MAPAN-JOURNAL ,
Publisher: METROLOGY SOC INDIA , NPL PREMISES,
Address: DR K S KRISHNAN MARG, NEW DELHI, INDIA,
ISSN / eISSN:0970-3950 / 0974-9853
Volume 26-Issue 1-(2026)
ARTICLE TYPE DOI: https://doi-001.org/1025/17689242268007

A Hybrid EAX-Enhanced NSGA-II Algorithm for Multi-Objective Flexible Job Shop Scheduling

Authors: **Jiake Wu**[1,a] , **Kewei Chen**[1,b,*]

Affiliation:
[1]Faculty of Mechanical Engineering & Mechanics, Ningbo University, Ningbo 315000, Zhejiang, China

All authors' email:
[a]Email: 2311090072@nbu.edu.cn
[b]Email: chenkewei@nbu.edu.cn

All authors' bio:

Jiake WU received the B.E. degree from Ningbo University, Zhejiang, China, in 2023. She is currently pursuing the M.E. degree with Ningbo University, Zhejiang. Her research focuses on the flexible workshop scheduling problem and optimization algorithms.

KEWEI CHEN is currently a Professor with Ningbo University. His main research interests include intelligent manufacturing, robotics, and computational intelligence.

Abstract

Aiming at the pain point that destructive crossover operations in traditional genetic algorithms damage key scheduling edge structures in the multi-objective flexible job shop scheduling problem, an improved genetic algorithm based on the Edge Assembly Crossover operator is proposed.In the crossover phase, a parent-cycle recombination strategy is employed to systematically preserve machine-task adjacency relationships while exploring the solution space in a structured way. Then, the improved algorithm is integrated with the non-dominated sorting framework based on NSGA-II. By protecting key adjacency relationships during the crossover process, the convergence of the Pareto front and the diversity of solutions are significantly enhanced. Through experiments on the MK01-10 and LA01-40 datasets, the improved EAX-GA algorithm demonstrates significant advantages in both convergence results and convergence speed.

KEYWORDS

flexible job shop problem, multi-objective optimization, edge assembly crossover, energy saving

1 | INTRODUCTION

Manufacturing serves as a vital cornerstone of the modern economy. Optimizing the Job Shop Scheduling Problem enables simultaneous leaps in production efficiency and maximization of resource utilization. Traditional JSP approaches struggle to meet contemporary manufacturing demands. The Flexible Job Shop Scheduling Problem (FJSP), as an extension of JSP, offers greater practical applicability. As research depth and problem complexity expand, various specialized FJSP models have emerged for specific scenarios, including: real-time scheduling with order insertion or process changes [1,2], dual-resource collaborative constraint optimization, and trade-offs between energy conservation and production efficiency. Genetic algorithms demonstrate applicability in most of these scenarios. To enhance their problem-specific solution efficiency, algorithmic architectures are typically tailored through targeted improvements [3], leading to the systematic integration of diverse improvement operators and adaptive mechanisms within the genetic algorithm framework. Chowdhury et al. [4] introduced three key methods into genetic algorithms: hybrid crossover-mutation operators, circular parent topology, and a ternary selection protocol with family constraints, significantly improving the efficiency of final solutions. Jiang et al. [5] refined the single mutation approach in genetic algorithms by designing four mutation operators based on process coding and machine coding. By updating their weights according to performance during iterations and adjusting their selection mutation operators, they enhanced local search capabilities and accelerated convergence speed.

Gonçalves et al. [6] proposed a hybrid genetic algorithm based on random key chromosomal coding. Its core advantage lies in dynamically balancing global search with local optimization, making it suitable for large-scale complex scheduling scenarios. Choi et al. [7] developed a variable neighborhood search algorithm based on hybrid genetic algorithms (GA-VNS) for scheduling problems in flexible parallel machine workshops. Gu et al. [8] proposed an improved adaptive variable neighborhood search genetic algorithm (IGA-AVNS). The IGA-AVNS algorithm employs OS-MA hybrid initialization to generate populations, followed by group-parallel optimization. Optimal solutions are stored in an elite pool, enabling adaptive neighborhood local search. Its effectiveness was validated through experiments based on three standard benchmark problems. Beyond methods such as a variable neighborhood search in genetic algorithms, another targeted improvement strategy exists. The edge-based approach, initially proposed by Nagata et al. [9], has been demonstrated by Nikfarjam et al. [10] to be effective in solving large-scale Traveling Salesman Problems (TSP). However, no existing studies have applied this method to the flexible flow shop scheduling problem. This paper aims to fill this research gap. To evaluate its performance, the algorithm was tested on both single-objective and multi-objective variants of the flexible flow shop problem.

Wu et al. [11], based on the extended HFSP model, proposed a multi-objective evolutionary collaborative learning framework that combines evolutionary algorithms and reinforcement learning.A multi-resolution grid strategy, based on the traditional Bacterial Foraging Optimization (BFO) algorithm, was proposed by Ji et al. [12] to enhance the diversity and distribution of solution sets in multi-objective optimization problems. In contrast to the aforementioned approach, NSGA-II represents another mainstream methodology for addressing multi-objective optimization problems. Numerous studies have proposed improvements to the NSGA-II algorithm. [13] proposed an improved NSGA-III algorithm (I-NSGA-III) that used a good point set instead of random initialization to generate a more uniformly distributed initial population. [14] proposed a hybrid simulated binary and improved arithmetic crossover (SBAX) operator for bi-objective optimisation and applied it to NSGA-II to enhance its performance. [15] proposed NSGA-II algorithm integrated with the TOPSIS method to optimize parameters in biclustering algorithms for analyzing complex relationships in large datasets. [16] proposed a novel optimization model that integrates Fuzzy set theory with the NSGA-III to solve the time-cost trade-off (TCT) problem in construction projects. Results showed that the proposed Fuzzy-NSGA-III outperforms traditional multi-objective algorithms in convergence, diversity, and solution quality, supported by metrics including GD, HV, and QM.

In contrast to prior research, this study proposes an improved NSGA-II algorithm that incorporates an efficient crossover operator to improve convergence efficiency. The Edge Assembly Crossover (EAX) operator employs a recombination mechanism based on AB cycles combined with a local optimization strategy. The core of this hybrid method consists of: structurally decomposing parental edge information, constructing relaxed intermediate solutions, and guiding the generation of high-quality offspring through localized refinement. By adaptively extending AB cycle paths to enhance neighborhood search capabili- ties, EAX exhibits significant advantages in combinatorial optimization particularly in multi-objective optimization scenarios involving makespan and energy consumption.

The remainder of this paper is structured as follows: Section 2 formulates a mathematical model for flexible job shop scheduling. Section 3 elaborates on the enhanced NSGA-II algorithm integrated with a neighbor-edge pairing crossover operator, which leverages an AB-cycle recombination mechanism to strengthen global exploration within the solution space and achieve efficient convergence toward the Pareto front. Section 4 provides experimental validation of the proposed approach. Finally, Section 5 concludes the study and suggests directions for future research.

2 | PROBLEM DESCRIPTION OF FJSP

2.1 | Description and Assumptions of the Mathematical Model

The FJSP problem is specifically defined as processing N jobs (denoted as $J = J_1, J_2, J_3, ..., J_n$) sequentially on M machines (denoted as$M = M_1, M_2, M_3, ..., M_n$). Each job J_i (where$i = 1, 2, ..., N$) undergoes O_i processing steps, and the processing time for each step on each machine is known. The goal is to determine the optimal processing sequence while establishing the following assumptions for the mathematical model:

1. Once an operation begins on a machine, it cannot be interrupted until completion.
2. Each machine can process only one workpiece at a time; a workpiece can be processed by only one machine simultaneously.
3. Each operation must complete preceding operations before starting subsequent ones.
4. All jobs arrive at time zero, and all machines are available at time zero.
5. Transportation failures are ignored.
6. Sudden equipment failures and urgent orders are not considered.
7. Material path interference is ignored.

TABLE 1 parameters of FJSP

Parameter	Description
n	Number of workpieces to be processed
m	Number of machines available for processing
O_{ij}	Process j for workpiece i
S_{ijk}	Start time of operation O_ij on machine k
F_{ijk}	Finish time of operation O_ij on machine k
T_{ijk}	Processing time required for operation O_ij on machine k
U_{ijk}	Boolean variable determining whether operation O_ij is performed on machine k
Q_{abij}	Variable representing the sequence relationship between operations O_ab and O_ij
makespan	Maximum completion time

2.2 | Objective Function and Constraints

Mathematical symbols for model parameters are defined in detail in Table 1. The variable indices in the table are defined as follows: $i \in \{1, 2, ..., n\}, j \in \{1, 2, ..., O_i\}, k \in \{1, 2, ..., m\}$. The constraints correspond to the following six equations: Equation (1): The completion time of any operation equals the sum of its start time and processing time, indicating that the manufacturing process is non-preemptive operations proceed continuously from initiation to completion. Equation (2): Ensures each operation is assigned to exactly one machine. Equation (3): The maximum completion time of the entire scheduling plan must exceed the completion time of any single operation. Equation (4): Only one operation can be processed on the same machine at the same time. Equation (5): The completion time of a preceding operation on the same workpiece must strictly precede the start time of a subsequent operation to guarantee the temporal validity of the production sequence. Equation (6): All time variables must satisfy non-negativity constraints.

$$F_{ijk} = S_{ijk} + T_{ijk} \cdot U_{ijk} \quad \forall i, j, k \tag{1}$$

$$\sum_{k=1} U_{ijk} = 1 \quad \forall i, j \tag{2}$$

$$F_{ijk} \leq makespan \quad \forall i, j, k \tag{3}$$

$$U_{aby} \cdot F_{aby} + m \cdot (Q_{abij} - 1) < U_{ijk} \cdot S_{ijk} \tag{4}$$

$$F_{ijk} \leq S_{i(j+1)k'} \quad \forall i, j(5) \tag{5}$$

$$F_{ijk} \geq 0, S_{ijk} \geq 0, T_{ijk} \geq 0 \quad \forall i, j, k \tag{6}$$

This paper aims to minimize both the maximum completion time and total energy consumption. Reducing the maximum completion time typically requires increasing parallelism or employing high-performance machines, which often exhibit higher energy intensity. Conversely, lowering energy consumption necessitates selecting low-power devices or extending task execution cycles. These objectives present a time-energy tradeoff that cannot be simultaneously optimized through a single solution. The mathematical expression for minimizing the maximum completion time is as follows:

$$\min(makespan) = \max F_i \quad \mathrm{i} \in \{1, 2, \ldots, \mathrm{n}\} \tag{7}$$

The total energy consumption of a machine comprises two components: processing energy consumption and idle energy consumption. Processing energy consumption represents resource expenditure during workpiece machining, exhibiting a positive correlation with processing time and machine processing power:

$$E_{proc} = \sum_{ijk} T_{ijk} \cdot U_{ijk} \cdot P_k^{proc} \quad \forall i, j, k \tag{8}$$

Idle energy consumption denotes standby consumption when the machine is powered on but not machining workpieces, exhibiting a positive correlation with idle time and machine idle power:

$$E_{idle} = \sum_{k} (F_{max} - \sum_{ij} T_{ijk} \cdot U_{ijk}) \cdot P_k^{idle} \quad \forall i, j, k \tag{9}$$

The total energy consumption of the machine is the sum of the above two components.

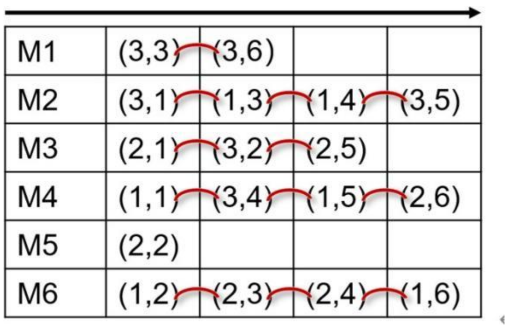


FIGURE 1 Schematic diagram of the adjacent edge concept

3 | IMPROVED GENETIC ALGORITHM BASED ON EDGE ASSEMBLY CROSSOVER OPERATOR

3.1 | Machine Instruction Encoding

Solutions to the FJSP scheduling problem can be mapped to a chromosome encoding structure, which represents the task execution sequence of each machining device through a multidimensional vector. Task allocation for a single device is encoded using a one-dimensional vector. Table 2 presents a small-scale example to illustrate the problem characteristics. This example involves three workpieces, each requiring six machining operations. This example involves three workpieces, each requiring six processing steps. As shown in the table, each step can be executed on different machines, with processing time varying based on equipment selection. For instance, the first step of Workpiece 2 can be performed on Machine 1 or Machine 3. It should be noted that the configuration in this example, where the number of processing machines exceeds the number of workpieces, represents a non-typical discrete manufacturing scenario.

TABLE 2 An instance of FJSP

	Process 1	Process2	Process 3	Process 4	Process5	Process6
Workpiece 1	4 or 3	6 or 3 or 2	6 or 2 or 1	2	4 or 2	6
Workpiece 2	3 or 1	5 or 3 or 2	6	6 or 2 or 1	3	6 or 4
Workpiece 3	2	3	1	4or 2	6 or 2	1

Machine-task assignments are encoded using a sparse mapping structure. Specifically, the indices i and j of process $O_{(ij)}$ are defined as the horizontal and vertical coordinates in a two-dimensional coordinate system. Based on the example in Table 2, one possible solution for the task queue is illustrated in Figure 1. Each machine processes parts in left-to-right order. An element connecting two adjacent vertices is termed an edge, represented by red arcs in Figure 1. Edges sharing a common vertex form an adjacency relationship. This adjacency concept provides the theoretical foundation for the proposed adjacent-edge assembly intersection operator.

3.2 | Population Initialization

The conventional NSGA-II algorithm relies on stochastic population initialization. This approach frequently induces an uneven distribution of individuals across the search space, characterized by overcrowding in certain areas and scarcity in others. Such distributional imbalance can decelerate the convergence rate and increase the propensity for the algorithm to become trapped in local optima in later stages. To mitigate these limitations, an enhanced initialization strategy utilizing the concept of a well-pointed set is introduced. Ensuring a homogeneous initial distribution of individuals is crucial for accelerating convergence and improving the quality of the final solutions. Following the definition provided by He et al. [17], a well-pointed set is characterized by these criteria:

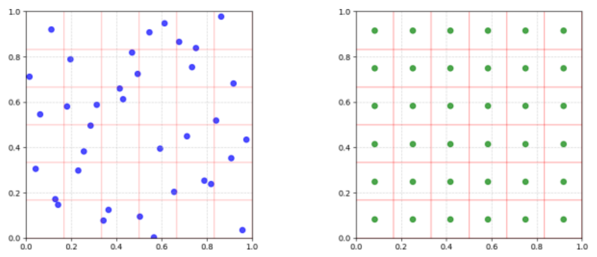


FIGURE 2 2D initial population distribution

1. Consider the hypercube G_s, which is the set of all points x in an S-dimensional Euclidean space.

$$x = x_1, x_2, ..., x_s \in G_s, 0 \leq x_j \leq 1, j = 1, 2, ..., s \tag{10}$$

2. Within G_s, a set $P_n(i)$ comprising n points is defined, expressed as:

$$\begin{aligned} P_n(i) &= x^{(1)}, x^{(2)}, ..., x^{(n)} \\ x^{(i)} &= x_1^{(i)}, x_2^{(i)}, ..., x_j^{(i)}, ..., x_s^{(i)} \end{aligned} \tag{11}$$

for indices $i = 1, 2, ..., n$ and $j = 1, 2, ..., s$.

3. For a given point $r = (r_1, r_2, ..., r_s) \in G_s$, the $M_n(r)$ quantifies the number of points within $P_n(i)$ that adhere to the constraint:

$$0 \leq x_j^{(i)} \leq r_j, j = 1, 2, ..., s \tag{12}$$

$$\varphi(n) = \sup_{r \in G_s} |\frac{M_n(r)}{n} - |r|, |r| = r_1 r_2 r_3 \cdot \cdot \cdot r_s, \tag{13}$$

$\varphi(n)$ is the discrepancy of points set $P_n(i)$. $P_n(i)$ is therefore considered to be uniformly distributed on G_s that has a deviation of $\varphi(n)$.

4. The designation of a point $r \in G_s$ as "good" is contingent upon the bounded discrepancy of its associated point set P_n, $\varphi(n) \leq C(r, \epsilon).n^{-1+\epsilon}$.

5. For practical implementation, a common construction heuristic is defined by equation(14).

$$r_j = 2 \cos \frac{2\pi j}{p}, 1 \leq j \leq s \tag{14}$$

where p is the least prime satisfying $p \geq 2s + 3$.

Two distribution plots were generated for 36 populations: one using a random method and the other using a good point set method. As shown in Figure 2, the point distribution from the good point sequence is more uniform than that from the random point sequence. Although the good point set requires more execution time, its impact on the overall algorithm cost is minimal.

3.3 | EAX Operator

Crossover, as a core mechanism in evolutionary computation, simulates biological genetic recombination through structured information exchange between parent chromosomes to enhance solution quality. By randomly combining parental gene fragments, offspring can inherit desirable traits while exploring new regions of the solution space. The neighbor-edge crossover operator demonstrates unique advantages in combinatorial optimization: (1) Systematically preserves critical edge information from parent solutions, ensuring solution feasibility; (2) The AB-cycle mechanism enables non-destructive recombination of machine-task assignments; (3) Dynamic neighborhood evaluation guides the crossover process toward advantageous search directions, accelerating convergence toward the Pareto optimal frontier. After population initialization, each individual stores individual storage information containing the scheduling sequences for all machines. For each machine, extract the task sequence T_m.The process of finding the AB-cycle between parent A and parent B is shown in the figure 3.

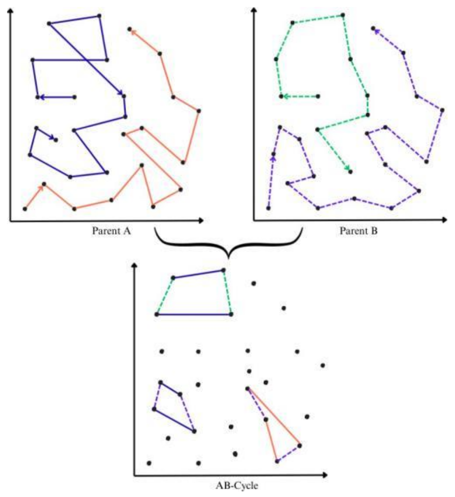


FIGURE 3 The process of forming the AB-cycle

After population initialization, the individual storage information contains the scheduling sequences for all machines. First, two individuals are selected as parents A and B. Then, by traversing each scheduling sequence, adjacent edges are generated and stored in the adjacent edge sets, labeled as E_A and E_B. Combining the adjacent edge assembly crossover operator, the pseudocode is as follows: After constructing the intermediate solution in Step 3, its feasibility must be verified: First, check against the constraint function to avoid conflicts in machine allocation or process sequence caused by path reorganization. If conflicts or infeasibilities exist, trace back from the conflict point to reassign machines for conflicting tasks or adjust their schedules. If tracing back fails to resolve the conflict, perform local search optimization to find the solution with the smallest disturbance correction. After correction, merge the subtours for the new generation.

3.4 | Non-Dominant Sorting Framework Based on EAX Operators

Multi-objective optimization aims to achieve the coordinated optimization of multiple competing objectives. When conflicts arise between objectives, the complexity of the problem increases significantly. The solution process for such optimization problems inevitably generates a set of trade-off optimal solutions, collectively referred to as the Pareto optimal solution set. A non-dominated relationship implies that if solution A is no worse than solution B across all objectives and strictly outperforms B on at least one objective, then A dominates B. Through dominance relation identification and recursive hierarchical partitioning, the algorithmic complexity is re- duced from $O(mn^3)$ to $O(mn^2)$,where m denotes the number of objectives and n represents the population size. This study proposes a novel fusion of the NSGA-II with EAX operator to address workflow scheduling challenges in multi- objective optimization scenarios. The merged framework leverages EAX's edge recombination mechanism to enhance the global search capability within the NSGA-II evolutionary architecture, achieving effective convergence toward high-quality Pareto-optimal scheduling solutions. In workflow scheduling optimization, each candidate solution is encoded as a chromo- some, where individual genes represent discrete workflow tasks. Allele values at each locus correspond to specific identifiers. The evolutionary population consists of N chromosomes. During the crossover process, the EAX operator replaces the traditional genetic crossover operator. Figure 4 illustrates the specific flowchart.

To ensure diversity in population distribution, the crowding distance is introduced. The crowding distance $D_{i,j}$ between chromosomes i and j within the same class is defined as follows.

$$\begin{aligned} D_{ij} =& |Makespan(i) - Makespan(j)| \\ &+ |E_{sum}(i) - E_{sum}(j)| \end{aligned} \tag{15}$$

$$C_d = \min\{D_{i1}, D_{i1}, \cdots, D_{in}\} \ (j \neq k) \tag{16}$$

Algorithm 1 Edge Assembly Crossover (EAX)

```
Require: Two parent tours $p_a$ and $p_b$
Ensure: Offspring tour
Let $E_a$ and $E_b$ be the edge sets of $p_a$ and $p_b$ respectively
Construct an undirected multigraph $G$ from $E_a$ and $E_b$
Step 1: Label edges
for each edge in $E_a$ do
    Label the edge as belonging to $E_a$
end for
for each edge in $E_b$ do
    Label the edge as belonging to $E_b$
end for
Step 2: Construct AB-cycles
Initialize an empty set $E_{\mathrm{set}}$ for AB-cycles
while there are unvisited nodes in $G$ do
    Select a random unvisited node $v$ as starting point
    Initialize an empty cycle $C$
    $current \rightarrow v$
    $first \rightarrow$ true
    repeat
        if $first$ is true then
            Select an edge $(current, u)$ from $E_a$
            $first \rightarrow$ false
        else
            Alternate between selecting edges from $E_a$ and $E_b$
        end if
        Add the selected edge to $C$
        $current \rightarrow u$
        Mark the edge as visited
    until return to the starting node $v$
    if $C$ is a valid AB-cycle (not consisting of two overlapping edges) then
        Add $C$ to $E_{\mathrm{set}}$
    end if
end while
Step 3: Construct intermediate solution
$t \rightarrow p_a$
for each AB-cycle in $E_{\mathrm{set}}$ do
    Remove edges in $E_a \cap$ AB-cycle from $t$
    Add edges in $E_b \cap$ AB-cycle to $t$
end for
Step 4: Merge subtours
Identify all subtours in $t$
while number of subtours $> 1$ do
    Select the smallest subtour $M$
    Find the optimal combination of edges $e^*, e^{*'}, e^{*''}, e^{*'''}$ that minimizes:
        $$-d(e) - d(e') + d(e'') + d(e''')$$
    Remove $e^*$ and $e^{*'}$ from $t$
    Add $e^{*''}$ and $e^{*'''}$ to $t$
end while
Step 5: Output offspring
return the resulting tour $t$
```

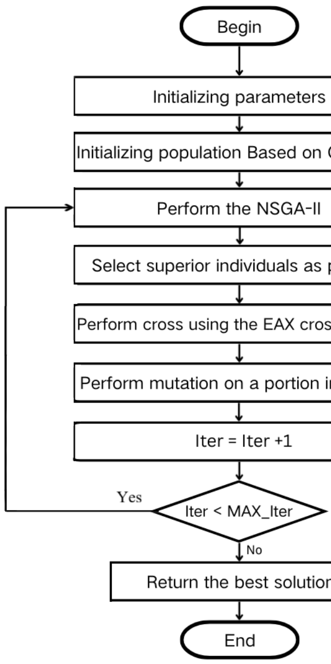

Begin
Initializing parameters
Initializing population Based on Good Set
Perform the NSGA-II
Select superior individuals as parents
Perform cross using the EAX cross operator
Perform mutation on a portion individuals
Iter = Iter +1
Yes
Iter < MAX_Iter
No
Return the best solution
End

FIGURE 4 Flowchart of proposed hybrid algorithm

Classify chromosomes based on dominance relationships, assigning ranks of 1, 2, ..., n. Calculate fitness using the following formula based on their rank k and crowding distance. After computing each individual's fitness, compare values to select superior individuals for inheritance into the next generation population.

$$fit(i) = \exp\left(\frac{k}{T_0} + \frac{\exp\left(-\frac{C_d(i)}{T_0}\right)}{1+\exp\left(-\frac{C_d(i)}{T_0}\right)}\right) \tag{17}$$

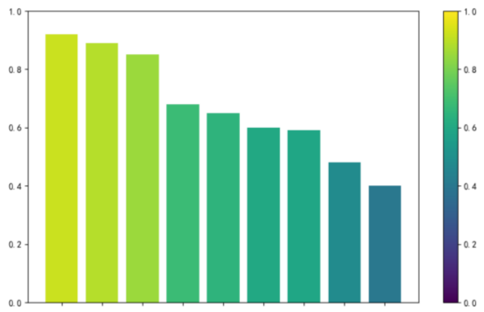
For generating a set of Pareto optimal solutions to multi-objective optimization problems, this paper proposes a method combining the Entropy Weight Method (EWM) [18] and the Two-Objective Performance Selection by Similarity to Ideal Solution (TOPSIS) [19]. The Entropy Weight Method is an objective weighting approach based on information entropy theory, which determines weights through the variability of indicator data, thereby eliminating biases caused by subjective weighting. Lower information entropy indicates reduced uncertainty in an indicator, resulting in a higher weight. Since the two objective functions defined in this study exhibit higher individual fitness with smaller numerical values, equation 18 is required to normalize the objective values within the Pareto solution set. In the formula, $x(ij)$ represents the j^{th} objective value of the i^{th} solution. Next, Equations 19 and 20 are used to calculate the information entropy. In Equation 20, e_j represents the information entropy of the j^{th} objective, and n denotes the number of solutions.Then, the target weights ω_j are calculated using information entropy to eliminate subjective weighting bias, as shown in Equation 21.

$$r_{ij} = \frac{\max(x_j) - x_{ij}}{\max(x_j) - \min(x_j)} \tag{18}$$

$$p_{ij} = \frac{x_{ij}}{\sum_{j=1}^{n} x_{ij}} \tag{19}$$

$$e_j = -\frac{1}{\ln n} \sum_{i=1}^{n} p_{ij} \ln p_{ij} \tag{20}$$

$$\omega_j = \frac{1 - e_j}{\sum_{j=1}^{m} (1 - e_j)} \tag{21}$$



F I G U R E 5 Visualization of index

After obtaining the respective weights, the rest procedure proceeds as follows: Step 1: Construct the weighted decision matrix by multiplying the standardized objective values by the weights calculated using the entropy weight method, as shown in Equation 22, where $r(ij)$ de- notes the standardized decision matrix element and $v(ij)$ denotes the weighted matrix element; Step 2: Determine the positive/negative ideal solutions using Equations 23 and 24. Step 3: Calculate the distance of each solution to the positive and negative ideal solutions. The distance to the positive ideal solution is given by Equation 25, and the distance to the negative ideal solution by Equation 26. Step 4: Calculate the proximity, defined as C_i. A higher proximity value indicates that the solution is closer to the positive ideal solution, signifying superior overall performance of the scheme in multi-objective decision-making.

$$v_{ij} = \omega_j \cdot r_{ij} \tag{22}$$

$$A^+ = (\max(v_{1j}), \max(v_{2j}), \ldots, \max(v_{mj})) \tag{23}$$

$$A^- = (\min(v_{1j}), \min(v_{2j}), \ldots, \min(v_{mj})) \tag{24}$$

$$D_i^+ = \sqrt{\sum_{j=1}^{n} (v_{ij} - A_i^+)^2} \tag{25}$$

$$D_i^- = \sqrt{\sum_{j=1}^{n} (v_{ij} - A_i^-)^2} \tag{26}$$

$$C_i = \frac{D_i^-}{D_i^+ + D_i^-} \tag{27}$$

After calculating the proximity scores for all solutions in the optimal solution set, they are sorted in descending order, with the top-ranked solution selected as the final optimal solution. This method balances the dimensional differences and varying importance of different objectives through standardization and weight allocation. Figure 5 presents a proximity-ranked bar chart illustrating the quality of optimal solutions, where the horizontal axis displays the reordered solution IDs from highest to lowest proximity, and the vertical axis represents proximity scores within the range from 0 to 1.

4 | EXPERIMENT

4.1 | Experimental Environment

To validate the effectiveness of the improved genetic algorithm in solving flexible job shop scheduling problems, we implemented a computational framework using Python 3.9 within the PyCharm 2022.3 development environment. The experimental platform utilized an AMD Ryzen 7 6800H processor (3.2 GHz base frequency, 4.7 GHz boost frequency) with integrated Radeon Graphics, operating on a 64-bit Windows 11 Professional architecture. All numerical experiments were conducted on this computational system. The parameters used in the experiment is shown.

Parameter	Value
Generations	500
Population size	200
Selection	Tournament selection
Crossover probability	90%
Maximum depth for crossover	10
Mutation probability	10%
Maximum depth for mutation	5

4.2 | Experiment 1

Experiment 1 focuses on solving single-objective problems by minimizing makespan. 40 benchmark instances of different scales were used to verify the effectiveness of EAX-GA. These instances were LA01 – LA40[20]. In algorithm evaluation, FJSP instances of varying problem sizes were analyzed and compared with classical algorithms for solving FJSP.

The results were compared in Table 3 other algorithms: Genetic Algorithm (GA),Hippopotamus Optimization Algorithm (HO), Tabu Search(TS), chaos-based improved MPA (CIMPA)and Improved Scatter Search (ISS) proposed by Holland et al. [21], Amiri et al. [22], Li et al. [23], Zhang et al. [24] and Wang et al. [25] respectively.

To mitigate randomness, each test was run 100 times. In most cases, the EAX-GA and CIMPA algorithms achieved relatively low makespan values, demonstrating the most outstanding and stable performance overall. The ISS algorithm followed closely, delivering comparable results to the former two in the majority of test instances. The HO and TS algorithms generally performed at an intermediate level. In contrast, the traditional GA yielded the highest makespan values among all six algorithms in most instances, indicating that its baseline version performed relatively poorly under the given experimental conditions.

For the FJSP problem, specially designed and enhanced hybrid algorithms such as EAX-GA and CIMPA, as well as intelligent search strategies like ISS, significantly outperformed traditional metaheuristics and other heuristic methods. In particular, EAX-GA and ISS exhibited exceptional robustness, with minimal fluctuations in solution quality across 40 test instances of varying scales and characteristics, and rarely showed significant performance degradation. Although CIMPA performed strongly in general, it exhibited noticeable deviations on specific instances such as LA06 and LA30 (with makespan values as high as 1596, substantially higher than other algorithms), suggesting slightly lower stability compared to EAX-GA and ISS.When using the optimal solution as a benchmark, EAX-GA outperformed the remaining algorithms in 18 out of 40 benchmark instances. Even when not surpassing other algorithms, it still achieved relatively strong solutions. The superior performance of EAX-GA is largely attributable to its effective crossover operator (EAX), which successfully inherits high-quality task sequences and machine allocation patterns from parent solutions while leveraging the global search ability of genetic algorithms. This enables a well-balanced trade-off between exploration and exploitation. These findings demonstrate that EAX-GA exhibits greater robustness.

4.3 | Experiment 2

Experiment 2 focuses on solving multi-objective problems.Since the multi-objective model incorporates the total energy consumption indicator, parameter settings are required to generate random parameters within the feasible operational range, as shown in Table 4.

To validate the overall performance advantage of the proposed EAX-NSGA-II algorithm in multi-objective flexible manufacturing scheduling, this study designed ablation experiments to control variables and evaluate the independent contributions of core components such as initialization of good set and the EAX crossover operator. To isolate the respective effects of greedy initialization and the EAX operator, four contrasting algorithm versions were designed. The traditional NSGA-II serves as the reference group, employing the Process Ordering Crossover (POX)([27]) mutation operator and classic crowding distance sorting ([28]). In contrast to the heuristic construction logic of good-set initialization, greedy initialization randomly generates both machine assignments for population individuals and process sequences. All algorithms maintained identical hyperparameter settings, test case sets, and experimental procedures. With maximum completion time and total machine energy consumption as optimization objectives, each combination underwent 40 independent runs.

To evaluate algorithm performance, optimal results were selected for comparative analysis. Line charts were plotted for each algorithm version against both objective functions. In the legends: dotted lines represent the greedy-initialized traditional

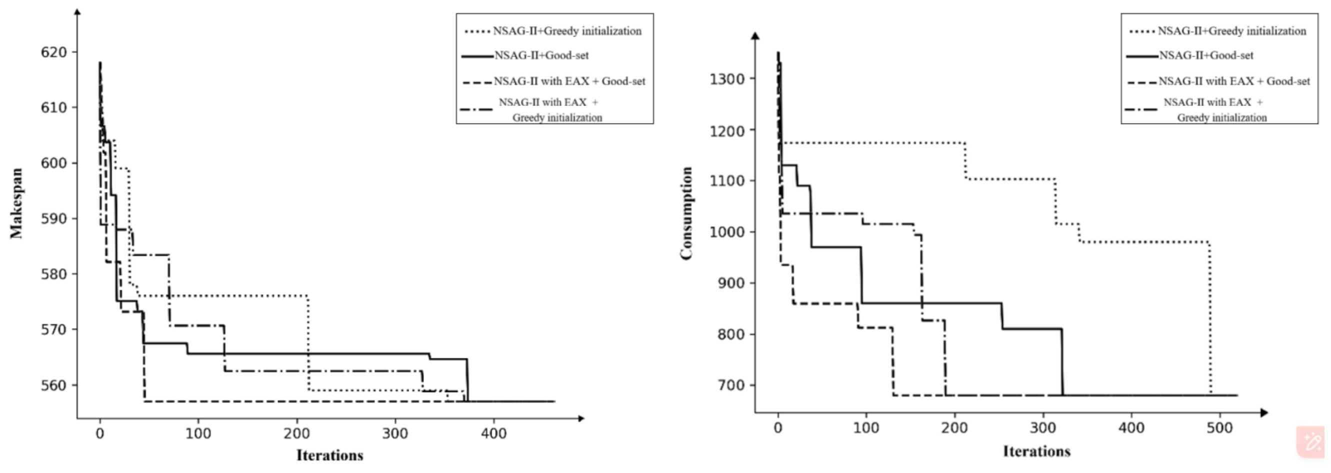
TABLE 3 LA Series Experimental Results Data Sheet

	Scale	EAX-GA	HO	TS	CIMPA	ISS	GA
LA 1	10x5	703	724	746	**675**	707	752
LA 2	10x5	692	728	748	**684**	721	761
LA 3	10x5	**620**	634	647	636	**620**	662
LA 4	10x5	632	635	646	**621**	625	667
LA 5	10x5	599	614	622	**593**	595	635
LA 6	15x5	**819**	861	878	926	821	892
LA 7	15x5	856	873	881	916	**848**	901
LA 8	15x5	**878**	924	948	874	882	961
LA 9	15x5	910	942	951	**900**	902	984
LA 10	15x5	956	965	979	958	**923**	1002
LA 11	20x5	**1070**	1105	1102	1222	1077	1148
LA 12	20x5	**1039**	1053	1061	1040	**1039**	1096
LA 13	20x5	1159	1163	1178	1150	**1117**	1197
LA 14	20x5	1302	1305	1310	1292	**1281**	1323
LA 15	20x5	**1210**	1231	1227	1281	1212	1251
LA 16	20x5	1034	1049	1052	1038	**1015**	1092
LA 17	20x5	**836**	867	875	842	838	906
LA 18	20x5	928	965	992	**923**	930	1009
LA 19	20x5	**919**	958	971	930	920	987
LA 20	20x5	953	978	979	**951**	959	1021
LA 21	20x5	1178	1209	1217	1193	**1169**	1234
LA 22	20x5	**1089**	1129	1131	1094	1105	1193
LA 23	20x5	1158	1162	1154	1130	**1116**	1199
LA 24	20x5	1078	1082	1075	**1068**	1070	1112
LA 25	20x5	**1062**	1129	1134	1077	1106	1181
LA 26	20x5	1399	1422	1429	1396	**1373**	1474
LA 27	20x5	1488	1447	1459	1428	**1407**	1533
LA 28	20x5	**1400**	1466	1487	**1400**	1439	1521
LA 29	20x5	**1370**	1392	1415	1388	1375	1452
LA 30	20x5	1488	1485	1478	1596	**1437**	1545
LA 31	20x5	1856	1878	1904	2000	**1829**	1961
LA 32	20x5	**1924**	2039	2044	2072	1944	2128
LA 33	20x5	1912	1947	1963	1916	**1878**	2001
LA 34	20x5	1866	2020	2033	**1864**	1966	2085
LA 35	20x5	**2022**	2085	2118	2068	2025	2167
LA 36	20x5	**1430**	1511	1519	**1430**	1462	1583
LA 37	20x5	1644	1693	1684	**1624**	1635	1708
LA 38	20x5	**1420**	1479	1482	1475	1421	1544
LA 39	20x5	**1450**	1495	1498	1482	1464	1571
LA 40	20x5	**1434**	1506	1525	**1434**	1466	1576

NSGA-II; solid lines represent the good-set-initialized traditional NSGA-II; dashed lines represent the good-set-initialized EAX-NSGA-II algorithm; and dotted-dashed lines represent the EAX-NSGA-II algorithm with greedy initialization. Figure 6 illustrates the trend of maximum completion time versus iteration count for different algorithms solving the MK08 problem instance. During the initial iterations from 0 to 25 generations, the solid line and dotted line exhibit smaller decreases than the dashed line and dashed-dot line. This demonstrates the advantage of EAX in rapidly eliminating combinations of adjacent edges with severe process conflicts and prolonged machine idle times. Around generation 30, the dashed line’s makespan value

TABLE 4 The energy consumption for machines

Machine	Processing Efficiency	Standby Efficiency
M1	2.0	0.5
M2	1.8	0.3
M3	1.6	0.3
M4	2.4	0.4
M5	2.4	0.4
M6	4.1	0.6
M7	3.5	0.8
M8	4.1	0.9
M9	2.8	0.3
M10	2.7	0.3



(a) Comparison of Makespan (b) Comparison of Total Energy Consumption Results

FIGURE 6 Algorithm Results Comparison Chart

reached approximately 575, the lowest among the four algorithms. This result demonstrates a synergistic effect between the good-set-initialization strategy and EAX's neighbor edge screening mechanism.

Figure 6(b) demonstrates that EAX can precisely identify and retain low-energy edge patterns from parent solutions during crossover. Furthermore, through its edge exploration strategy for AB loop paths, EAX achieves optimization by inheriting effective structures from parent generations. In contrast, traditional algorithms lack this capability, relying solely on incremental genetic adjustments per generation for energy optimization, resulting in a gradually declining curve. During the late stages between generations 300 and 500, the dashed line and dotted line converge to a steady state early on. The horizontal line enters steady-state convergence around generation 300, while the dotted line does not reach convergence until approximately generation 490, resulting in a 63.26% difference in temporal efficiency.

5 | CONCLUSIONS

Numerous studies have collectively established an algorithmic framework for multi-objective FJSP: from fundamental operator refinements to complex scenario extensions, and from single-objective optimization to multi-objective trade-offs, progressively advancing algorithms from theory to practical application. Building upon this foundation, this paper proposes the EAX-GA algorithm for multi-objective scheduling challenges in flexible manufacturing workshops. This approach integrates a neighbor-edge crossover operator with optimal point set initialization. The improved algorithm employs AB-cycle recombination, systematically preserving edge structures during crossover through depth-first search identification of minimal conflict loops, thereby mitigating the disruptive effects observed in traditional genetic algorithms. Dynamic AB-cycle path expansion further optimizes the global-local search balance, enabling structured exploration of the solution space. Experiments on benchmark instances validate that EAX-GA outperforms comparison algorithms in both convergence speed and solution quality.

Future research will focus on deepening the following aspects: 1. The experiments primarily utilized small-to-medium-scale MK and LA series test cases. For large-scale scenarios, the balance between the algorithm's time complexity and solution quality requires further validation. 2. The current model does not account for critical industrial dimensions such as transportation energy consumption, startup/shutdown energy costs, and equipment activation/deactivation energy expenditure. There is room for improvement in the model's practical applicability. Future work will prioritize machine learning ensembles to enhance algorithm adaptability. Reinforcement learning-driven cross-control: Develop dynamic parameter adjustment mechanisms using deep reinforcement learning to autonomously optimize cross-rate and mutation strategies based on real-time evolutionary patterns. Neural-assisted feasibility prediction: Implement graph neural networks (GNNs) during AB-cycle construction to pre-evaluate edge restructuring feasibility, reducing computational time in large-scale scheduling.

AUTHOR CONTRIBUTIONS

The results and experiments of this paper were all accomplished by the first author.

FINANCIAL DISCLOSURE

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

The authors declare no competing financial interests.

CONFLICT OF INTEREST

The authors declare no potential conflict of interests.

REFERENCES

1. Shahrabi J, Adibi MA, Mahootchi M. A reinforcement learning approach to parameter estimation in dynamic job shop scheduling. *Computers & Industrial Engineering.* 2017;110:75–82.
2. Zhou Y, Yang JJ, Zheng LY. Multi-agent based hyper-heuristics for multi-objective flexible job shop scheduling: A case study in an aero-engine blade manufacturing plant. *Ieee Access.* 2019;7:21147–21176.
3. Xiong H, Shi S, Ren D, Hu J. A survey of job shop scheduling problem: The types and models. *Computers & Operations Research.* 2022;142:105731.
4. Chowdhury A, Ghosh A, Sinha S, Das S, Ghosh A. A novel genetic algorithm to solve travelling salesman problem and blocking flow shop scheduling problem. *International Journal of Bio-Inspired Computation.* 2013;5(5):303–314.
5. Jiang M, Yu H, Chen J. Improved self-learning genetic algorithm for solving flexible job shop scheduling. *Mathematics.* 2023;11(22):4700.
6. Gonçalves JF, Magalhães Mendes dJJ, Resende MG. A hybrid genetic algorithm for the job shop scheduling problem. *European journal of operational research.* 2005;167(1):77–95.
7. Choi Y, Hwang HS, Kim CS. A Variable Group Parallel Flexible Job Shop Scheduling in a SMEs Manufacturing Platform. *IEEE Access.* 2023;11:79531–79541.
8. Gu X, Huang M, Liang X. An improved genetic algorithm with adaptive variable neighborhood search for FJSP. *Algorithms.* 2019;12(11):243.
9. Nagata Y, Kobayashi S. A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem. *INFORMS Journal on Computing.* 2013;25(2):346–363.
10. Nikfarjam A, Bossek J, Neumann A, Neumann F. Computing diverse sets of high quality TSP tours by EAX-based evolutionary diversity optimisation. In: 2021:1–11.
11. Wu J, Liu Y, Zhang Y. Multi-objective evolutionary co-learning framework for energy-efficient hybrid flow-shop scheduling problem with human-machine collaboration. *Swarm and Evolutionary Computation.* 2025;95:101932.
12. Ji J, Weng Y, Yang C, Wu T. A multi-resolution grid-based bacterial foraging optimization algorithm for multi-objective optimization problems. *Swarm and Evolutionary Computation.* 2022;72:101098.
13. Zhang X, Chen H. An improved NSGA-III integrating a good point set for multi-objective optimal design of power inductors. *Journal of the Brazilian Society of Mechanical Sciences and Engineering.* 2024;46(10):586.
14. Xiong Q, Dong L, Chen H, Zhu X, Zhao X, Gao X. Enhanced NSGA-II algorithm based on novel hybrid crossover operator to optimise water supply and ecology of Fenhe reservoir operation. *Scientific Reports.* 2024;14(1):31621.
15. Kocaturk A, Orkcu HH, Altunkaynak B. Parameter optimization in biclustering algorithms for large datasets using a combined approach of NSGA-II and TOPSIS. *International Journal of Data Science and Analytics.* 2025:1–18.
16. Dash B, Macedo VDJ, Mohanachandran DK, Pokkuluri KS, Rathinakumar V, Sethi KC. Optimizing time and cost in construction under uncertainty: a fuzzy-driven NSGA-III optimization approach. *Asian Journal of Civil Engineering.* 2025:1–16.
17. He G, Lu Xl. Good point set and double attractors based-QPSO and application in portfolio with transaction fee and financing cost. *Expert Systems with Applications.* 2022;209:118339.
18. Chen P. Effects of the entropy weight on TOPSIS. *Expert Systems with Applications.* 2021;168:114186.
19. Sianaki O. TOPSIS: Technique for order preference by similarity to ideal solution. *Retrieved from Maths Works File Exchange: https://www. mathworks. com/matlabcentral/fileexchange/57143-topsis-technique-for-order-preference-by-similarity-to-ideal-solution.* 2020.
20. Yuan Y, Xu H, Yang J. A hybrid harmony search algorithm for the flexible job shop scheduling problem. *Applied soft computing.* 2013;13(7):3259–3272.

21. Holland JH. Genetic algorithms. *Scientific american*. 1992;267(1):66–73.
22. Amiri MH, Mehrabi Hashjin N, Montazeri M, Mirjalili S, Khodadadi N. Hippopotamus optimization algorithm: a novel nature-inspired optimization algorithm. *Scientific Reports*. 2024;14(1):5032.
23. Li X, Gao L. An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. *International Journal of Production Economics*. 2016;174:93–110.
24. Zhang Y, Yao X, Xu S. Chaos-based improved marine predators algorithm for flexible job-shop scheduling problem. *Journal of Mechanical Science and Technology*. 2024;38(10):5581–5594.
25. Wang H, Xiong H, Zuo W, Shi S. An improved scatter search algorithm for solving job shop scheduling problems with parallel batch processing machine. *Scientific Reports*. 2025;15(1):11872.
26. Zhang C, Li P, Rao Y, Li S. A new hybrid GA/SA algorithm for the job shop scheduling problem. In: Springer. 2005:246–259.
27. Türkyılmaz A, Bulkan S. A hybrid algorithm for total tardiness minimisation in flexible job shop: genetic algorithm with parallel VNS execution. *International Journal of Production Research*. 2015;53(6):1832–1848.
28. Chang C, Han M. Production scheduling optimization of prefabricated building components based on DDE algorithm. *Mathematical Problems in Engineering*. 2021;2021(1):6672753.