be processed by decomposing the order, and these jobs have a capacity of 96 operations for company *A*. Company *A* uses the GNHHO algorithm to solve the FJSP. The solution result is shown in Fig. 10. The makespan is 125 hours.

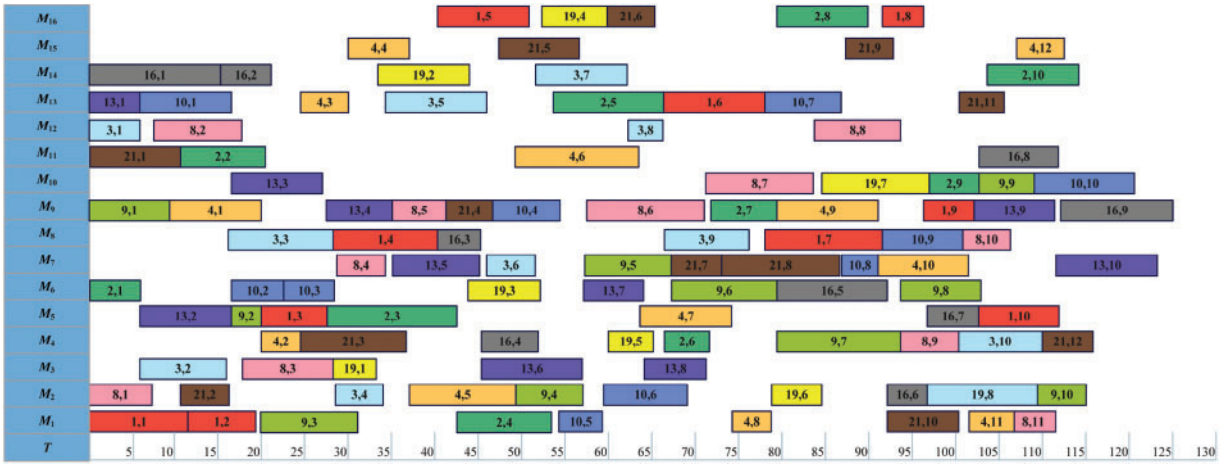


Figure 10: Gantt chart of static scheduling

Using the original manual scheduling scheme, the makespan is 174 hours. Therefore, the proposed GNHHO algorithm can shorten the makespan and help improve the order delivery rate.

4.2.3 Dynamic Scheduling for Company A

1. Dynamic adjustment strategy

Considering that company *A* will encounter various uncertain factors when carrying out production, particularly the uncertainty of the order. A dynamic event occurs at $t = 37$ hours. That is the demand for jobs increases. Under the influence of emergency order insertion, enterprises can adopt a rearranging strategy [37], which uses some intelligent algorithms to adjust the original scheduling when a new order is inserted. All the remaining orders and this inserted order are used as new production tasks to be re-formulated. This strategy allows most orders to be completed on time. When company *A* faces an urgent insertion order, the goal of dynamic adjustment is to minimize the makespan after adjustment.

2. Results of dynamic scheduling

When a dynamic event occurs at 37 h, we adopt a rearranging strategy to minimize the makespan after adjustment. Fig. 11 shows the new Gantt chart, and the makespan is 141 h. It is clear that the rescheduling schedule obtained by the GNHHO algorithm requires a much lower makespan than the company's original manual schedule, which required 203 h. The rearranging strategy is implemented to replace the initial strategy, which can respond to order changes more effectively.

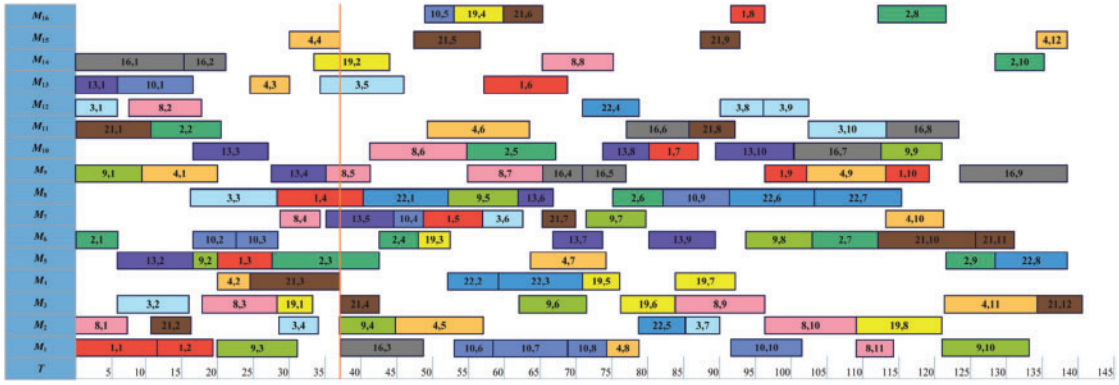


Figure 11: Gantt chart of the optimal scheduling after dynamic adjustment

4.2.4 Results Analysis

1. Comparison of order completion time

Table 9 and Fig. 12 show the specific data comparison of makespan. It shows that the optimized scheduling scheme can advance the makespan by 28.16% and 35.63% for static and dynamic scheduling, respectively. The reason is that the GNHHO algorithm introduces four strategies to jump out of the local optimum. This can demonstrate the effectiveness of the GNHHO algorithm in solving FJSP problems. The reduction of makespan is the basis for company *A* to achieve on-time delivery of orders.

Table 9: Comparison of company *A*'s static and dynamic scheduling optimization

Comparison	Makespan of static scheduling	Makespan of dynamic scheduling
Original scheme	174	203
Optimization scheme	125	141
Optimization rate	28.16%	35.63%

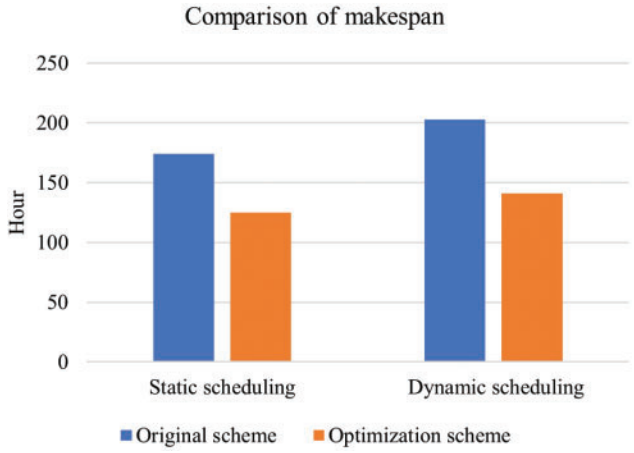


Figure 12: Comparison of makespan for static scheduling and dynamic scheduling

2. Comparison of order delivery rate

Based on the analysis that the optimized scheme can shorten the makespan of the order, this paper explores how the order delivery rate changes after optimization. We use the GNHHO algorithm to address the FJSP of company *A* for each quarter of 2021 and compare the new delivery time with the delivery time required by historical orders. The results are demonstrated in Table 10 and Fig. 13. Table 10 shows that the optimization rates for the four quarters are 24.50%, 20%, 23%, and 18.50%, respectively, because the GNHHO algorithm can obtain better solutions for FJSP. The data show an average 21.5% improvement in order delivery rates, which means that the optimized scheduling scheme can better solve the problem of the low delivery rate of orders of company *A*. Overall, the proposed GNHHO algorithm exhibits remarkable robustness, excellent optimization performance, and rapid convergence speed, which can effectively address the FJSP. Consequently, this can further promote the improvement of order delivery rates.

Table 10: Schedule delivery rate comparison of company *A* after scheduling optimization

Comparison	Quarter 1	Quarter 2	Quarter 3	Quarter 4
Original scheme	62%	71%	65%	74%
Optimization scheme	86.50%	91%	88%	92.50%
Optimization rate	24.50%	20%	23%	18.50%

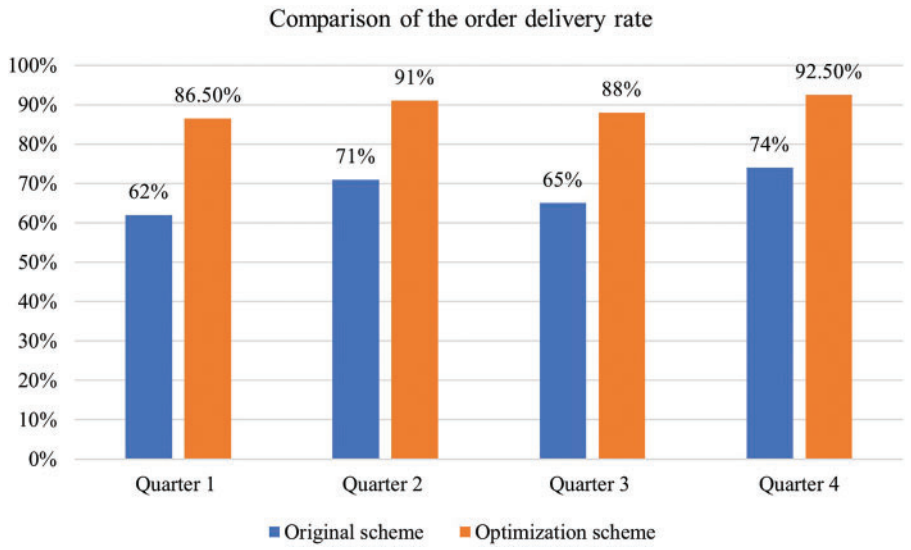


Figure 13: Comparison of the order delivery rate of the company *A*

5 Conclusion and Future Work

In this paper, we construct a flexible job shop scheduling model and propose a GNHHO algorithm to solve it. After verifying the effectiveness of the GNHHO algorithm by using benchmark test functions and standard JSP and FJSP examples, we use the algorithm to solve the FJSP of company *A*. According to the experimental results, the optimized scheduling scheme demonstrates significant

improvements in makespan, with an advancement of 28.16% for static scheduling and 35.63% for dynamic scheduling. Moreover, it achieves an average increase of 21.50% in the on-time order delivery rate. By comparing with previous studies, the GNHHO prevents the algorithm from premature convergence by replacing the optimal solution, adding a perturbation to the parameter r, proposing a nonlinear method to update the escaping energy E, and using a Gaussian random walk strategy to generate new individuals. The proposed GNHHO algorithm can solve FJSP efficiently.

In future research, we will focus on two aspects: Firstly, we aim to achieve a better balance between exploration and exploitation in the HHO algorithm by introducing four strategies, which increase the algorithm's complexity. Additionally, we plan to study other parameter tuning methods for further enhancement. Moreover, exploring alternative search strategies is expected to improve the overall effectiveness of the HHO algorithm. Secondly, our current paper on the FJSP problem has certain limitations as it does not consider uncertainties such as machine failures. For future work, we propose investigating and incorporating these uncertainties into the problem formulation to provide a more comprehensive and realistic analysis.

Acknowledgement: This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: conceptualization: Zhaolin Lv and Yuexia Zhao; methodology: Zhaolin Lv; software: Zhaolin Lv; resources: Yuexia Zhao; writing—original draft preparation: Zhaolin Lv and Yuhang Qin; writing—review and editing: Zhaolin Lv and Hongyue Kang; supervision: Zhenyu Gao; project administration: Yuexia Zhao. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The datasets used in the paper are benchmark and public datasets, which can be easily downloaded from the Internet. Other enterprise data was obtained from the company Bodeau Agricultural Company and are available from the authors with the permission of the company Bodeau Agricultural Company.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Zheng, L. Wang and J. J. Wang, "A cooperative coevolution algorithm for multi-objective fuzzy distributed hybrid flow shop," *Knowledge-Based Systems*, vol. 194, pp. 105536, 2020.

[2] Q. Pan, J. Tang, J. Zhan and H. Li, "Bacteria phototaxis optimizer," *Neural Computing & Applications*, vol. 35, no. 18, pp. 13433–13464, 2023.

[3] L. Abualigah, A. Diabat, P. Sumari and A. H. Gandomi, "Applications, deployments, and integration of Internet of Drones (IoD): A review," *IEEE Sensors Journal*, vol. 21, no. 15, pp. 25532–25546, 2021.

[4] B. Xu, Y. Tang, Y. Zhu, W. Yan, C. He *et al.,* "Bilateral collaborative optimization for cloud manufacturing service," *Computers, Materials & Continua*, vol. 64, no. 3, pp. 2031–2042, 2020.

[5] X. Zhang, Y. Han, G. Krolczyk, M. Rydel, R. Stanislawski *et al.,* "Rescheduling of distributed manufacturing system with machine breakdowns," *Electronics*, vol. 11, no. 249, pp. 249, 2022.

[6] L. Liu and L. Shi, "Automatic design of efficient heuristics for two-stage hybrid flow shop scheduling," *Symmetry*, vol. 14, no. 4, pp. 632, 2022.

[7] R. Chen, B. Yang, S. Li and S. Wang, "A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem," *Computers & Industrial Engineering*, vol. 149, pp. 106778, 2020.
[8] Y. Lu, C. Liu, K. I. K. Wang, H. Huang and X. Xu, "Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues," *Robotics and Computer-Integrated Manufacturing*, vol. 61, pp. 101837, 2020.
[9] B. Zhang, L. L. Meng, C. Lu, Y. Y. Han and H. Y. Sang, "Automatic design of constructive heuristics for a reconfigurable distributed flowshop group scheduling problem," *Computers Operations Research*, vol. 161, pp. 106432, 2024. https://doi.org/10.1016/j.cor.2023.106432
[10] C. Lu, L. Gao, J. Yi and X. Li, "Energy-efficient scheduling of distributed flow shop with heterogeneous factories: A real-world case from automobile industry in China," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 10, pp. 6687–6696, 2021.
[11] Q. Fan, Z. Chen and Z. Xia, "A novel quasi-reflected Harris hawks optimization algorithm for global optimization problems," *Soft Computing*, vol. 24, no. 19, pp. 14825–14843, 2020.
[12] M. A. Hamza, A. Abdelmaboud, S. Larabi-Marie-Sainte, H. M. Alshahrani, M. Al Duhayyim *et al.,* "Modified harris hawks optimization based test case prioritization for software testing," *Computers, Materials & Continua*, vol. 72, no. 1, pp. 1951–1965, 2022.
[13] C. Liu, "An improved harris hawks optimizer for job-shop scheduling problem," *Journal of Supercomputing*, vol. 77, no. 12, pp. 14090–14129, 2021.
[14] Y. H. Choo, Z. Cai, V. Le, M. Johnstone, W. K. Chan *et al.,* ""Multi-objective flexible job-shop scheduling with an ensemble optimization model,"," in *17th IEEE Industrial Electronics and Applications Conf. (IEACon)*, Chengdu, China, pp. 229–234, 2022.
[15] H. Jouhari, D. Lei, M. A. A. Al-qaness, M. A. Elaziz, R. Damaševičius *et al.,* "Modified Harris hawks optimizer for solving machine scheduling problems," *Symmetry*, vol. 12, no. 9, pp. 1460, 2020.
[16] D. A. Amer, G. Attiya, I. Zeidan and A. A. Nasr, "Elite learning Harris hawks optimizer for multi-objective task scheduling in cloud computing," *Journal of Supercomputing*, vol. 78, no. 2, pp. 2793–2818, 2022.
[17] I. Attiya, M. Abd Elaziz and S. Xiong, "Job scheduling in cloud computing using a modified Harris hawks optimization and simulated annealing algorithm," *Computational Intelligence and Neuroscience*, vol. 2020, no. 1, pp. 1–17, 2020.
[18] M. Shehab, I. Mashal, Z. Momani, M. K. Y. Shambour, A. AL-Badareen *et al.,* "Harris hawks optimization algorithm: Variants and applications," *Archives of Computational Methods in Engineering*, vol. 29, no. 7, pp. 5579–5603, 2022.
[19] E. H. Houssein, M. E. Hosney, D. Oliva, W. M. Mohamed and M. Hassaballah, "A novel hybrid Harris hawks optimization and support vector machines for drug design and discovery," *Computers & and Chemical Engineering*, vol. 133, pp. 106656, 2020. https://doi.org/10.1016/j.compchemeng.2019.106656
[20] A. A. Ewees and M. Abd Elaziz, "Performance analysis of chaotic multi-verse Harris hawks optimization: A case study on solving engineering problems," *Engineering Applications of Artificial Intelligence*, vol. 88, pp. 103370, 2020.
[21] Y. Wang, W. Peng, C. Lu and H. Xia, "A multi-objective cellular memetic optimization algorithm for green scheduling in flexible job shops," *Symmetry*, vol. 14, no. 4, pp. 832, 2022.
[22] S. Wang, H. Jia, L. Abualigah, Q. Liu and R. Zheng, "An improved hybrid Aquila optimizer and Harris Hawks algorithm for solving industrial engineering optimization problems," *Processes*, vol. 9, no. 9, pp. 1551, 2021.
[23] X. Zhong and P. Cheng, "An elite-guided hierarchical differential evolution algorithm," *Applied Intelligence*, vol. 51, no. 7, pp. 4962–4983, 2021.
[24] S. Y. D. Nezhad, N. Safdarian and S. A. H. Zadeh, "New method for fingerprint images encryption using DNA sequence and chaotic tent map," *Optik*, vol. 224, pp. 165661, 2020.
[25] F. Kutlu Onay, "A novel improved chef-based optimization algorithm with Gaussian random walk-based diffusion process for global optimization and engineering problems," *Mathematics and Computers in Simulation*, vol. 212, pp. 195–223, 2023. https://doi.org/10.1016/j.matcom.2023.04.027

[26] Y. Yuan, H. Xu and J. Yang, "A hybrid harmony search algorithm for the flexible job shop scheduling problem," *Applied Soft Computing*, vol. 13, no. 7, pp. 3259–3272, 2013.
[27] F. Luan, R. Li, S. Q. Liu, B. Tang, S. Li *et al.,* "An improved sparrow search algorithm for solving the energy-saving flexible job shop scheduling problem," *Machines*, vol. 10, no. 10, pp. 847, 2022.
[28] K. Gao, Z. Cao, L. Zhang, Z. Chen, Y. Han *et al.,* "A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 4, pp. 904–916, 2019.
[29] S. X. Lv, Y. R. Zeng and L. Wang, "An effective fruit fly optimization algorithm with hybrid information exchange and its applications," *International Journal of Machine Learning and Cybernetics*, vol. 9, no. 10, pp. 1623–1648, 2018.
[30] L. Abualigah, D. Yousri, M. Abd Elaziz, A. A. Ewees, M. A. A. Al-qaness *et al.,* "Aquila optimizer: A novel meta-heuristic optimization algorithm," *Computers and Industrial Engineering*, vol. 157, pp. 107250, 2021. https://doi.org/10.1016/j.cie.2021.107250
[31] B. Tutumlu and T. Sarac, "A MIP model and a hybrid genetic algorithm for flexible job-shop scheduling problem with job-splitting," *Computers & Operations Research*, vol. 155, pp. 106222, 2023.
[32] A. Amirteimoori, I. Mahdavi, M. Solimanpur, S. S. Ali and E. B. Tirkolaee, "A parallel hybrid PSO-GA algorithm for the flexible flow-shop scheduling with transportation," *Computers and Industrial Engineering*, vol. 173, pp. 108672, 2022. https://doi.org/10.1016/j.cie.2022.108672
[33] R. Chen, B. Yang, S. Li, S. Wang and Q. Cheng, "An effective multi-population grey wolf optimizer based on reinforcement learning for flow shop scheduling problem with multi-machine collaboration," *Computers and Industrial Engineering*, vol. 162, pp. 107738, 2021. https://doi.org/10.1016/j.cie.2021.107738
[34] M. A. A. Al-qaness, A. A. Ewees and M. Abd Elaziz, "Modified whale optimization algorithm for solving unrelated parallel machine scheduling problems," *Soft Computing*, vol. 25, no. 14, pp. 9545–9557, 2021.
[35] Y. Sun, J. S. Pan, P. Hu and S. C. Chu, "Enhanced equilibrium optimizer algorithm applied in job shop scheduling problem," *Journal of Intelligent Manufacturing*, vol. 34, no. 4, pp. 1639–1665, 2023.
[36] T. Ning and Y. Huang, "Low carbon emission management for flexible job shop scheduling: A study case in China," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, pp. 789–805, 2021. https://doi.org/10.1007/s12652-021-03330-6
[37] W. Jiang and L. Wu, "Flow shop optimization of hybrid make-to-order and make-to-stock in precast concrete component production," *Journal of Cleaner Production*, vol. 297, pp. 126708, 2021. https://doi.org/10.1016/j.jclepro.2021.126708