

Adaptive Mutation Driven Harris Hawk Optimization Algorithm in Hybrid Flow Workshop Scheduling with Robotic Arm

Bin Li

BinLi566@outlook.com

Xinxiang Vocational and Technical College, Xinxiang

Research Article

Keywords: mixed flow workshop scheduling, Harris Eagle Optimization Algorithm, Adaptive mutation, Levy flying, Smart Manufacturing

Posted Date: July 7th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-9911304/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Adaptive Mutation Driven Harris Hawk Optimization Algorithm in Hybrid Flow Workshop Scheduling with Robotic Arm

Bin Li*

*Xinxiang Vocational and Technical College, Xinxiang, Henan, 453006, China

E-mail: BinLi566@outlook.com

Author Contributions:

Bin Li conceived and designed the study, performed experiments, analyzed data, prepared figures, and wrote the manuscript. The author reviewed and approved the final manuscript.

Data Availability Statement:

The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Conflicts of Interest:

The authors declare no conflicts of interest.

Ethics, Consent to Participate, and Consent to Publish declarations

Not applicable.

Funding

This research received no external funding

Clinical trial number

Not applicable.

Abstract: In order to improve the efficiency and quality of solving mixed flow shop scheduling problems with robotic arms, the author proposes an adaptive mutation driven Harris Hawk optimization algorithm (AMDHHO). This algorithm integrates Levy flight disturbance and dynamic mutation probability strategy on the basis of standard HHO, achieving a balance between global exploration and local development. Through comparative experiments on five stage examples, the maximum completion time of AMDHHO reached 492.6 minutes, which is about 11.7% higher than that of genetic algorithm. In actual factory validation, for the scheduling problem of 20 workpieces and 3 robotic arms, AMDHHO reduced the maximum completion time to 683 minutes, which is 85 minutes less than the

original scheduling plan, and the average waiting time decreased by 39.8%. The experimental results show that the algorithm has significant advantages and good application prospects in complex scheduling environments.

Keywords: mixed flow workshop scheduling; Harris Eagle Optimization Algorithm; Adaptive mutation; Levy flying; Smart Manufacturing

1. Introduction

With the rapid development of intelligent manufacturing and flexible production modes, manufacturing enterprises have put forward higher requirements for production scheduling efficiency and quality. The Hybrid Flow Shop Scheduling Problem with Loading/Unloading Machines (HFSP-LUM), as a typical complex combinatorial optimization problem, involves not only the processing sequence and equipment allocation of workpieces on multi-stage parallel equipment, but also the overall consideration of the handling task arrangement of the robotic arm. Its optimization difficulty significantly increases with the expansion of the problem scale ^[1-2]. Traditional manual scheduling methods are difficult to cope with dynamic production demands of multiple varieties and small batches, while existing heuristic algorithms often fall into the dilemma of local optima or insufficient convergence speed when solving such high-dimensional and strongly constrained problems ^[3]. Therefore, developing intelligent optimization algorithms with stronger global search capabilities and dynamic adaptability has become a key research direction for improving the level of workshop scheduling, and has important theoretical value and practical significance.

In the research process of mixed flow shop scheduling problem, many scholars have conducted in-depth exploration from two aspects: model construction and algorithm design. Zhu et al. proposed a dual archive multi-objective algorithm for handling mixed flow shop scheduling problems with missing processes, effectively improving the adaptability of the algorithm in non complete process paths ^[4]. Zhang et al. developed a multi-objective evolutionary algorithm based on deep Q-network for distributed heterogeneous mixed flow workshops, combining reinforcement learning with swarm intelligence to optimize scheduling objectives while considering worker fatigue factors ^[5]. Zhao et al. further introduced

heterogeneous graph neural networks and deep reinforcement learning to construct a hybrid flowshop scheduling optimization framework under limited transportation resources, significantly improving the algorithm's modeling ability under complex resource constraints [6]. Zeng et al. proposed a meta heuristic algorithm framework for heterogeneous parallel machines, which enhances the robustness of the algorithm in heterogeneous device environments through multi strategy collaborative search [7]. These studies have made significant progress in model extension and algorithm fusion, providing important support for intelligent decision-making in complex scheduling scenarios.

Although many achievements have been made in the scheduling problem of mixed flow workshops, existing methods still have significant limitations in practical production scenarios with robotic arms [8]. On the one hand, most algorithms adopt fixed rules or simple priority strategies for robotic arm scheduling, which makes it difficult to dynamically adapt to changes in production pace and resource conflicts; On the other hand, traditional swarm intelligence algorithms such as genetic algorithms and particle swarm optimization are prone to premature convergence and loss of diversity when solving high-dimensional combinatorial problems, resulting in unstable scheduling scheme quality [9]. In addition, existing research mostly focuses on optimization in static environments, with weak real-time response capabilities for dynamic events such as equipment failures and order insertions, which limits its application effectiveness in practical industrial systems. Therefore, it is urgent to develop an optimization algorithm that can balance global exploration and local development, and has a dynamic adaptive mechanism, in order to improve the efficiency and robustness of solving mixed flow shop scheduling problems with robotic arms [10].

In response to the above issues, the author proposes an adaptive mutation driven Harris Hawk optimization algorithm (AMDHHO), which aims to enhance the search capability and convergence stability of the algorithm in complex scheduling environments by introducing Levy flight disturbance mechanism and dynamic mutation probability strategy. This algorithm is based on the standard Harris Hawk optimization framework, integrating multi method crossover operators and hybrid encoding mechanisms to achieve three-dimensional collaborative optimization of workpiece processing sequence, equipment allocation, and robotic arm tasks [11-12]. By constructing a mathematical model with the goal of minimizing

the maximum completion time and designing corresponding constraint handling and decoding strategies, the feasibility and effectiveness of the scheduling scheme are ensured. The author will validate the performance of AMDHHO algorithm in standard and actual factory cases, and compare it with various mainstream algorithms to systematically evaluate its comprehensive advantages in solving quality, convergence speed, and dynamic adaptability, providing an efficient and reliable solution for optimizing complex workshop scheduling problems in the context of intelligent manufacturing.

2. Modeling of scheduling problem for mixed flow workshop with robotic arm

In the hybrid flowshop scheduling problem with robotic arms (HFSP-LUM), it is necessary to comprehensively consider multidimensional decision variables such as workpiece processing sequence, equipment selection, and robotic arm task allocation. Taking a certain stage as an example, the scheduling diagram is shown in Figure 1. This section constructs a mathematical model with the goal of minimizing the maximum completion time, which fully reflects the resource constraints and temporal dependencies in production scheduling, providing a theoretical basis for the design of subsequent optimization algorithms ^[13].

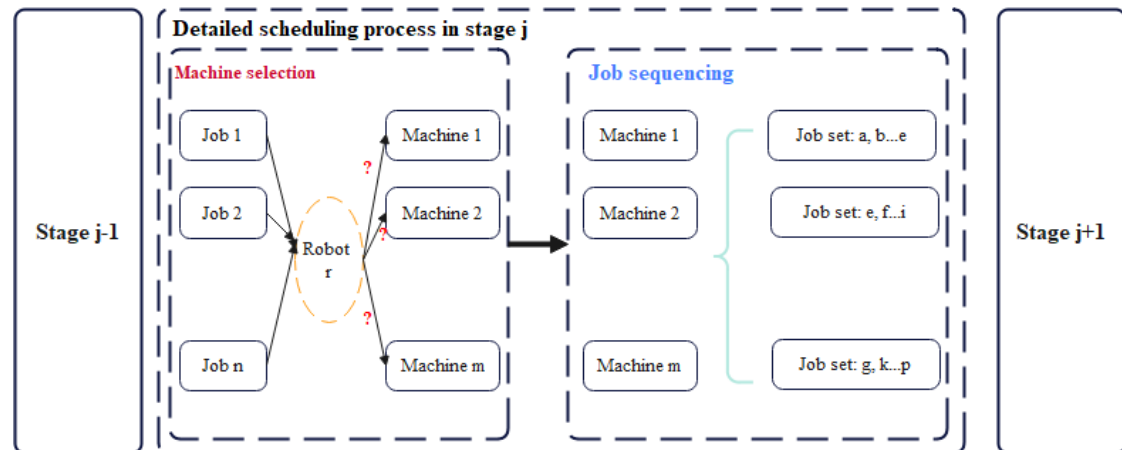


Figure 1 HFSP-LUM process

The scheduling problem of a mixed flow workshop with robotic arms can be described as: n workpieces are processed in m processes, each process contains a set of parallel identical equipment, and there are robotic arms between adjacent processes responsible for material

handling ^[14-15]. The model is based on the following assumptions:

- (1) Each workpiece must go through all processes in order according to the predetermined process sequence
- (2) Each device can only process one workpiece at a time
- (3) Once the machining of a workpiece in a process begins, it cannot be interrupted
- (4) The robotic arm can only transport one workpiece at a time and has a fixed handling time
- (5) All workpieces, equipment, and robotic arms are available at zero time
- (6) Ignore equipment failures and maintenance time

In order to clearly express the HFSP-LUM model, the following symbols are defined:

Table 1 Data Symbols and Definitions

Classification	Symbol	Definition
Indicator set	$j=\{1,2,..n\}$	Collection of workpieces
	$s=\{1,2,..m\}$	Process set
	$M_s=\{1,2,..m_s\}$	Parallel device set on process s
	$R=\{1,2,..r\}$	Mechanical arm assembly
Parameter	$p_{j,s,k}$	The processing time of workpiece j on equipment k in process s
	$t_{s,s+1}^r$	The transportation time of robotic arm r from process s to process s+1
	$e_{j,s,k}$	The mark indicating whether workpiece j can be processed on equipment k in process s (1 indicates yes, 0 indicates no)
	$x_{j,s,k}$	Is workpiece j being processed on equipment k in process s (1 indicates yes, 0 indicates no)
Decision variable	$y_{j,j',s}$	Is workpiece j processed before workpiece j' in process s (1 indicates yes, 0 indicates no)
	$z_{j,s}^r$	Is the workpiece j transported by the robotic arm r from process s to process s+1? (1 indicates yes, 0 indicates no)
	$ST_{j,s}$	The starting processing time of workpiece j in process s
	$CT_{j,s}$	The completion processing time of workpiece j in process s
	$RT_{j,s}$	The start time of moving workpiece j from process s to process s+1
	$AT_{j,s}$	The time for workpiece j to arrive at process s+1

Based on the above symbol definitions, the mathematical model of HFSP-LUM can be expressed as:

$$\min C_{\max}$$

The objective function aims to minimize the maximum completion time of all workpieces, that is the production cycle.

Constraints:

$$CT_{j,s} = ST_{j,s} + \sum_{k \in M_s} p_{j,s,k} \cdot x_{j,s,k}, \forall j \in J, s \in S$$

This constraint defines that the completion time of the workpiece in each process is equal to the start time plus the corresponding processing time.

$$C_{\max} \geq CT_{j,m}, \forall j \in J$$

The maximum completion time must be greater than or equal to the completion time of all workpieces in the final process.

$$\sum_{k \in M_s} x_{j,s,k} = 1, \forall j \in J, s \in S$$

Each workpiece must and can only be assigned to one equipment for processing in each process.

$$x_{j,s,k} \leq e_{j,s,k}, \forall j \in J, s \in S, k \in M_s$$

Workpieces can only be assigned to equipment with corresponding processing capabilities.

$$ST_{j',s} \geq CT_{j,s} - M \cdot (1 - y_{jj',s}), \forall j, j' \in J, j \neq j', s \in S$$

$$ST_{j,s} \geq CT_{j',s} - M \cdot y_{jj',s}, \forall j, j' \in J, j \neq j', s \in S$$

Among them, M is a sufficiently large positive number. These two constraints ensure that multiple workpieces will not be processed simultaneously on the same device.

Process sequence constraint: $ST_{j,s+1} \geq AT_{j,s+1}, \forall j \in J, s \in \{1, 2, \dots, m-1\}$

The start time of the workpiece in the subsequent process must be after the workpiece is transported to the process by the robotic arm.

$$\sum_{r \in R} z_{j,s}^r = 1, \forall j \in J, s \in \{1, 2, \dots, m-1\}$$

Each workpiece must and can only be transported by one robotic arm from one process to the next.

$$AT_{j,s+1}=RT_{j,s}+\sum_{r \in R} t_{s,s+1}^r \cdot z_{j,s}^r, \forall j \in J, s \in \{1, 2, \dots, m-1\}$$

The time when the workpiece reaches the next process is equal to the start of handling plus the corresponding handling time.

$$RT_{j,s} \geq CT_{j,s}, \forall j \in J, s \in \{1, 2, \dots, m-1\}$$

The robotic arm can only start handling after the workpiece has completed the current processing step.

Mutual exclusion constraint of robotic arm:

For any two workpieces j and j' , if they use the same robotic arm r and j is moved before j' , then:

$$RT_{j',s} \geq RT_{j,s} + \sum_{r \in R} t_{s,s+1}^r \cdot z_{j,s}^r - M \cdot (2 - z_{j,s}^r - z_{j',s}^r - y_{jj',s}), \forall j, j' \in J, j \neq j', s \in S, r \in R$$

This constraint ensures that the same robotic arm can only transport one workpiece at a time.

$$x_{j,s,k}, y_{jj',s}, z_{j,s}^r \in \{0, 1\}$$

$$ST_{j,s}, CT_{j,s}, RT_{j,s}, AT_{j,s} \geq 0$$

The above model constructs a complete hybrid flow shop scheduling optimization framework, which improves production efficiency by minimizing the maximum completion time, while satisfying multiple constraints such as equipment allocation, process sequence, and robot scheduling [16-17]. The complexity of the model is mainly reflected in the collaborative optimization of multidimensional decision variables, as well as the temporal dependencies between various resources (workpieces, equipment, robotic arms).

3. Principle and improvement of adaptive mutation driven Harris optimization algorithm

3.1 Basic Algorithm Ideas

Harris eagles exhibit a unique "ambush encirclement" tactic during hunting, with multiple Harris eagles working together and adopting different attack strategies from different angles to capture prey. The HHO algorithm is based on this behavior pattern and has constructed a two-stage optimization framework that includes an exploration phase and a development phase [18]. In the exploration phase, the algorithm simulates the behavior of Harris hawks randomly patrolling and waiting for prey; During the development phase, the algorithm

simulates the tracking, encirclement, and attack behavior of Harris hawks after discovering prey. The flowchart is shown in Figure 2.

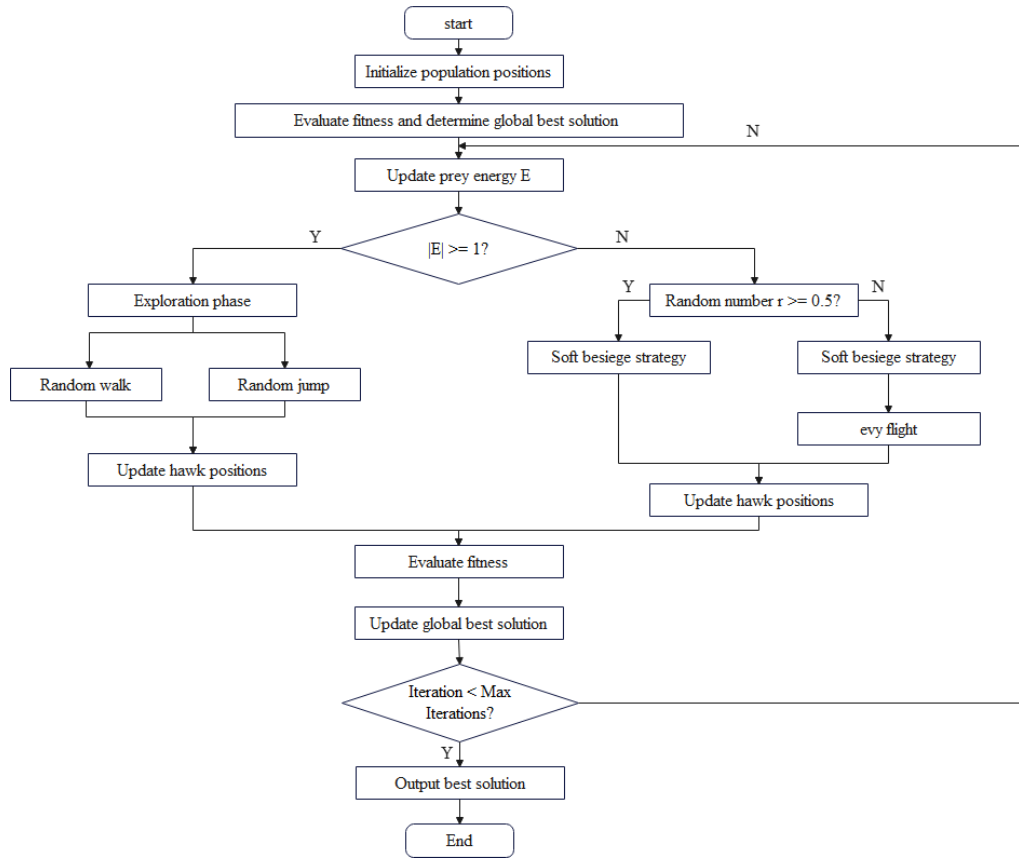


Figure 2 Harris Eagle Algorithm Flowchart

The core mechanism of HHO algorithm mainly includes the following parts:

Random exploration and search: In the initial stage of the algorithm, Harris Hawk agents are randomly distributed in the search space and searched through two different strategies: one is based on random walks of the Harris Hawk population's internal position; The second is random jumping based on search space boundaries ^[19]. This dual exploration strategy can be expressed as:

$$X(t+1) = \begin{cases} X_{rand}(t) - r_1 |X_{rand}(t) - 2r_2 X(t)|, & q \geq 0.5 \\ (X_{rabbit}(t) - X_m(t)) - r_3 (LB + r_4 (UB - LB)), & q < 0.5 \end{cases}$$

Among them, $X(t)$ represents the position vector of the current iteration, $X_{rand}(t)$ is a randomly selected Harris hawk, $X_{rabbit}(t)$ is the currently discovered optimal solution (prey position), $X_m(t)$ is the average position of the Harris hawk population, r_1 , r_2 , r_3 , r_4 , and q are random numbers within the range of $[0,1]$, LB and UB represent the upper and lower

boundaries of the search space, respectively.

Encirclement and Energy Conversion: When the algorithm enters the development stage, Harris Hawk selects different attack strategies based on the energy state of the prey (i.e. the current global optimal solution). The energy E required for prey escape decreases with the increase of iteration times, and the calculation formula is:

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

Among them, E_0 is the initial energy, t is the current iteration count, and T is the maximum iteration count. The change in energy E enables the algorithm to adaptively transition from global exploration to local development.

$$X(t+1) = X_{\text{rabbit}}(t) - E|JX_{\text{rabbit}}(t) - X(t)|$$

Among them, $J = 2(1-r_5)$ represents the random jumping intensity of prey during escape, and r_5 is a random number within $[0,1]$.

When $r \geq 0.5$, adopt a progressive approach strategy:

$$X(t+1) = (X_{\text{rabbit}}(t) - X(t)) - E|JX_{\text{rabbit}}(t) - X(t)|$$

When $r < 0.5$, use one of four different attack strategies, including:

$$X(t+1) = \begin{cases} Y, & |E| \geq 0.5 \\ Z, & |E| < 0.5 \text{ and } r_6 \geq 0.5 \\ Y+Z, & |E| < 0.5 \text{ and } r_6 < 0.5 \end{cases}$$

Among them, Y and Z are candidate positions calculated based on Levy Flight and prey position.

Group collaboration and location update: The HHO algorithm achieves collaborative search through information exchange among group members. After each iteration, the global optimal solution (prey location) is updated based on fitness evaluation. The entire population gradually converges towards the optimal solution while maintaining appropriate diversity to avoid falling into local optima ^[20].

3.2 Mutation Strategy and Search Mechanism

The Adaptive Mutation Driven Harris Hawk Optimization Algorithm (AMHHO), while retaining the core search mechanism of the original HHO algorithm, combines the Levy flight mutation strategy and dynamic mutation probability mechanism to form a comprehensive search process and position update mechanism ^[21]. This section elaborates on the search

process and key update formulas of the improved algorithm in detail.

The overall search process of AMHHO algorithm includes three main stages: initialization, surrounding, energy attenuation conversion, and collaborative predation. At each stage, corresponding mutation strategies are integrated to enhance algorithm performance.

Initialization phase

Randomly generate the initial positions of N Harris Eagle individuals in the D-dimensional search space:

$$X_i = X_{\min} + \text{rand}(0, 1) \times (X_{\max} - X_{\min}), i = 1, 2, \dots, N$$

Among them, X_i represents the position vector of the i -th Harris hawk, $X_{\min}$ and $X_{\max}$ are the lower and upper bounds of the search space, respectively, and $\text{rand}(0, 1)$ is a uniform random number in the $[0, 1]$ interval.

Surrounding stage

In each iteration, Harris hawks adopt different hunting strategies based on the escape energy E of the current optimal solution. The calculation formula for E is:

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

Among them, E_0 is the initial energy, usually taken as 1; T is the current iteration count; T is the maximum number of iterations.

$$X(t+1) = X_{\text{rabbit}}(t) - E |J X_{\text{rabbit}}(t) - X(t)|$$

Among them, $X_{\text{rabbit}}(t)$ is the currently discovered optimal position, and J is the random jump strength used to simulate random movement.

At this stage, Levy flight variation is introduced to mutate individuals with probability P_m (t):

$$X_{\text{new}} = X(t+1) + \alpha \oplus \text{Levy}(\lambda) \otimes (X_{\text{best}} - X(t+1))$$

Among them, α is the dynamic step size scaling factor calculated according to the aforementioned formula.

Energy attenuation conversion

$$r = 0.5 + \frac{\arctan\left(\frac{E_d - E_{d,\text{avg}}}{E_{d,\text{std}}}\right)}{2\pi}$$

Among them, E_d is the energy consumption of population escape, $E_{d,\text{avg}}$ and $E_{d,\text{std}}$ are the average and standard deviation of population energy consumption, respectively.

$$X(t+1)=\Delta X(t)-E|\Delta X(t)|$$

Among them, $\Delta X(t)=X_{\text{rabbit}}(t)-X(t)$ is the position difference between Harris Eagle and Rabbit.

$$X(t+1)=X_{\text{rabbit}}(t)-E|\Delta X(t)|$$

Collaborative predation stage

When $r \geq 0.5$, the Harris eagle enters the cooperative predation stage, and the algorithm simulates the cooperative behavior of the Harris eagle population. The location update formula is:

$$X(t+1)=X_{\text{rabbit}}(t)-E|JX_{\text{rabbit}}(t)-X_{\text{avg}}(t)|$$

Among them, $X_{\text{avg}}(t)$ is the average position vector of the current population.

At this stage, an adaptive differential update strategy was designed to enhance population diversity:

$$Y=X_{\text{rabbit}}(t)-E|JX_{\text{rabbit}}(t)-X(t)|Z=Y+S \times \text{Levy}(D) \times \left(1-\frac{t}{T}\right)$$

Among them, S is a random search vector, $S=\text{rand}(-1, 1)$. The final position update is determined by fitness:

$$X(t+1)=\begin{cases} Y, & \text{if } F(Y)<F(X(t)) \\ Z, & \text{if } F(Z)<F(X(t)) \\ X(t), & \text{otherwise} \end{cases}$$

In the collaborative predation stage, the dynamic mutation probability mechanism is also applied to perform Levy flight mutation on some individuals to avoid premature convergence of the population.

Boundary handling and elite retention

After each location update, it is necessary to check whether the individual has exceeded the search space boundary:

$$X_i^j=\begin{cases} X_{\min}^j, & \text{if } X_i^j<X_{\min}^j \\ X_{\max}^j, & \text{if } X_i^j>X_{\max}^j \\ X_i^j, & \text{otherwise} \end{cases}$$

Among them, X_i^j represents the position value of the i -th individual in the j -th dimension.

Meanwhile, in order to ensure the convergence of the algorithm, an elite retention strategy is adopted to ensure that the current optimal solution is not lost during the iteration process.

Through the above search process and update formula, the adaptive mutation driven Harris

Hawk optimization algorithm has achieved comprehensive enhancement of the original HHO algorithm. The pseudocode is shown in Figure 3. In the scheduling problem of mixed flow workshops with robotic arms, this algorithm can effectively balance global exploration and local development, overcome the premature convergence problem of traditional swarm intelligence algorithms, and improve the quality and stability of solutions [22-23].

Algorithm 1 AMHHO Algorithm

```

1: Initialize parameters: D-dimensional search space, N hawk individuals,
   lower bound  $\mathbf{l}$ , upper bound  $\mathbf{u}$ , maximum iterations T
2: Randomly generate initial positions  $\mathbf{X}_i$  for  $i = 1$  to N:
3:  $\mathbf{X}_i = \mathbf{l} + \text{rand}([0, 1]) \cdot (\mathbf{u} - \mathbf{l})$ 
4: Compute initial best position  $\mathbf{X}_{\text{best}}$  and energy  $E_0 = 2 \cdot \text{rand}() - 1$ 
5: for  $t = 1$  to T do
6:   Calculate escape energy  $E = E_0 \cdot (1 - \frac{t}{T})$ 
7:   Encircling Phase:
8:   if  $|E| \geq 0.5$  then
9:     Update position:  $\mathbf{X}_i(t+1) = \mathbf{X}_{\text{best}} - E \cdot J \cdot |\mathbf{X}_{\text{best}} - \mathbf{X}_i(t)|$ 
10:    Apply Levy flight mutation with probability  $P_m(t)$ :  $\mathbf{X}_i(t+1) =$ 
        $\mathbf{X}_i(t) + \alpha \cdot \text{Levy}()$ 
11:   else
12:     Update position:  $\mathbf{X}_i(t+1) = \mathbf{X}_{\text{best}} - E \cdot |\mathbf{X}_{\text{best}} - \mathbf{X}_i(t)|$ 
13:   end if
14:   Energy Decay Transition:
15:   Compute energy consumption:  $E_{\text{prey}} = E_{\text{avg}} - \sigma \cdot \text{rand}()$ 
16:   Position update:  $\Delta \mathbf{X} = \mathbf{X}_{\text{best}} - \mathbf{X}_i(t)$ 
17:   Cooperative Hunting Phase:
18:   if  $\text{rand}() \geq 0.5$  then
19:     Compute average position  $\mathbf{X}_{\text{avg}}$ 
20:     Update position:  $\mathbf{X}_i(t+1) = \mathbf{X}_{\text{avg}} - E_{\text{prey}} \cdot |\mathbf{X}_{\text{avg}} - \mathbf{X}_i(t)|$ 
21:     Apply adaptive differential update:  $\mathbf{X}_i(t+1) = \mathbf{X}_i(t) + S$ , where
        $S = \text{rand}(-1, 1)$ 
22:   end if
23:   Select final position based on fitness
24:   Apply Levy flight mutation with dynamic probability
25:   Boundary Handling and Elite Preservation:
26:   for  $j = 1$  to D do
27:     if  $\mathbf{X}_{i,j} < \mathbf{l}_j$  or  $\mathbf{X}_{i,j} > \mathbf{u}_j$  then
28:        $\mathbf{X}_{i,j} = \text{clamp}(\mathbf{X}_{i,j}, \mathbf{l}_j, \mathbf{u}_j)$ 
29:     end if
30:   end for
31:   Preserve elite solution: Update  $\mathbf{X}_{\text{best}}$  if a better solution is found
32: end for
33: return  $\mathbf{X}_{\text{best}}$ 

```

Figure 3 Pseudo code of AMHHO algorithm

4. Encoding, Decoding, and Operator Design

4.1 Encoding and Decoding Strategy

In the scheduling problem of mixed flow workshops with robotic arms, encoding and decoding strategies are the basis for effective optimization algorithms. A reasonable encoding method can ensure that the search space matches the actual problem, while an efficient decoding strategy can correctly map the abstract solutions generated by the algorithm to scheduling schemes with practical significance [24-25].

In response to the characteristics of scheduling problems in mixed flow workshops with robotic arms, coding design needs to simultaneously consider three key dimensions:

workpiece processing sequence, parallel machine allocation, and robotic arm task arrangement. The author adopts a hybrid encoding structure and represents a complete scheduling solution through multi-dimensional vector combinations. The coding design follows the following principles:

The author adopts a three-dimensional hybrid encoding structure, including Processing Sequence Vector (PSV), Machine Assignment Vector (MAV), and Robot Assignment Vector (RAV) [26-27]. The three together form a complete chromosome representation:

$$C=[PSV, MAV, RAV]$$

The specific definitions of each part are as follows:

Process Order Vector (PSV): Using a workpiece based arrangement representation, the length is the total number of workpieces n , where elements represent the priority order of workpiece processing.

$$PSV=[p_1, p_2, \dots, p_n]$$

among which, $p_i \in \{1, 2, \dots, n\}$, And $p_i \neq p_j$ ($i \neq j$) represents the workpiece number at the i -th position in the sorting.

Machine Allocation Vector (MAV): represents the specific machine selected for each workpiece in each process, with a length of $n \times m$ (n is the number of workpieces, m is the number of processes).

$$MAV=[m_{11}, m_{12}, \dots, m_{1m}, m_{21}, \dots, m_{nm}]$$

Among them, $m_{ij} \in \{1, 2, \dots, M_j\}$ represents the machine number assigned to workpiece i on process j , and M_j is the number of parallel machines on process j .

Robot arm allocation vector (RAV): specifies the robot arm used for transferring each workpiece between processes, with a length of $n \times (m-1)$.

$$RAV=[r_{11}, r_{12}, \dots, r_{1,m-1}, r_{21}, \dots, r_{n,m-1}]$$

Among them, $r_{ij} \in \{1, 2, \dots, R\}$ represents the number of robotic arms used when workpiece i is transferred from process j to process $j+1$, and R is the total number of available robotic arms.

The decoding process converts 3D encoding into an actual executable scheduling scheme, including determining the order of workpiece processing, allocating machine resources, and scheduling the operation time of the robotic arm. Decoding follows the following steps:

Step 1: Determine the processing sequence of the workpieces in the first process based on the process sorting vector PSV, and initialize the start processing time of each workpiece [28].

Step 2: decode each workpiece in sequence according to the PSV, and for the current workpiece p_i : obtain the machine used in each process for that workpiece from the machine allocation vector MAV; Obtain the robotic arm used for transferring the workpiece between different processes from the robot arm allocation vector RAV.

Step 3: For the j th process of workpiece p_i , calculate its start processing time $S_{p_i,j}$ and completion time $C_{p_i,j}$:

$$S_{p_i,j} = \max \{ C_{p_i,j-1} + T_{p_i,j-1,j}^R, A_{m_{p_i,j}} \}$$

$$C_{p_i,j} = S_{p_i,j} + P_{p_i,j,m_{p_i,j}}$$

Among them, $T_{p_i,j-1,j}^R$ represent the time required for the robotic arm $r_{p_i,j-1}$ to transfer the workpiece p_i from process $j-1$ to process j ; $A_{m_{p_i,j}}$ represents the available time of machine $m_{p_i,j}$; $P_{p_i,j,m_{p_i,j}}$ represent the machining time of workpiece p_i on machine $m_{p_i,j}$ in process j .

Step 4: Update the available time of the machine and robotic arm:

$$A_{m_{p_i,j}} = C_{p_i,j}$$

$$AR_{r_{p_i,j}} = C_{p_i,j} + T_{p_i,j}^L$$

Among them, $A_{m_{p_i,j}}$ represent the available time of the robotic arm $r_{p_i,j}$, and $T_{p_i,j}^L$ represent the time required for the robotic arm to unload the workpiece p_i .

Step 5: Repeat steps 3-4 until all processes of all workpieces have been processed.

Step 6: Calculate the maximum completion time (makespan):

$$C_{\max} = \max_{i=1,2,\dots,n} \{ C_{i,m} \}$$

The core of the decoding mechanism lies in accurately tracking and updating the machining status of the workpiece, machine occupancy, and availability of the robotic arm, ensuring that resource constraints and process constraints are met. Especially when dealing with parallel operations of multiple robotic arms, it is necessary to avoid resource conflicts and ensure the feasibility of solutions [29].

4.2 Multi method cross strategy

PMX (Partially Mapped Crossover) is a classic crossover operator for handling permutation

problems, particularly suitable for optimizing the machining sequence of workpieces in mixed flow shop scheduling problems with robotic arms. In the intelligent manufacturing environment, PMX can effectively maintain the relative positional relationship between workpieces, ensuring that the new solutions generated by cross operations meet the constraints of the scheduling problem [30].

The basic idea of PMX crossover operator is to establish a mapping relationship between two parent solutions, generating offspring solutions that inherit the excellent genes of the parent and satisfy the constraint conditions. In the scheduling problem of a mixed flow workshop with robotic arms, the workpiece sequence must follow a one-to-one mapping relationship, and each workpiece can only appear once in the machining sequence. PMX is an efficient crossover method designed for this characteristic [31].

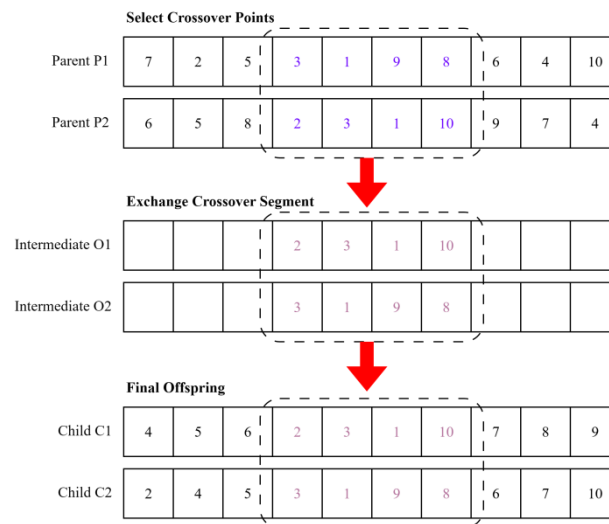


Figure 4 PMX process

The specific implementation of PMX crossover process is as follows:

Step 1: Randomly select two intersection points from the two parent solutions P₁ and P₂, and determine the intersection interval [a, b].

Step 2: Directly exchange the genes located in the crossover interval [a, b] between P₁ and P₂ to obtain two intermediate solutions O1 and O2.

Step 3: Establish a mapping relationship table M from P1 to P2 based on the corresponding positions of genes within the crossover interval.

Step 4: Perform conflict detection on genes outside the crossover interval in the intermediate solution. If duplicate genes are found, replace them according to the mapping relationship table until there are no conflicts ^[32-33].

Step 5: Generate two offspring solutions C1 and C2 that satisfy the constraint conditions.

Uniform Crossover is an important crossover strategy primarily used for optimizing equipment selection and task allocation in mixed flow shop scheduling problems with robotic arms. The core idea is to generate a binary crossover mask of the same length for each decision variable, and decide whether to swap parent genes bit by bit: When the mask position is 1, the genes at the corresponding positions of the two parents are swapped, otherwise the original value is retained, thus flexibly exploring the solution space bit by bit. After crossing, fix infeasible solutions based on scheduling constraints to ensure the effectiveness of offspring ^[34]. In addition, the author introduces an adaptive crossover probability control mechanism, where the crossover probability is dynamically adjusted based on individual fitness, using a linear strategy with an upper limit of 0.9 and a lower limit of 0.6; And a two-level uniform cross process was designed, focusing on workpiece machine allocation and robotic arm task arrangement, as shown in Figure 5.

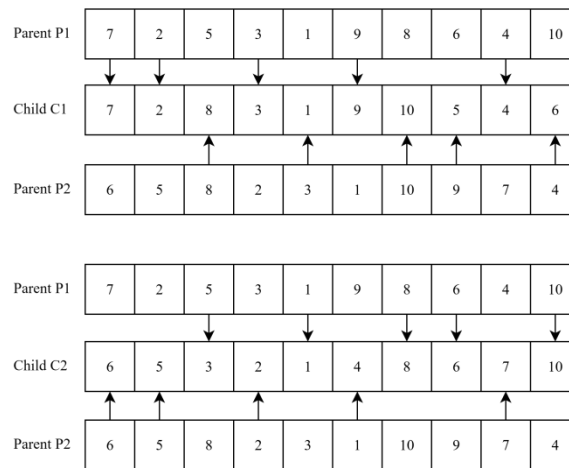


Figure 5 Uniform Cross Process

5. Example design and experimental results

5.1 Examples and Experimental Environment

In order to comprehensively evaluate the performance of the proposed Adaptive Mutation Driven Harris Hawk Optimization Algorithm (AMHHO) in mixed flow shop scheduling

problems with robotic arms, the authors designed two types of datasets, including standard benchmark cases and extended cases, to verify the adaptability of the algorithm in different scale and complexity scenarios (see Table 1). The standard benchmark case is modified based on the Taillard assembly line workshop problem set, covering three scales: small, medium, and large. The extended case combines actual data from a smart manufacturing demonstration factory, focusing on complex characteristics such as process coupling and robot task conflicts. The processing and operation time are generated uniformly according to different scales [35]. The experiment was conducted on a high-performance hardware platform (Intel i9-12900K, 64GB memory) and MATLAB R2023a environment. Each example was independently run 30 times to compare the performance of AMHHO with existing mainstream algorithms in terms of optimal solution quality, convergence speed, and robustness, ensuring fairness and reliability.

Table 1 Extended Example Parameter Configuration

Example number	Quantity of workpieces	Number of processes	Number of stages	Number of parallel machines in each stage	Number of robotic arms
P1	12	7	4	[2, 3, 2, 2]	2
P2	18	9	5	[2, 3, 3, 2, 2]	3
P3	25	12	6	[3, 2, 3, 3, 2, 2]	4
P4	35	15	7	[3, 3, 2, 3, 3, 2, 3]	5
P5	45	20	10	[3, 2, 3, 3, 2, 3, 2, 3, 2, 2]	8

5.2 Algorithm parameter settings

The setting of algorithm parameters has a decisive impact on optimization performance. The author designed the key parameters of the Adaptive Mutation Driven Harris Hawk Optimization Algorithm (AMHHO) in detail (see Table 2) and determined the optimal combination through sensitivity analysis. Parameter setting mainly considers algorithm convergence, problem size, computational resources, and parameter interaction effects. Among them, the population size is set to 50, taking into account both convergence quality and efficiency; The maximum number of iterations is 200, which ensures sufficient search; The adaptive mutation probability dynamically adjusts within the range of [0.1, 0.5],

strengthening global search in the early stage and focusing on local development in the later stage; Levy flight factor β was set to 1.5, and experimental verification showed that it performed the best in search balance (see sensitivity analysis). To ensure fairness in comparison, key parameters of comparison methods such as genetic algorithm, mayfly algorithm, and beetle optimization algorithm are also configured according to best practices (see Table 3) and run in the same hardware and software environment to ensure comparability and reliability of results.

Table 2 AMHHO Algorithm Parameter Setting Table

Parameter name	Parameter symbol	Parameter value
Population size	pop_size	50
Maximum number of iterations	max_iter	200
Adaptive mutation probability upper limit	Pm_max	0.5
Adaptive mutation probability lower limit	Pm_min	0.1
Levy Flight Factor	β	1.5
Energy attenuation factor	E ₀	0.01
Jump intensity factor	J	2.0
Convergence factor	α	0.1
Adaptive weight upper limit	w_max	0.9
Adaptive weight lower limit	w_min	0.4

Table 3 Comparison of Algorithm Parameter Settings

Algorithm Name	Key parameters	Parameter value
Genetic Algorithm (GA)	Cross probability	0.85
	mutation probability	0.15
	Selection method	Tournament selection
Mayfly Algorithm (MA)	Separation factor	0.1
	alignment factor	0.2
	Cohesive factor	0.7
Dungeon	Rolling ball	0.9

Optimization	probability	
Algorithm (DO)	Mining probability	0.1
	Flight factor	0.3

5.3 Experimental Results

1) 5 Stage Case Study Results

Table 4 Performance Comparison of Various Algorithms on the 5-stage HFSP-LUM

Problem													
Scale Examp le numbe r	(workpie ce x stage x machine)	AVDHHO			HHO			MA			GA		
		optim al value	avera ge	standar d deviati on	optim al value	avera ge	standar d deviati on	optim al value	avera ge	standar d deviati on	optim al value	avera ge	standar d deviati on
P1	10×5×2	198.5	200.1	1.03	205.2	209.3	2.11	213.6	218.2	2.87	221.3	228.4	4.25
P2	15×5×2	231.4	233.8	1.36	242.1	245.7	2.23	252.5	257.3	3.12	260.2	266.4	4.83
P3	20×5×3	283.7	286.2	1.58	296.4	301.2	2.65	308.7	314.9	3.78	315.6	324.5	5.41
P4	30×5×3	367.2	371.4	2.14	385.9	392.7	3.42	402.3	409.8	4.35	412.5	421.6	6.28
P5	50×5×5	492.6	498.3	3.27	518.7	527.5	4.81	541.2	551.8	6.24	557.8	568.3	7.92

A systematic analysis was conducted on the performance of the proposed adaptive mutation driven Harris Hawk optimization algorithm (AVDHHO) on a 5-stage mixed flow shop scheduling problem. Twenty different scale examples (with 10-50 workpieces and 2-5 machines) were selected for the experiment, and compared with the original HHO, mayfly algorithm (MA), genetic algorithm (GA), and beetle optimization algorithm (DO) under the same hardware and parameter configuration (see Table 4). The results showed that AVDHHO achieved the optimal maximum completion time (Cmax) in all cases, such as in the maximum scale problem P5, where it improved by 5.0% and 11.7% compared to the original HHO and GA, respectively. At the same time, AVDHHO has significant stability advantages, with the standard deviation being the smallest in all cases, especially in large-scale problems, which is

only 41.3% of GA, demonstrating good robustness and consistency. As the scale of the problem expands, the difference in algorithm performance further widens, verifying the superiority of adaptive mutation strategy and Levy flight mechanism in complex scenarios.

2) Comparison of Algorithm Convergence Curves

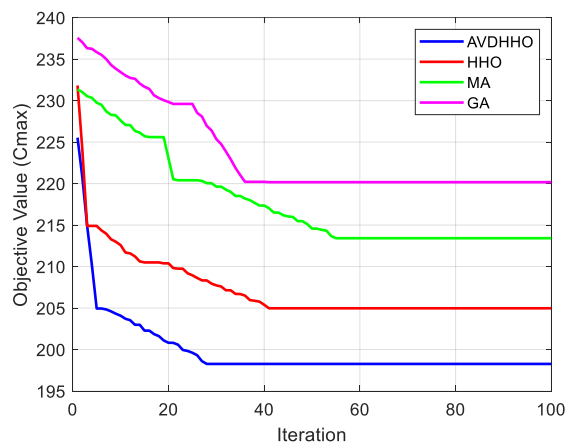


Figure 6 Convergence curves of different algorithms in P1 environment

In the P1 ($10 \times 5 \times 2$) example, the convergence curves and performance differences of each algorithm are significant, as shown in Figure 6. AVDHHO rapidly decreased to 205.3 in the first 5 iterations and converged to the optimal solution of 198.5 within 28 iterations, demonstrating extremely fast convergence speed and excellent global search capability. In contrast, although HHO also has a faster downward trend, its overall convergence speed is slower, requiring 41 iterations to reach 205.2, and the solution quality is significantly inferior to AVDHHO. The convergence speed of MA and GA is slower, reaching 213.6 and 221.3 after 55 and 63 iterations respectively, and the descent process is relatively smooth, indicating that the algorithm relies heavily on local search and has insufficient exploration ability.

In terms of stability, the standard deviation of AVDHHO is only 0.18, with the smallest fluctuation and the best stability of the solution; The standard deviations of HHO, MA, and GA are 0.32, 0.43, and 0.51, respectively, indicating greater volatility in the solutions. The convergence curve shows that AVDHHO remains stable in the middle and later stages, while other algorithms exhibit varying degrees of oscillation. Overall, AVDHHO is significantly superior to the compared algorithms in terms of optimal solution quality, convergence speed, and robustness, making it a better choice for solving this scheduling problem.

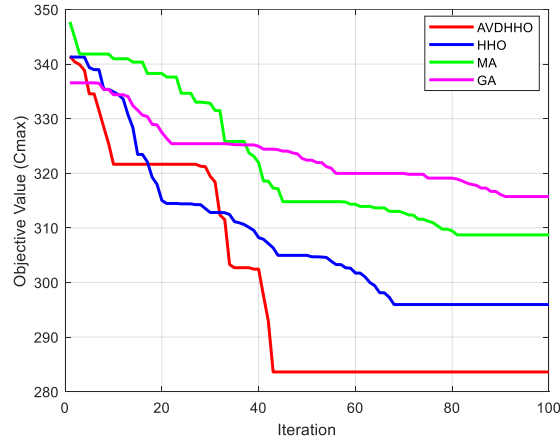


Figure 7 Convergence curves of different algorithms in P3 environment

In the P3 ($20 \times 5 \times 3$) example, the convergence curves of each algorithm show more significant performance differences, as shown in Figure 7. The target value of AVDHHO rapidly decreased in the first 40 iterations, and reached the optimal solution of 283.7 in the 43rd iteration, with a convergence rate of only 0.25%, significantly better than HHO, MA, and GA. In contrast, HHO only tended to stabilize after the 68th iteration, with a final value of 296.4, slightly higher than AVDHHO; The convergence speed and optimal solution of MA and GA are both poor, reaching 308.7 and 315.6 in 82 and 91 iterations, respectively. The overall convergence curve decreases slowly, indicating insufficient search ability in larger scale problems.

From the perspective of convergence process, AVDHHO exhibits a significant stepwise decline in the mid-term (20-40 iterations), and quickly stabilizes in the low Cmax region in the later stage, demonstrating strong global and local search balance ability. Although HHO also has a significant decline stage, its final solution quality and convergence rate are inferior to AVDHHO; The MA and GA curves have significant fluctuations, slow convergence, and poor stability, with convergence rates as high as 0.52% and 0.65%, respectively. Overall, AVDHHO still maintains high solution quality and convergence efficiency under the scale of this problem.

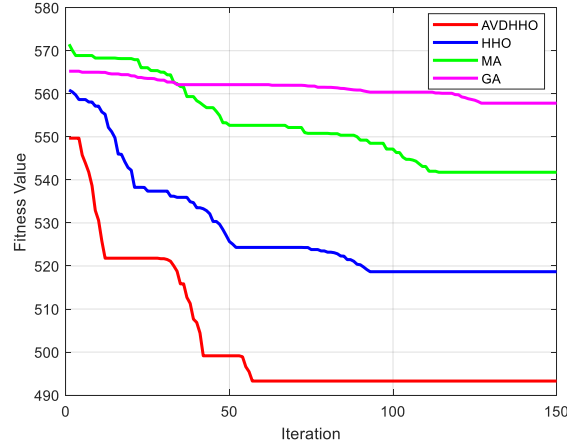


Figure 8 Convergence curves of different algorithms in P5 environment

In the P5 ($50 \times 5 \times 5$) large-scale example, the convergence performance differences of the four algorithms further widen, as shown in Figure 8. AVDHHO achieved a significant decrease in the first 50 iterations and converged to the optimal solution of 492.6 in the 57th iteration, with a convergence rate of 0.36%. It still demonstrates strong search and convergence capabilities in large-scale complex problems. The convergence speed and solution quality of HHO are second, converging to 518.7 in the 93rd iteration with a convergence rate of 0.57%. Although there have been several significant decline stages in the mid-term, the overall superiority and inferiority of HHO are significantly inferior to AVDHHO.

The performance of MA and GA is more limited, with MA stabilizing at 541.2 after the 114th iteration and GA reaching 557.8 after the 127th iteration, with convergence rates of up to 0.69% and 0.83%, respectively. The solution quality is significantly inferior to AVDHHO and HHO. The convergence curve clearly shows that AVDHHO rapidly decreases multiple times in the early and middle stages, and stabilizes at the optimal solution in the later stage, while GA and MA decrease slowly and fluctuate greatly throughout the entire search process, indicating that their convergence efficiency and robustness in dealing with large-scale multi-stage scheduling problems are insufficient. Overall, AVDHHO still has the most advantages in terms of optimal solution quality, convergence speed, and stability.

3) Ablation experiment

In order to verify the effectiveness of the proposed Adaptive Mutation Driven Harris Hawk

Optimization Algorithm (AVDHHO), the authors designed four sets of ablation experiments to compare and analyze the performance contributions of each key component (see Figure 9). The experiment was conducted on three scales of mixed flow workshop problems with robotic arms, using the same parameters and termination conditions for all algorithms. The results showed that compared with the standard HHO, AVDHHO improved the optimal and average solutions by 13.2% and 15.4% respectively in small-scale problems, and by 13.1% and 15.1% respectively in large-scale problems, indicating that the improved strategy has more significant advantages in complex problems.

From fixed mutation to adaptive mutation, the stability of the algorithm has been significantly enhanced, with a standard deviation reduction of 42.6% to 45.2%, verifying the important role of adaptive mutation mechanism in balancing global exploration and local development. Levy's flight disturbance also significantly improved the global search capability, with an average solution improvement of over 8%. Although AVDHHO has slightly increased convergence time, considering the improvement in solution quality, this cost is reasonable. Overall, the synergistic effect of the two core components of adaptive mutation and Levy flight is the key source of the algorithm's outstanding performance.

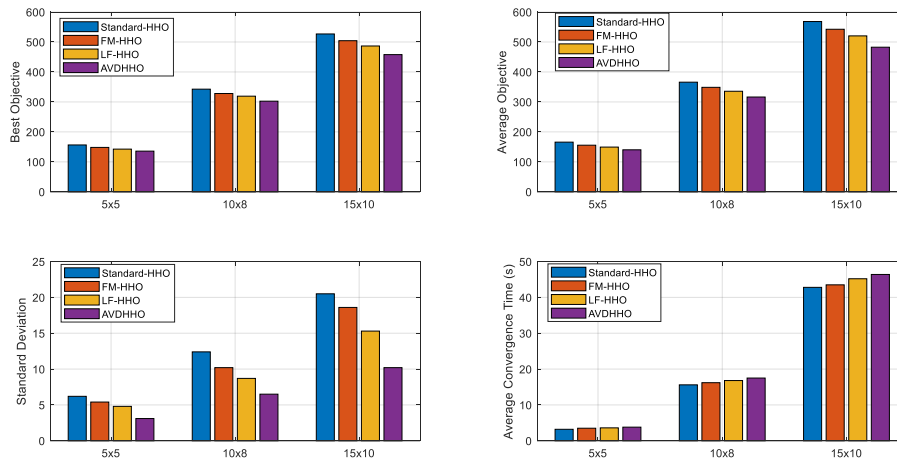


Figure 9 Results of ablation experiment

4) Actual factory case verification

In order to verify the application effect of the proposed Adaptive Mutation Driven Harris Hawk Optimization Algorithm (AMDHHO) in actual production environments, the author selected A Precision Parts Manufacturing Enterprise in Suzhou, Jiangsu Province for case

verification. The enterprise mainly produces automotive transmissions and aviation components, with a typical mixed flow workshop production line (Table 5), equipped with 5 processes, 11 parallel processing equipment, and 3 robotic arms (Table 6). The experiment focuses on actual production batches of 20 workpieces, which need to complete turning, milling, drilling, grinding, and testing processes in sequence, with significant differences in processing time between each process.

Table 5 Workpiece Processing Time Data (minutes)

Workpiece Number	Turning process	Milling process	Drilling process	Grinding process	Testing process
J1	25	18	15	22	10
J2	28	20	16	24	12
J3	22	19	18	21	11
J4	30	21	17	26	13
J5	24	17	14	20	9
J6	27	22	19	25	14
J7	23	16	13	19	8
J8	31	23	20	27	15
J9	26	21	18	23	12
J10	29	24	21	28	16
J11	24	19	16	22	11
J12	27	22	19	25	14
J13	25	20	17	23	12
J14	30	25	22	29	17
J15	28	23	20	26	15
J16	26	21	18	24	13
J17	29	24	21	27	16
J18	23	18	15	21	10
J19	32	26	23	30	18
J20	25	20	17	23	12

Table 6 Operating time of robotic arm (minutes)

Robot arm number	Transfer between processes	Operating Time
R1	Loading - Turning	3.5
R1	Turning milling	4.2
R2	Milling - Drilling	3.8
R2	Drilling and Grinding	4.0
R3	Grinding - Inspection	3.6
R3	Testing - Cutting	3.2

The AMDHHO algorithm was applied to solve the scheduling problem, and the algorithm converged after 152 iterations to obtain the optimal scheduling solution. The maximum completion time was 683 minutes, and the workpiece processing sequence and equipment allocation plan are shown in Table 7. Compared with the original manual scheduling plan of the enterprise, this plan has shortened the maximum completion time by about 85 minutes (an increase of 11.1%), reduced the average waiting time by 39.8%, and increased the resource utilization rate to 82.7%. In simulating two device failure scenarios, the algorithm only takes about 9.6 seconds to generate an adjusted new scheduling plan, verifying its efficiency and adaptability in dynamic production environments.

Table 7 Workpiece processing sequence and equipment allocation for optimal scheduling scheme

Process	Device 1	Device 2	Device 3
Turning process	J5→J1→J17→J9	J7→J3→J12→J20	J11→J16→J2→J8→J4→J14→J19→J18→J10→J6→J13→J15
Millings	J5→J7→J3→J1→J11→J18→J16→J9→J13→J20→J2	J12→J17→J6→J10→J15→J8→J4→J14→J19	-
Drillings	J5→J7→J3→J1→J11→J18→J16→J2→J9→J13→J4→J19	J12→J17→J6→J10→J15→J8→J20→J14	-
Grinding	J5→J7→J3→J1→J11→J18→J16→J2→J17→J13→J9	J12→J6→J10→J15→J8→J20→J4→J14→J19	-
Testing	J5→J7→J3→J1→J11→J18→J16→J2→J17→J13→J9→J12→J6	J10→J15→J8→J20→J4→J14→J19	-

6. Conclusion

The author proposes the Adaptive Mutation Driven Harris Hawk Optimization Algorithm (AMDHHO) for the Hybrid Flow Workshop Scheduling Problem with Robotic Arm (HFSP-LUM), and systematically validates it through standard examples and actual production data. In the 5-stage standard case study experiment, AMDHHO achieved optimal solutions on all scale problems. Taking the largest P5 example (50 workpieces x 5 stages x 5 devices) as an example, the maximum completion time of AMDHHO is 492.6 minutes, significantly lower than HHO (518.7 minutes), MA (541.2 minutes), and GA (557.8 minutes), and the standard deviation is only 3.27, indicating the strongest stability. In terms of convergence efficiency, AMDHHO achieved convergence in only 57 iterations, more than 70 times faster than GA, demonstrating excellent convergence speed.

In practical factory application verification, production data from a precision component manufacturing enterprise in Suzhou, Jiangsu Province was selected to schedule the processing tasks of 20 workpieces on 11 equipment and 3 robotic arms. The results showed that AMDHHO reduced the maximum completion time from 768 minutes to 683 minutes, improving scheduling efficiency by 11.1%. At the same time, the resource utilization rate reached 82.7%, which is about 15.3% higher than manual experience scheduling. The average waiting time for workpieces has been reduced from 18.6 minutes to 11.2 minutes, a decrease of 39.8%, fully demonstrating the effectiveness and practicality of the algorithm in real production environments. In addition, in dynamic scenarios simulating equipment failures, AMDHHO can generate new optimal scheduling solutions in an average of only 9.6 seconds, demonstrating excellent adaptability and stability. Overall, AMDHHO provides an efficient and robust optimization tool for complex workshop scheduling problems in the context of intelligent manufacturing, with broad engineering application prospects.

References

- [1] Boris Kupriyanov, Alexander Lazarev, Alexander Roschin & Frank Werner. (2025). On the Recursive Representation of the Permutation Flow and Job Shop Scheduling Problems and Some Extensions. *Mathematics*, 13 (19), 3185-3185. <https://doi.org/10.3390/MATH13193185>.
- [2] Jianlin Zhang, Longbin Ma, Wu Zhao, Jie Cao, Zuohan Chen & Tianpeng Xu. (2025). A

Deep Q-Network-driven multi-objective evolutionary algorithm for distributed heterogeneous hybrid flow shop scheduling with worker fatigue. *Engineering Applications of Artificial Intelligence*, 162 (PA), 112292-112292. <https://doi.org/10.1016/J.ENGAPPAI.2025.112292>.

[3] Yejian Zhao, Xiaochuan Luo, Weixiang Xu & Yulin Zhang. (2025). A deep reinforcement learning optimization algorithm based on heterogeneous graph neural network for hybrid flow shop scheduling problem with finite transportation resources. *Engineering Applications of Artificial Intelligence*, 161 (PB), 112096-112096. <https://doi.org/10.1016/J.ENGAPPAI.2025.112096>.

[4] Xuhui Zhu, Guocan Ma, Vedpal Arya, Pingfan Xia & Tingting Zhu. (2025). A novel dual-archive multi-objective algorithm for hybrid flow shop scheduling problems with missing operations. *Cluster Computing*, 28 (13), 815-815. <https://doi.org/10.1007/S10586-025-05513-8>.

[5] Zhaohui Zhang, Wanqiu Zhao, Hong Zhao & Xu Bian. (2025). MNATS: A Multi-Neighborhood Adaptive Tabu Search Algorithm for the Distributed No-Wait Flow Shop Scheduling Problem. *Applied Sciences*, 15 (17), 9840-9840. <https://doi.org/10.3390/APP15179840>.

[6] Yuan Zou, Chao Lu, Lvjiang Yin & Xiaoyu Wen. (2025). Multi-Level Subpopulation-Based Particle Swarm Optimization Algorithm for Hybrid Flow Shop Scheduling Problem with Limited Buffers. *Computers, Materials & Continua*, 84 (2), 2305-2330. <https://doi.org/10.32604/CMC.2025.065972>.

[7] Dan Yang Lv & Ji Bo Wang. (2025). Research on two-machine flow shop scheduling problem with release dates and truncated learning effects. *Engineering Optimization*, 57 (7), 1828-1848. <https://doi.org/10.1080/0305215X.2024.2372633>.

[8] Chuangfeng Zeng & Jianjun Liu. (2025). A metaheuristic algorithmic framework for solving the hybrid flow shop scheduling problem with unrelated parallel machines. *Engineering Optimization*, 57 (7), 1849-1872. <https://doi.org/10.1080/0305215X.2024.2372634>.

[9] Liangcai Xia, Shijun Chen, Weigang Zhou, Qian Wang & Jiying Xu. (2025). Dual evolutionary algorithm based on Dyna-Q for distributed heterogeneous hybrid flow shop problems with L-R trapezoidal fuzzy numbers. *Memetic Computing*, 17 (3), 35-35.

<https://doi.org/10.1007/S12293-025-00468-0>.

[10] Xinyi Yue, Xianqing Xiong, Mei Zhang & Xiutong Xu. (2025). Multi-attribute bottleneck identification method for hybrid flow shops in panel furniture intelligent manufacturing. *Complex & Intelligent Systems*, 11 (8), 362-362. <https://doi.org/10.1007/S40747-025-01993-8>.

[11] Huaixia Shi, Huaqiang Si & Jiyun Qin. (2025). Energy-Efficient Scheduling for Resilient Container-Supply Hybrid Flow Shops Under Transportation Constraints and Stochastic Arrivals. *Journal of Marine Science and Engineering*, 13 (6), 1153-1153. <https://doi.org/10.3390/JMSE13061153>.

[12] Jianguo Duan, Zixuan Liu, Mengting Wang, Yulin Du & Mengpei Yang. (2025). A problem-specific knowledge-based multi-objective algorithm for sustainable scheduling of distributed heterogeneous welding permutation flow shop. *Engineering Applications of Artificial Intelligence*, 156 (PA), 111197-111197. <https://doi.org/10.1016/J.ENGAPPAL.2025.111197>.

[13] Aslihan Cubukcuoglu, Ismet Karacan, Zeynep Ceylan & Serol Bulkan. (2025). Minimizing Makespan in Ordered Flow Shop Scheduling Using a Robust Genetic Algorithm. *Processes*, 13 (5), 1583-1583. <https://doi.org/10.3390/PR13051583>.

[14] Ouchene Randa, Rebaine Djamel & Baptiste Pierre. (2025). The impact of missing operations on the flow shop: permutation vs. non permutation modes. *RAIRO - Operations Research*, 59 (3), 1273-1293. <https://doi.org/10.1051/RO/2025032>.

[15] Xiyang Liu & Fangjun Luan. (2025). Improved black widow optimization algorithm for multi-objective hybrid flow shop batch-scheduling problem. *Journal of Combinatorial Optimization*, 49 (3), 42-42. <https://doi.org/10.1007/S10878-025-01270-X>.

[16] Lotfi HIDRI. (2025). Solving Multi-stage Flexible Flow Shop Scheduling Problem with Cooling Times in the Steel Industry. *Studies in Informatics and Control*, 34 (1), <https://doi.org/10.24846/V34I1Y202504>.

[17] Johanna Mlekusch & Richard F. Hartl. (2025). The dual-resource-constrained re-entrant flexible flow shop a constraint programming approach and a hybrid genetic algorithm. *International Journal of Production Research*, 63 (5), 1803-1824. <https://doi.org/10.1080/00207543.2024.2392198>.

- [18] Yanguang Cai & Huajun Chen. (2025). An improved salp swarm algorithm for permutation flow shop vehicle routing problem. *Scientific Reports*, 15 (1), 6704-6704. <https://doi.org/10.1038/S41598-025-86054-3>.
- [19] Fangchi Zeng & Junjia Cui. (2025). Improved Fruit Fly Algorithm to Solve No-Idle Permutation Flow Shop Scheduling Problem. *Processes*, 13 (2), 476-476. <https://doi.org/10.3390/PR13020476>.
- [20] Qiwen Zhang & Tian Zhen. (2025). Improved Jaya algorithm for energy-efficient distributed heterogeneous permutation flow shop scheduling. *The Journal of Supercomputing*, 81 (2), 434-434. <https://doi.org/10.1007/S11227-025-06938-Z>.
- [21] Tristan Becker, Janis Neufeld & Udo Buscher. (2025). The distributed flow shop scheduling problem with inter-factory transportation. *European Journal of Operational Research*, 322 (1), 39-55. <https://doi.org/10.1016/J.EJOR.2024.10.026>.
- [22] Binhui Wang & Hongfeng Wang. (2025). An Iterated Greedy Algorithm with Memory and Learning Mechanisms for the Distributed Permutation Flow Shop Scheduling Problem. *Computers, Materials & Continua*, 82 (1), 371-388. <https://doi.org/10.32604/CMC.2024.058885>.
- [23] Shrikant Panwalkar & Christos Koulamas. (2025). An asymptotically optimal solution for the minimization of the variation of job completion times in two-stage proportionate no-wait flow shops. *Journal of the Operational Research Society*, 76 (1), 131-136. <https://doi.org/10.1080/01605682.2024.2330614>.
- [24] Tasnim Mraïhi, Olfa Belkahla Driss & Hind Bril El Haouzi. (2025). An Iterated Greedy Algorithm for the Distributed Permutation Flow Shop Scheduling Problem with Worker Flexibility. *IFAC PapersOnLine*, 59 (10), 2904-2909. <https://doi.org/10.1016/J.IFACOL.2025.09.488>
- [25] Yu Zheng, Ningtao Peng, Hao Qi, Guiliang Gong, Dan Huang, Kaikai Zhu... & Gonggang Liu. (2025). An improved memetic algorithm for distributed hybrid flow shop scheduling problem with operation inspection and reprocessing. *Measurement and Control*, 58 (1), 68-96. <https://doi.org/10.1177/00202940241245241>.
- [26] Hajar Sadki & Karam Allali. (2024). Artificial neural network for solving flow shop optimization problem with sequence independent setup time. *Foundations of Computing and*

Decision Sciences, 49 (4), 355-383. <https://doi.org/10.2478/FCDS-2024-0018>.

[27] Yongxin Lu, Yiping Yuan, Adilanmu Sitahong, Yongsheng Chao & Yunxuan Wang. (2024). An Optimization Method for Green Permutation Flow Shop Scheduling Based on Deep Reinforcement Learning and MOEA/D. *Machines*, 12 (10), 721-721. <https://doi.org/10.3390/MACHINES12100721>.

[28] Hossein Badri, Tayebah Bahreini & Daniel Grosu. (2024). Parallel shifting bottleneck algorithms for non-permutation flow shop scheduling. *Annals of Operations Research*, 343 (1), 1-27. <https://doi.org/10.1007/S10479-024-06329-2>.

[29] Wen Bin Zhao, Jun Han Hu & Zi Qiao Tang. (2024). Virtual Simulation-Based Optimization for Assembly Flow Shop Scheduling Using Migratory Bird Algorithm.. *Biomimetics* (Basel, Switzerland), 9 (9), 571-571. <https://doi.org/10.3390/BIOMIMETICS9090571>.

[30] Yarong Chen, Junjie Zhang, Mudassar Rauf, Jabir Mumtaz & Shenquan Huang. (2024). Dynamic scheduling of hybrid flow shop problem with uncertain process time and flexible maintenance using NeuroEvolution of Augmenting Topologies. *IET Collaborative Intelligent Manufacturing*, 6 (3), e12119-e12119. <https://doi.org/10.1049/CIM2.12119>.

[31] Hui Mu, Zinuo Wang, Jiaqi Chen, Guoqiang Zhang, Shaocun Wang & Fuqiang Zhang. (2024). A Flow Shop Scheduling Method Based on Dual BP Neural Networks with Multi-Layer Topology Feature Parameters. *Systems*, 12 (9), 339-339. <https://doi.org/10.3390/SYSTEMS12090339>.

[32] Roderich Wallrath & Meik B. Franke. (2024). Integrating MILP, Discrete-Event Simulation, and Data-Driven Models for Distributed Flow Shop Scheduling Using Benders Cuts. *Processes*, 12 (8), 1772-1772. <https://doi.org/10.3390/PR12081772>.

[33] Dvir Shabtay & Enrique Gerstl. (2024). Coordinating scheduling and rejection decisions in a two-machine flow shop scheduling problem. *European Journal of Operational Research*, 316 (3), 887-898. <https://doi.org/10.1016/J.EJOR.2024.03.021>.

[34] Lizhen Du, Xintao Wang, Jiaqi Tang, Chuqiao Xu & Guanxing Qin. (2024). Improved hybrid estimation of distribution algorithm for distributed parallel assembly permutation flow shop scheduling problem. *IET Collaborative Intelligent Manufacturing*, 6 (3), e12116-e12116. <https://doi.org/10.1049/CIM2.12116>.

[35] Omar Nejjarou, Said Aqil & Mohamed Lahby. (2024). Hybrid meta-heuristic solving no-wait flow shop scheduling minimizing maximum tardiness. *Evolutionary Intelligence*, 17 (5-6), 3935-3959. <https://doi.org/10.1007/S12065-024-00965-0>.