applied sciences

MDPI

Article

Multi-Objective Optimization of Energy-Efficient Multi-Stage, Multi-Level Assembly Job Shop Scheduling

Yingqian Dong [1,†], Weizhi Liao [2,*,†] and Guodong Xu [2]

[1] School of Mechanical and Electrical Engineering, Kunming University of Science and Technology, Kunming 650500, China; dyq@stu.kust.edu.cn
[2] School of Mechanical and Electrical Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China; 202222040611@std.uestc.edu.cn
* Correspondence: lwz@uestc.edu.cn
† These authors contributed equally to this work and should be considered co-first authors.

Abstract: The multi-stage, multi-level assembly job shop scheduling problem (MsMlAJSP) is commonly encountered in the manufacturing of complex customized products. Ensuring production efficiency while effectively improving energy utilization is a key focus in the industry. For the energy-efficient MsMlAJSP (EEMsMlAJSP), an improved imperialist competitive algorithm based on Q-learning (IICA-QL) is proposed to minimize the maximum completion time and total energy consumption. In IICA-QL, a decoding strategy with energy-efficient triggers based on problem characteristics is designed to ensure solution quality while effectively enhancing search efficiency. Additionally, an assimilation operation with operator parameter self-adaptation based on Q-learning is devised to overcome the challenge of balancing exploration and exploitation with fixed parameters; thus, the convergence and diversity of the algorithmic search are enhanced. Finally, the effectiveness of the energy-efficient strategy decoding trigger mechanism and the operator parameter self-adaptation operation based on Q-learning is demonstrated through experimental results, and the effectiveness of IICA-QL for solving the EEMsMlAJSP is verified by comparing it with other algorithms.

Keywords: multi-stage and multi-level; assembly job shop; energy-efficient scheduling; decoding design; Q-learning

check for updates

Citation: Dong, Y.; Liao, W.; Xu, G. Multi-Objective Optimization of Energy-Efficient Multi-Stage, Multi-Level Assembly Job Shop Scheduling. *Appl. Sci.* **2024**, *14*, 8712. https://doi.org/10.3390/app14198712

Academic Editor: Arkadiusz Gola

Received: 22 August 2024
Revised: 19 September 2024
Accepted: 23 September 2024
Published: 26 September 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Since the manufacturing industry is one of the most energy-intensive sectors, the question of how to effectively improve energy efficiency while maintaining production efficiency is currently a major focus within the industry [1]. Energy-efficient scheduling (EES) utilizes limited resources to rationally arrange production and optimize the production process, which is an effective way to effectively reduce the energy consumption in manufacturing systems [2]. Recently, EES has been successfully applied to various scheduling problems in discrete manufacturing industries, yielding positive results.

As an extension of the classic job shop scheduling problem (JSP), the multi-stage, multi-level assembly job shop scheduling problem (MsMlAJSP) not only involves multiple stages of production from processing to assembly but also considers complex assemblies composed of multiple jobs with a tree-like multi-level structure consisting of components and subassemblies [3]. It plays an important role in quickly responding to the manufacturing of customized products with multi-variety and small batches.

However, as the complexity of the problem increases, leading to more product structural levels and manufacturing stages, constraints become more complex. This results in idle time between and within stages, significantly increasing idle energy consumption [4]. Therefore, further research is needed to explore the energy-efficient multi-stage, multi-level assembly job shop scheduling problem (EEMsMlAJSP) for complex multi-level products.

The EEMsMlAJSP is highly complex with a vast solution space, presenting challenges for efficient resolution. Currently, solutions to such complex energy-efficient scheduling problems typically focus on two main areas: energy-efficient strategies and algorithm design. Energy-efficient strategies primarily include three key approaches: turning off/on [5], block-moving strategy [6], and machine speed adjustment [7]. Among these, the block-moving energy-efficient strategy is one of the most widely used techniques [8]. It allows for a reduction in energy consumption by adjusting the processing sequence of operations without compromising other objectives. However, while this strategy has been extensively applied, an area that has not been fully explored is how to enhance its computational efficiency while maintaining solution quality. In terms of algorithm design, the imperialist competitive algorithm (ICA) has been successfully applied to various energy-efficient scheduling problems due to its strong global and local search capabilities [9,10]. However, further research is needed to explore and design mechanisms that can effectively balance the exploration and exploitation capabilities of the ICA when solving large-scale, complex scheduling problems like the EEMsMlAJSP.

To address these issues, we propose an improved ICA based on Q-learning (IICA-QL) to solve the EEMsMlAJSP, aiming to minimize the maximum completion time and total energy consumption. The main contributions of this paper are as follows:

(1) A mathematical model for the EEMsMlAJSP with the optimization objectives of minimizing the maximum completion time and total energy consumption is established.
(2) An energy-efficient strategy decoding trigger mechanism is designed to simultaneously enhance solution quality and search efficiency.
(3) A dynamic adaptive adjustment strategy for the assimilation operator parameter is implemented based on Q-learning to enhance convergence speed while maintaining population diversity; thus, the exploration and exploitation capabilities of the algorithm are better balanced.

The specific structure of this paper is as follows: The literature review and related work are shown in Section 2. The problem definition and mathematical model are established in Section 3. The encoding and decoding design is presented in Section 4. The introduction of the improved imperialist competitive algorithm based on Q-learning (IICA-QL) is described in Section 5. The experimental results and analysis are provided in Section 6. The conclusion to this paper is given in Section 7.

2. Literature Review and Related Work

2.1. Multi-Stage, Multi-Level Assembly Job Shop Scheduling Problem

In recent years, with the increase in customized product orders, the production model of multi-variety and small-batch manufacturing has gradually become mainstream [11]. Products under this model typically have complex multi-level structures, which significantly increases the complexity of scheduling. Additionally, processed jobs need to be transported in batches to the assembly stage, and the transportation time cannot be overlooked [12]. As a result, the MsMlAJSP has become a key focus in recent research.

Wang et al. [13] addressed the distributed MsMlAJSP considering maintenance, aiming to minimize the maximum tardiness. Cheng et al. [14] proposed batch transfer and mixed-model assembly for the MsMlAJSP with the optimization objectives of minimizing manufacturing and transportation costs. Building on this, Cheng et al. [15] further considered lot streaming with the objective of minimizing total completion time and production cost. All of the above studies already conducted relevant work on the MsMlAJSP, primarily focusing on improving production efficiency and economic benefits. However, with the increasing awareness of energy consumption and sustainability in modern manufacturing systems [16], there is a growing need to address energy efficiency in the EEMsMlAJSP, where the optimization objectives are not only centered on production efficiency but also on minimizing energy consumption throughout the entire scheduling process.

In the EEMsMlAJSP, energy consumption becomes a critical factor due to the involvement of multiple production stages and levels, where idle times, machine setups, and

transportation processes can lead to significant energy wastage. Therefore, optimizing both production schedules and energy usage concurrently presents a complex, multi-objective problem that requires advanced strategies and algorithms to balance production efficiency and energy conservation.

2.2. Energy-Efficient Strategies in Scheduling Problem

Currently, solving the energy-efficient scheduling problem in complex manufacturing systems mainly focuses on energy-efficient strategies, aiming to reduce idle energy consumption without deteriorating other objectives [17]. Among these strategies, the block-moving energy-efficient strategy, which makes operations within stages more compact by moving process operations, can effectively reduce idle energy consumption. This is currently an effective method for handling multi-operation and complex multi-stage scheduling problems. Gong et al. [18] addressed the energy-efficient flexible job shop scheduling problem and proposed a two-stage Memetic algorithm (TMA) to solve the problem. In the TMA, a block-moving energy-efficient strategy operator was designed to further optimize total energy consumption and the number of machine restarts without altering the makespan. Wang et al. [19] addressed the energy-aware distributed hybrid flow shop scheduling problem and designed an energy-efficient strategy based on the block-moving strategy. This approach further reduces the energy consumption of the non-dominated solutions without altering the makespan, thereby obtaining superior non-dominated solutions. Meng et al. [20] tackled the multi-objective flexible job shop scheduling problem with controllable processing times by incorporating an energy-efficient strategy based on the block-moving strategy into the individual decoding process, thereby achieving lower individual energy consumption.

From the aforementioned literature, it can be seen that there are two main approaches to using the block-moving energy-efficient strategy: One is to apply the block-moving strategy only to the solutions in the non-dominated set to further reduce energy consumption; however, this operation may cause the loss of some potential solutions, resulting in a local rather than a global optimum. The other approach is to embed the block-moving strategy into the individual decoding process to avoid losing the global optimal solution; however, such a design can lead to the waste of computational resources and severely slow down the search efficiency of the algorithm. Therefore, it is an urgent need to design an efficient block-moving strategy that improves solution quality and search efficiency without losing the global optimum.

2.3. Imperialist Competitive Algorithm

The imperialist competitive algorithm (ICA) is an effective evolutionary algorithm framework [21]. It facilitates population evolution through mechanisms such as assimilation, colonial revolutions, and imperial competition, demonstrating efficient global and local search capabilities. For the past few years, the ICA has been successfully applied to solving various multi-objective energy-efficient scheduling problems.

Lei et al. [22] designed a two-phase meta-heuristic (TPM) algorithm based on the ICA and variable neighborhood search (VNS) for a multi-objective flexible job shop scheduling problem (FJSP) with energy consumption thresholds, aiming to minimize the makespan and total tardiness. Li et al. [23] proposed a two-level imperialist competitive algorithm (TICA) for the energy-efficient hybrid flow shop scheduling problem. In this algorithm, the strongest empire and other empires are divided into two levels, with the strongest empire not participating in the competition. To generate high-quality solutions, different assimilation and revolution operations are performed during various search phases within the empire. Zhou et al. [24] addressed the energy-efficient hybrid flow shop scheduling problem with uncertain processing times by proposing an Empire Grouping Imperialist Competitive Algorithm (EGICA). This algorithm constructs initial empires through the normalized total cost under interval scenarios and groups them into empire groups. By executing assimilation operations on different parts of the encoding and performing adaptive

revolutions, it enhances performance. Additionally, a two-phase imperialist competition operation is used to maintain population diversity.

From the above literature, it is evident that many researchers have made significant improvements to the ICA and successfully applied it to various scheduling problems. For the assimilation in the global search of the algorithm, existing studies mostly adopt the uniform crossover (UX) operator [17], with the probability parameters preset through experiments. However, applying traditional static parameters to the ICA may not ensure optimal search efficiency and solution quality due to the complexity of the EEMsMlAJSP [25]. Therefore, it is necessary to design a reasonable dynamic parameter adaptive adjustment mechanism to balance the exploration and exploitation of the algorithm.

2.4. Evolutionary Algorithms Integrated with Reinforcement Learning

Reinforcement Learning (RL), such as Q-learning, possesses strong decision-making and optimization capabilities. Recent studies have shown that Q-learning can effectively adjust algorithm parameters, enhancing the balance between exploration and exploitation and improving solution quality. Chen et al. [26] applied SARSA and Q-learning to adaptively adjust the crossover and mutation probabilities of a genetic algorithm (GA) during the early and late stages of optimization in flexible job shop scheduling problems. Chen et al. [25] used Q-learning to dynamically adjust key algorithm parameters in multi-objective planning and scheduling for PCB assembly lines. Li et al. [27] applied Q-learning to adaptively adjust neighborhood sizes in solving green flexible job shop scheduling problems with Type-II fuzzy processing times (ET2FJSP), thereby increasing the diversity of Pareto solutions. These studies demonstrate the effectiveness of Q-learning in parameter adjustment, highlighting its potential to improve the performance of algorithms in various scheduling problems. These successful applications of Q-learning in parameter adjustment encourage us to adopt this approach for solving the EEMsMlAJSP.

3. Problem Formulation

3.1. Basic Concepts of Multi-Objective Optimization

In general, a multi-objective optimization problem can be expressed as follows:

$$\min F(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})], \mathbf{x} \in \mathbf{X}$$

where $F(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})]$ represents the k objective functions to be optimized simultaneously, and $\mathbf{x} \in \mathbf{X}$ is the decision variable in the solution space.

For two solutions $\mathbf{x_1}$ and $\mathbf{x_2}$, if and only if $\forall i \in \{1, 2, \ldots, k\}$, $f_i(\mathbf{x_1}) \leq f_i(\mathbf{x_2})$ and $\exists i' \in \{1, 2, \ldots, k\}$, $f_i(\mathbf{x_1}) < f_i(\mathbf{x_2})$, then $\mathbf{x_1}$ is said to dominate $\mathbf{x_2}$ (denoted as $\mathbf{x_1} \prec \mathbf{x_2}$). If no other solution dominates $\mathbf{x_1}$, then $\mathbf{x_1}$ is called a non-dominated solution. The set of all non-dominated solutions is called the Pareto front.

3.2. Problem Definition

The EEMsMlAJSP addressed in this work consists of multiple stages, including processing, transportation, and assembly. The notations defined for use in the EEMsMlAJSP mathematical model are shown in Table 1. The problem is defined as follows. In the processing stage, transportation stage, and assembly stage, there are N_M flexible processing machines $\mathbf{M} = \{M_1, \cdots, M_m, \cdots, M_{N_M}\}$, N_T transportation vehicles $\mathbf{V} = \{V_1, \cdots, V_t, \cdots, V_{N_T}\}$ with the same load capacity G_{max}, and N_K assembly machines $\mathbf{K} = \{K_1, \cdots, K_k, \cdots, K_{N_K}\}$, respectively. There is a product set $\mathbf{P} = \{P_1, \cdots, P_z, \cdots, P_{N_p}\}$ that needs to be manufactured, where each product $P_z \in \mathbf{P}$ is composed of a set of various jobs $\mathbf{J_z}$ and should undergo an assembly operation set $\mathbf{A_z}$ to be assembled into the final product. Each job $J_j \in \mathbf{J_z}$ has a release time and consists of a set of processing operations $\mathbf{O_j}$. All operations $O_j^i \in \mathbf{O_j}$ must be completed before J_j can be transported in batches by vehicles with the load capacity constraint. Once all jobs J_j belonging to a product P_z have been transported to the assembly stage, the assembly of P_z can begin. In the processing stage, each operation

O_j^i has a set of compatible machines $\mathbf{M_j^i}$ from which one machine is selected for processing. Once the operation is completed, the next operation for job J_j is processed according to the specified sequence. In the transportation stage, a fleet of vehicles $\mathbf{V}$ is available to transport jobs from the processing stage to the assembly stage. Each vehicle $V_t \in \mathbf{V}$ departs from the processing shop and must return after completing its transportation tasks. The vehicles are subject to capacity constraints and are allowed to make multiple trips, with each trip carrying a batch of components produced in the processing stage. In the assembly stage, once all components belonging to product P_z are ready, all operations $A_z^a \in \mathbf{A_z}$ are sequentially completed according to the assembly process.

Table 1. Notations of mathematical model.

Notations	Descriptions
Indices	
z	Index of products, $z = 1, 2, \cdots, N_P$
j	Index of jobs, $j = 1, 2, \cdots, N_Q$
i	Index of processing operations, $i = 1, 2, \cdots, N_O$
a	Index of assembly operations, $a = 1, 2, \cdots, N_A$
m	Index of processing machines, $m = 1, 2, \cdots, N_M$
t	Index of transportation vehicles, $t = 1, 2, \cdots, N_T$
w	Index of transportation vehicle load transportation, $w = 1, 2, \cdots, N_{tw}$
k	Index of assembly machines, $k = 1, 2, \cdots, N_K$
Parameters	
$\mathbf{M}$	Set of processing machines, $\mathbf{M} = \{M_1, \cdots, M_m, \cdots, M_{N_M}\}$
$\mathbf{V}$	Set of transportation vehicles, $\mathbf{V} = \{V_1, \cdots, V_t, \cdots, V_{N_T}\}$
$\mathbf{K}$	Set of assembly machine, $\mathbf{K} = \{K_1, \cdots, K_k, \cdots, K_{N_K}\}$
$\mathbf{P}$	Set of products, $\mathbf{P} = \{P_1, \cdots, P_z, \cdots, P_{N_P}\}$
N_p	Number of products
N_Q	Number of jobs
N_O	Number of processing operations
N_A	Number of assembly operations
N_M	Number of processing machines
N_T	Number of transportation vehicles
N_K	Number of assembly machines
N_O^m, N_A^k	Number of operations on processing machine m and assembly machine k
N_{tw}	Number of load transportations by vehicle t
G_{max}	Maximum load capacity of vehicle
$J_j, \mathbf{J_z}$	Job indexed by j, set of all jobs belonging to P_z
$O_j^i, \mathbf{O_j}$	i-th processing operation of J_j, set of all operations of J_j
$A_z^a, \mathbf{A_z}$	a-th assembly operation and set of assembly operations of P_z
$\mathbf{M_j^i}$	Set of machines available for processing operation O_j^i
PP_m, PP_k	Unit power consumption of processing machine m and assembly machine k in working mode
PI_m, PI_k	Unit power consumption of processing machine m and assembly machine k in standby mode
PS_m	Unit power consumption of processing machine m in setup mode
PT_t	Unit power consumption of transportation vehicle t
Dis_{P-A}	Distance between processing shop and assembly shop
V_0, V_t^w	Speed of transportation vehicles when unloaded and speed of transportation vehicle t during w-th load transportation
$P(\pi_i^m), P(\pi_a^k)$	Processing time of operation π_i^m and π_a^k
$SP(\pi_{i-1}^m, \pi_i^m)$	Setup time of operation π_{i-1}^m and π_i^m
$IP(\pi_{i-1}^m, \pi_i^m)$	Idle time between operation π_{i-1}^m and π_i^m excluding setup time
$IP(\pi_{a-1}^k, \pi_a^k)$	Idle time between operation π_{a-1}^m and π_a^m
$RT(\pi_1^m)$	Release time of job π_1^m
Decision variables	
$\boldsymbol{\pi^m} = [\pi_1^m, \pi_2^m, \cdots, \pi_{N_O^m}^m]$	Sequence of operations on processing machine m
$\boldsymbol{\pi^k} = [\pi_1^k, \pi_2^k, \cdots, \pi_{N_A^k}^k]$	Sequence of operations on assembly machine k

Table 1. *Cont.*

Notations	Descriptions
$\boldsymbol{\pi^{st1}}, \boldsymbol{\pi^{st2}}, \boldsymbol{\pi^{st3}}$	Sequence of operations on processing stage, transportation stage, and assembly stage
$ST(\pi_a^k), CT(\pi_a^k)$	Start time and completion time of operation π_a^k
$EC_R^{st1}, EC_S^{st1}, EC_P^{st1}, EC_I^{st1}$	Release energy consumption, setup energy consumption, processing energy consumption, and idle energy consumption in processing stage
EC_A^{st3}, EC_I^{st3}	Assembly energy consumption and idle energy consumption in assembly stage
$EC^{st1}, EC^{st2}, EC^{st3}$	Energy consumption in processing stage, transportation stage, and assembly stage
$C_{\max}$	Maximum completion time of schedule
TEC	Total energy consumption

The optimization objectives of this problem are to simultaneously minimize $C_{\max}$ and TEC. It is a multi-objective problem and encompasses several sub-problems, including the sequencing of processing operations, assignment of processing machines, sequencing of job transportation, assignment of vehicles, sequencing of product assembly, sequencing of assembly operations, and assignment of assembly machines. The computational complexity of the mathematical model can be calculated as $O = ((N_O!) \times {N_M}^{N_O} \times (N_Q!) \times {N_T}^{N_Q} \times (N_P!) \times {N_K}^{N_A})$. So as the problem size increases, the complexity grows exponentially. Moreover, as an extension of the well-known NP-hard problem JSP, the EEMsMlAJSP integrates assembly and batch transportation, making it an NP-hard problem as well and even more complex.

A schematic diagram of the EEMsMlAJSP model is shown in Figure 1. This figure illustrates the workflow of two products (P_1, P_2), each made up of multiple components and jobs. In the processing stage, these jobs are assigned to flexible machines (M_1, M_2, M_3) for operations. Once processed, jobs are batch transported by three vehicles (V_1, V_2, V_3) to the assembly stage. In this final stage, each product's components are assembled on different machines (K_1, K_2) to form the final product. Some assumptions are listed below:

- In the processing and assembly stages, any machine can only process one operation at a time.
- Once started, processing operations, transportation processes, and assembly operations cannot be interrupted.
- All machines and vehicles are available at time zero.
- The transportation time between machines for consecutive processing (or assembly) operations is not considered.
- Vehicles maintain a constant speed during transportation.
- The initial setup time is included in the release time of the workpiece.

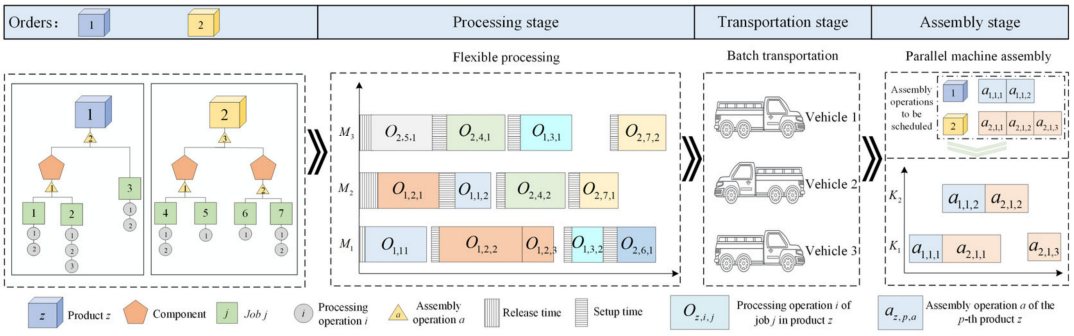


Figure 1. A schematic diagram of the EEMsMlAJSP.

3.3. Mathematical Model

The mathematical model for the three stages with two optimization objectives in the EEMsMlAJSP is constructed as follows:

The first objective, the maximum completion time, is the time when all products are fully assembled, calculated as follows:

$$C_{\max} = \max\left\{CT(\pi_a^k)\right\}, \forall a, k \tag{1}$$

The second objective, namely total energy consumption, is composed of three parts: total energy consumption in the processing stage, total energy consumption in the transportation stage, and total energy consumption in the assembly stage. The total energy consumption for each stage is calculated as follows:

(1) Processing stage: The total energy consumption in this stage is composed of the energy consumption of the processing machines, which is determined by four modes of machines: release mode, setup mode, working mode, and idle mode. The energy consumption for each mode of the processing machines is constructed as follows:

Release energy consumption: This refers to the energy consumption of machines before the first arrival time of the job. The release energy consumption in the processing stage is calculated as follows:

$$EC_R^{st1} = \sum_{m=1}^{N_M} PP_m \times RT(\pi_1^m) \tag{2}$$

Setup energy consumption: This energy consumption is generated during the preparation process of the machine before entering the working mode. The setup energy consumption in the processing stage is calculated as follows:

$$EC_S^{st1} = \sum_{m=1}^{N_M} \sum_{i=2}^{N_{Om}} PS_m \times SP(\pi_{i-1}^m, \pi_i^m) \tag{3}$$

Working energy consumption: This energy consumption is generated when the processing machine is in working mode. The working energy consumption in the processing stage is calculated as follows:

$$EC_P^{st1} = \sum_{m=1}^{N_M} \sum_{i=1}^{N_{Om}} PP_m \times P(\pi_i^m) \tag{4}$$

Idle energy consumption: This energy consumption is generated during the time between processing two consecutive operations, excluding setup time. The idle energy consumption in the processing stage is calculated as follows:

$$EC_I^{st1} = \sum_{m=1}^{N_M} \sum_{i=2}^{N_{Om}} PI_m \times IP(\pi_{i-1}^m, \pi_i^m) \tag{5}$$

The total energy consumption in the processing stage is calculated as follows:

$$EC^{st1} = EC_R^{st1} + EC_S^{st1} + EC_P^{st1} + EC_I^{st1} \tag{6}$$

(2) Transportation stage: The total energy consumption in this stage consists of the energy consumption of the transportation vehicles, which is composed of two parts: loaded energy consumption and unloaded energy consumption. The total energy consumption in the transportation stage is calculated as follows:

$$EC^{st2} = \sum_{t=1}^{N_T} \sum_{w=1}^{N_{tw}} PT_t \times \left(\frac{Dis_{P-A}}{V_0} + \frac{Dis_{P-A}}{V_t^w}\right) \tag{7}$$

(3) Assembly stage: The total energy consumption in this stage consists of the energy consumption of the assembly machines, which is composed of two modes of modes:

working mode and idle mode. The energy consumption for each mode of the assembly machines is constructed as follows:

Working energy consumption: This energy consumption is generated when the assembly machine is in working mode. The working energy consumption in the assembly stage is calculated as follows:

$$EC_A^{st3} = \sum_{k=1}^{N_K} \sum_{a=1}^{N_{Ak}} PP_k \times P(\pi_a^k) \tag{8}$$

Idle energy consumption: This refers to the energy consumption generated between two consecutive operations on the same assembly machine. The idle energy consumption of the machine in the assembly stage is calculated as follows:

$$EC_I^{st3} = \sum_{k=1}^{N_K} \sum_{a=2}^{N_{Ak}} PI_k \times IP(\pi_{a-1}^k, \pi_a^k) \tag{9}$$

The total energy consumption in the assembly stage is calculated as follows:

$$EC^{st3} = EC_A^{st3} + EC_I^{st3} \tag{10}$$

The total energy consumption of the three stages is calculated as follows:

$$TEC(\boldsymbol{\pi}) = EC_{st1} + EC_{st2} + EC_{st3} \tag{11}$$

The optimization objective is to find an optimal sequence in the set of all three-stage sequences $\prod(\boldsymbol{\pi} = [\boldsymbol{\pi}^{\mathbf{st1}}, \boldsymbol{\pi}^{\mathbf{st2}}, \boldsymbol{\pi}^{\mathbf{st3}}] \in \prod)$ that simultaneously minimizes the maximum completion time and total energy consumption. The specific details are as follows:

$$C_{\max}(\boldsymbol{\pi}) = \max\left\{CT(\pi_a^k)\right\}, \forall a, k \tag{12}$$

$$TEC(\boldsymbol{\pi}) = EC_{st1} + EC_{st2} + EC_{st3} \tag{13}$$

$$\begin{cases} C_{\max}(\boldsymbol{\pi}^*) = \min\{C_{\max}(\boldsymbol{\pi}^{\mathbf{st1}}, \boldsymbol{\pi}^{\mathbf{st2}}, \boldsymbol{\pi}^{\mathbf{st3}})\}, TEC(\boldsymbol{\pi}^*) = \min\{TEC(\boldsymbol{\pi}^{\mathbf{st1}}, \boldsymbol{\pi}^{\mathbf{st2}}, \boldsymbol{\pi}^{\mathbf{st3}})\} \\ \boldsymbol{\pi}^* = \arg\left\{ \begin{array}{c} C_{\max}(\boldsymbol{\pi}^{\mathbf{st1}}, \boldsymbol{\pi}^{\mathbf{st2}}, \boldsymbol{\pi}^{\mathbf{st3}}) \\ TEC(\boldsymbol{\pi}^{\mathbf{st1}}, \boldsymbol{\pi}^{\mathbf{st2}}, \boldsymbol{\pi}^{\mathbf{st3}}) \end{array} \right\} \rightarrow \min, \forall(\boldsymbol{\pi}^{\mathbf{st1}}, \boldsymbol{\pi}^{\mathbf{st2}}, \boldsymbol{\pi}^{\mathbf{st3}}) \in \prod \end{cases} \tag{14}$$

4. Encoding and Decoding Design

As described in Section 2.1, the problem characteristics of the EEMsMlAJSP result in an exceptionally large solution space. If full encoding is applied to each stage, although the entire solution space can be effectively represented, it also results in the algorithm convergence being slowed and invalid solutions being generated during the algorithm iteration process needing to be repaired, thus affecting the search efficiency of the algorithm. Heuristic rules can effectively trim the solution space while maximizing the discovery of near-optimal solutions to the problem; thereby, the convergence speed of the algorithm is improved. Therefore, based on the coupling properties between the stages of the EEMsMlAJSP [28,29], only the processing operations in the processing stage are encoded, with the assignment of processing machines and the sub-problems in subsequent stages being decoded using heuristic rules.

4.1. Encoding

In this study, the solution to the problem is represented using an operation-based encoding method. The operations of a job are indicated by the number of the job, and the number of times that the job number appears represents which operation it is for that job. Due to the design order requirements of the products considered in this paper, and the possibility of identical jobs existing among different products, product information needs to

be read sequentially to determine the required jobs for the product, and each job is assigned a unique code for identification. The pseudocode for the encoding is shown in Algorithm 1.

To better illustrate the encoding process, we use an example where the demand for product 1 is 2, and the demand for product 2 is 1. Product 1 consists of components 1 and 2, while product 2 consists of components 3 and 4. Component 1 is composed of job 1; component 2 is composed of jobs 2 and 3; component 3 is composed of jobs 2 and 4; and component 4 is composed of jobs 5, 6, and 7. Each job is processed through corresponding operations. For example, job 1 includes operation 1, operation 2, and operation 3. The bottom encoding sequence represents the order of operations for each job, such as [1 1 1 2 3 1 1 1 2 3 2 4 4 5 6 6 6 7], where [1 1 1] denotes the three operations of job 1. The specific encoding method for this example is shown in Figure 2.

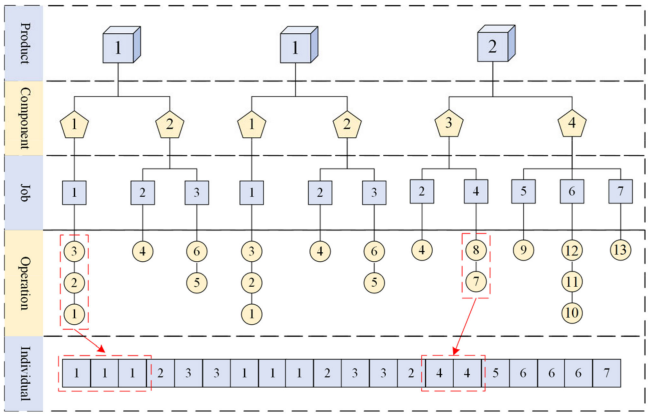


Figure 2. Schematic diagram of encoding.

Algorithm 1. Encoding Method

Input: order information; empty set $chrom$
Output: individual encoding $chrom$
Read order information, including product types N_P, number of jobs required for each product N_{Pj}, product composition P_part, and product demand P_need.

```
For i = 1 to N_P do
    For j = 1 to N_Pj do
        job ← P_i_part_j
        For k = 1 to P_j_need do
            chrom ← chrom ∪ job
        End For
    End For
End For
```

4.2. Decoding Rules

To narrow the solution space while maximizing the discovery of near-optimal solutions to the problem, the heuristic decoding rules for allocation in the processing stage and subsequent transportation and assembly stages are as follows:

(1) Machine assignment in the processing stage

The earliest completion time (ECT) rule is used to assign operations to the machine that can complete them the earliest. If multiple machines can complete the operation at the same time, the minimum energy consumption first (MECF) rule is applied to select the machine with the lowest energy consumption.

(2) Job transportation sequence and vehicle allocation in the transportation stage

Under the constraint of vehicle load capacity, the first completed first transported (FCFT) rule is used to organize the batch transportation of jobs. This means that jobs are loaded into available vehicles in the order of their completion times, ensuring that no single vehicle is overloaded. Subsequent jobs are transported by following vehicles according to the same rule.

(3) Product assembly sequence and machine assignment in the assembly stage

When the jobs transported to the assembly shop are sufficient for the complete assembly of the product, the assembly is carried out according to the product assembly process. If multiple products simultaneously meet the assembly conditions, the minimum jobs first (MJF) rule is used to determine the assembly sequence. During product assembly, the machine assignment for assembly operations follows the same rules as in the processing stage to determine the assembly machine for each operation.

4.3. Design of Energy-Efficient Strategy Decoding Trigger Mechanism

To address the issue of substantial idle energy consumption caused by the hierarchical coupling characteristics of the EEMsMlAJSP, an energy-efficient strategy based on block-moving can reduce machine idle time by making dispersed operations more compact without changing the maximum completion time, thus reducing the idle energy consumption generated by machines.

However, as the problem scale increases, executing the energy-efficient strategy usually takes a long time and consumes significant computational resources. In fact, not all solutions benefit from this strategy in terms of improving the quality of non-dominated solutions. Therefore, we propose an adaptive trigger mechanism for the energy-efficient strategy decoding, which ensures potential high-quality solutions are not lost while improving the search efficiency. The specific operations are as follows:

Step 1: Apply the decoding rules and the active decoding method [30] to the individual to obtain the scheduling results $C_{\max}(\boldsymbol{\pi})$ and $TEC(\boldsymbol{\pi})$, and record the idle time of all machines.

Step 2: According to Equations (15) and (16), obtain the upper bound of energy savings ES_{UB} and the lower bound of energy consumption E_{LB}, respectively, for the scheduling scheme after applying the energy-efficient strategy.

Step 3: Determine the relationship between $(C_{\max}(\boldsymbol{\pi}), E_{LB})$ and the current non-dominated solutions. If they are not dominated by the current non-dominated solutions or if they are mutually non-dominating, proceed to Step 5; otherwise, proceed to Step 4.

Step 4: Use the objective values $C_{\max}(\boldsymbol{\pi})$ and $TEC(\boldsymbol{\pi})$ obtained from the decoding rules as the decoding results of the scheduling scheme.

Step 5: Following the product assembly sequence of the scheduling scheme, check each product in reverse order from the last to the first according to the process sequence, and determine whether the block-moving condition is met according to Equation (17). If the condition is met, update the completion time of the operation according to Equation (18). Otherwise, check the preceding operation of this operation.

Step 6: If all operations have been checked, use Equation (19) to determine whether the idle intervals meet the shutdown conditions for profit and loss time, and output the results $C_{\max}(\boldsymbol{\pi})$ and $TEC(\boldsymbol{\pi})$ of the scheduling scheme after executing the energy-efficient strategy. Then, update the non-dominated solution set. Otherwise, return to Step 5.

$$ES_{UB} = \sum_{a=2}^{N_A^k} IP(\pi_{a-1}^k, \pi_a^k) \times PI_k \tag{15}$$

$$E_{LB} = TEC(\boldsymbol{\pi}) - ES_{UB} \tag{16}$$

$$P(\pi_a^k) \leq IP(\pi_l^k, \pi_{l+1}^k) \tag{17}$$

$$CT(\pi_a^k) = \begin{cases} \max(CTI_i^k),\ \forall i,\ \pi_a^k \text{ is the final assembly operation of product} \\ \min\left\{C(\pi_{lasta}^k), \max(CTI_i^k)\right\},\ \text{otherwise} \end{cases} \tag{18}$$

$$T_{off-on} \leq EC_{off-on} / \overline{EC} \tag{19}$$

where $IP(\pi_l^k, \pi_{l+1}^k)$ is the idle time after the operation π_a^k on machine m, $l = a, a+1, \cdots, N_A^k - 1$. CTI_i^k denotes the end time of the insertable idle interval for π_a^k on machine k, and π_{lasta}^k denotes the immediate subsequent operation of π_a^k. T_{off-on} is the time required for the machine to turn off and on, EC_{off-on} is the energy consumption for the machine to turn off and on, and $\overline{EC}$ is the energy consumption per unit time of the machine. The flow of the energy-efficient strategy decoding trigger mechanism is shown in Figure 3.

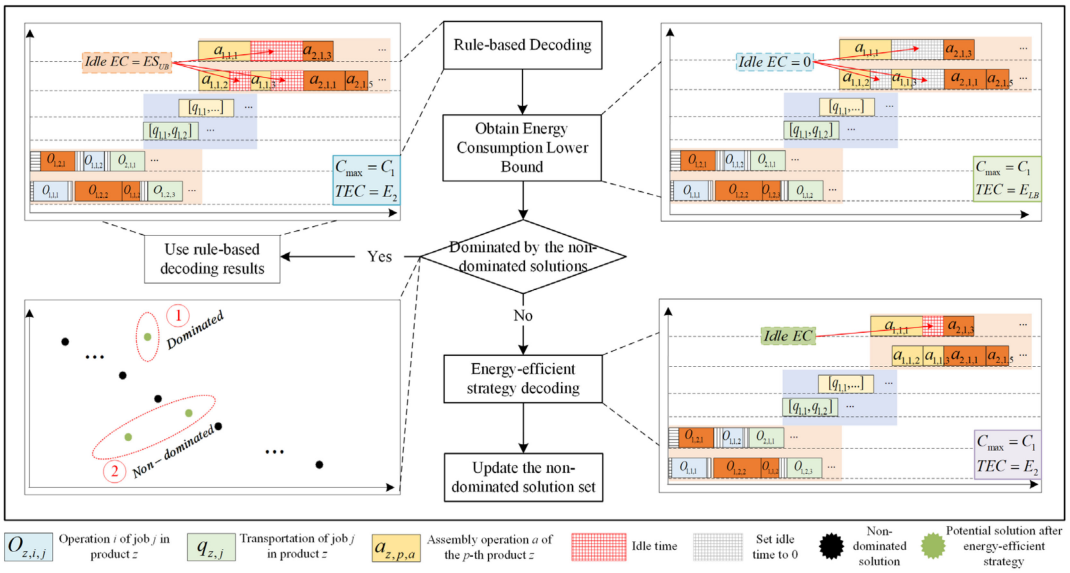


Figure 3. Illustration of trigger mechanism for decoding energy-efficient strategy.

5. IICA-QL for EEMsMlAJSP

5.1. Initialization of Empires

In IICA-QL, a country $\boldsymbol{\pi^i}$ denotes a solution to the problem, where $i = 1, 2, \cdots, Ps$, and Ps denotes the population size. The population $\mathbf{Pop(g)}$ in generation g is composed of all countries in generation g, denoted as $\mathbf{Pop(g)} = \{\boldsymbol{\pi^{g,1}}, \boldsymbol{\pi^{g,2}}, \ldots, \boldsymbol{\pi^{g,Ps}}\}$. To obtain a high-quality initial population while maintaining population diversity, two heuristic methods are used to initialize the population. $\lambda \times Ps$ countries, with λ set to 0.2, are generated using product aggregation (PA) and job aggregation (JA) rules, while the remaining countries are generated randomly. This approach ensures a high starting point for the search while maintaining search breadth.

Subsequently, according to Equation (20) [21], the cost c_i^1 of each country in the initial population is calculated in turn, $i = 1, 2, \cdots, Ps$. After sorting c_i^1 in non-ascending order, the countries with the highest costs are selected as N_{im} imperialists for N_{im} empires, while the other countries in the population are designated as colonies. The number of colonies N_{col} is then calculated according to Equation (21). Next, the normalized power of each imperialist $Imp_k (k = 1, 2, \cdots, N_{im})$ is calculated according to Equation (22), and the number of colonies assigned to each colonial country NC_k is determined based on their normalized power. Finally, the corresponding number of colonies is randomly selected and assigned to the corresponding empires, and the selected colonies are removed from the original colony population.

$$c_i^g = \max_{l=1,2,\cdots,Ps} \{rank_l\} - rank_i + dist_i / \sum_{j \in \theta_{rank_i}} dist_j \tag{20}$$

$$N_{col} = PS - N_{im} \tag{21}$$

$$Pow_k = c_k^g / \sum_{k=1}^{N_{im}} c_k^g \tag{22}$$

$$NC_k = round(Pow_k \times N_{col}) \tag{23}$$

Before the iteration begins, an empty external archive set Ω is established to store the non-dominated set of the initial population, and it is dynamically updated during the iteration process.

5.2. Assimilation Based on Q-Learning Adaptive Adjustment of Operator Parameter

Assimilation is the process by which colonies within an empire are moved towards the imperialist, with population evolution achieved through the assimilation of colonies by the imperialist. To ensure convergence speed while maintaining population diversity, a dynamic adaptive adjustment strategy for the operator parameter based on Q-learning is designed. First, two different operator co-evolution operations are employed. Then, a Q-learning-based adaptive adjustment strategy for the UX operator parameter is implemented to enhance the convergence and diversity of the algorithm. Finally, an elite retention strategy is used to update the individuals of each empire, thereby improving the convergence speed. The pseudocode is shown in Algorithm 2.

Algorithm 2. Assimilation based on Q-learning adaptive adjustment of operator parameter

Input: empires *Emp*; external archive set Ω; number of imperialists N_{im}; number of colonies NC; assimilation probability P_a; probability of assimilation direction P_d

Output: updated *Emp* and Ω

For $k = 1$ to N_{im} **do**

  $Emp_k_offspring \leftarrow \varnothing$, $Emp_k_subPop \leftarrow \varnothing$;

  For $i = 1$ to NC_k **do**

    If $rand < P_a$ **then**

      If $rand < 0.4$ **the**

        $Offspring \leftarrow$ *Perform UX between colony and imperialist*

      Else

        $Offspring \leftarrow$ *Perform UX between colony and solution in* Ω

      End If

      $Emp_k_offspring \leftarrow Emp_k_offspring \cup Offspring$

    End If

  End For

  $Emp_k_subPop \leftarrow Emp_k_colonies \cup Emp_k_offspring$

  $Emp_k_colonies \leftarrow$ Perform non-dominated sorting on Emp_k_subPop and update with the top NC_k countries

  Imperialist ← Perform TPX between the *imperialist* and the non-dominated solutions in Ω, and update using a greedy strategy

End For

Update the external archive set Ω

5.2.1. Adaptive Adjustment of UX Operator Parameter Based on Q-Learning

During the assimilation process, the uniform crossover (UX) operator [17] and the two-point Crossover (TPX) operator [31] are used, respectively applied to the movement of colonies towards the imperialist and the external archive set. Among these, the UX operator is used as the main evolutionary operator to guide the evolutionary process of population and can also be applied for local search [17], playing an important role in the quality of population evolution. The parameter of the UX operator determines the number

of high-quality genes passed from parents to offspring. In the early stages of evolution, a larger parameter is helpful for the algorithm convergence, while in the later stages of evolution, a larger parameter can result in poor population diversity, causing the algorithm to fall into local optima. Therefore, a Q-learning-based adaptive adjustment strategy for the UX operator parameter is devised to dynamically adjust the evolutionary process of the population, enabling the algorithm to converge more efficiently to the Pareto front.

State: The state of the entire population is determined by evaluating the convergence metric ΔGD of the non-dominated solution set compared to the reference set and the diversity metric $\Delta Ratio$, which represents the coverage ratio of the solution space. Here, ΔGD represents the generational distance difference in the non-dominated solutions obtained compared to the Pareto front PF^* between generation t and $t+1$. PF^* is a preset reference point, and $\Delta Ratio$ represents the proportion of subregions occupied by the non-dominated solutions in the current generation in the objective space. The objective space is divided into several non-overlapping regions by weight vectors [32]. ΔGD indicates the degree of convergence of generation $t+1$ compared to generation t; the smaller the value of ΔGD, the better the convergence. $\Delta Ratio$ represents the change in non-dominated set diversity, with $\Delta Ratio < 0$ indicating a decrease in diversity and $\Delta Ratio > 0$ indicating an increase in diversity. The calculations for ΔGD and $\Delta Ratio$ are given by Equations (24) and (25), respectively, where $|PF_t|$ and $|\psi_t|$ represent the number of non-dominated solutions in generation t and the proportion of the objective space they occupy. During the evolution, there are four possible states: (1) $\Delta GD >= 0,\ \Delta Ratio > 0$; (2) $\Delta GD >= 0,\ \Delta Ratio <= 0$; (3) $\Delta GD < 0,\ \Delta Ratio > 0$; and (4) $\Delta GD < 0,\ \Delta Ratio <= 0$.

$$\Delta GD = \left(\sqrt{\sum_{x \in PF_{t+1}} \min_{y \in PF^*} d(x,y)^2}/|PF_{t+1}|\right) - \left(\sqrt{\sum_{x \in PF_t} \min_{y \in PF^*} d(x,y)^2}/|PF_t|\right) \tag{24}$$

$$\Delta Ratio = |\psi_{t+1}| - |\psi_t| \tag{25}$$

Action: The action set $\mathbb{A}$ is composed of ten actions, representing ten levels of the UX operator parameter. By determining the current state of the population, the action with the highest Q-value in the Q-table is selected based on the ε-greedy strategy.

ε-greedy: In the initial iterations, Q-values usually lack prior knowledge, making it easy for the agent to fall into local optima during learning. Therefore, an ε-greedy strategy based on a sigmoid function is used to guide the learning of the agent. The mathematical expression for ε-greedy is as follows:

$$\varepsilon = 0.1 + (\varepsilon - 0.1) \times \frac{1}{1 + e^{(-0.2 \times (g-50))}} \tag{26}$$

where g represents the number of iterations.

Reward: By calculating the values of ΔGD and $\Delta Ratio$ after executing an action, the agent receives a corresponding reward and updates the Q-value of the action in the Q-table for the corresponding state according to Equation (27). If $\Delta GD \geq 0, \Delta Ratio > 0$, then reward = 1; if $\Delta GD \geq 0, \Delta Ratio <= 0$, then reward = -2; if $\Delta GD < 0, \Delta Ratio > 0$, then reward = 4; if $\Delta GD < 0, \Delta Ratio \leq 0$, then reward = 2.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[reward_{t+1} + \gamma \max_{a \in \mathbb{A}} Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{27}$$

where α and γ are the learning rate and discount factor, respectively, and $\mathbb{A}$ is the set of available actions.

Figure 4 provides an example. In Figure 4, the evolutionary process of the population is illustrated through the interaction between states and actions within the Q-learning framework. Initially, in generation t, the population is in state S_3, where action A_2 is selected, and after executing this action, a reward of 4 is received based on the performance indicators. The population then transitions to state S_1 in generation $t+1$, where action A_{10} is chosen. However, the performance does not improve significantly, resulting in a negative reward of -2, and the

population moves to state S_4 in generation $t + 2$. In this iterative process, rewards guide the selection of actions to achieve better solutions across generations, dynamically adjusting through Q-values (as shown in the table on the right), where specific actions in certain states (e.g., S_3 with A_2) yield high Q-values like 0.4, indicating the optimal choices learned over time.

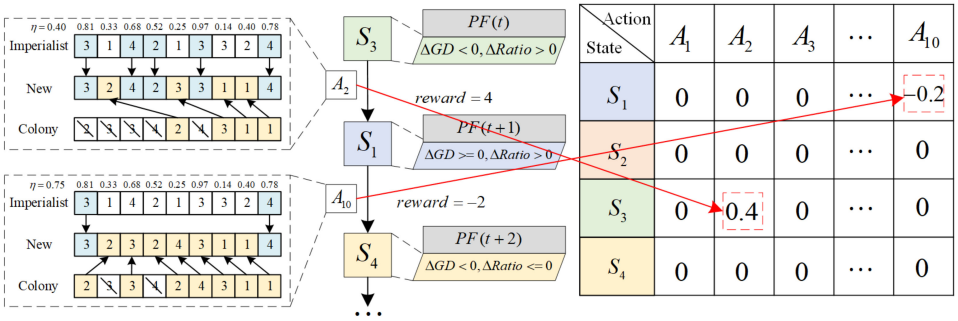


Figure 4. Q-table update diagram.

5.2.2. Update Strategy

To achieve the rapid convergence of the population, an elite retention strategy is employed to select colonies that need to be retained after the assimilation process. The specific operations are as follows: Individuals generated during the assimilation process of imperialist Imp_k are placed into the colony offspring population $Offspring_k$. After the assimilation operation within the empires is completed, the original colony population is merged with the colony offspring population to form a new population $SubPop_k$. Then, the individuals in $SubPop_k$ are non-dominated sorted, and their crowding distances are calculated. A number of NC_k colonies are selected as the next generation of colonies. Additionally, for new solutions generated by the assimilation of the imperialist, if they are not dominated by the old solutions, the original imperialist is replaced by the new solutions; otherwise, no replacement is made.

5.3. Revolution Based on Hyper-Heuristic Variable Neighborhood Search

Colonial revolution is a process of further exploring and exploiting the neighborhood of superior colonies after completing the assimilation operation. In this paper, various neighborhood search (VNS) operations are designed, and high-level information constructed from a two-dimensional probability matrix is used to guide the selection of corresponding low-level variable neighborhood search operation sets. The selection of variable neighborhood search operators is dynamically adjusted by the high-level strategy based on the feedback information received from the environment after each application of the low-level variable neighborhood search operation set, thereby improving the search performance of the algorithm. The pseudocode is shown in Algorithm 3.

Algorithm 3. Revolution based on hyper-heuristic variable neighborhood search

Input: empires *Emp*; number of imperialists N_{im}; assimilation probability P_r; probability matrix *LSM*
Output: updated *Emp* and *LSM*
For $k = 1$ to N_{im} **do**
  $N_{kr} \leftarrow NC_k \times P_r$
  Select high-quality colonies based on national cost to perform revolution
  For $i = 1$ to N_{kr} **do**
    NS ← Sample *LSM* to generate variable neighborhood operation set
    Colonies perform VNS based on the neighborhood operation set and update using a greedy strategy
    Update the *LSM* based on feedback
  End For
  If the colony cost is greater than the imperialist cost, replace the imperialist
End For

5.3.1. Neighborhood Search (NS) Operations

To implement an effective perturbation strategy for individuals, two single-operation-based neighborhood operations and five block-based neighborhood operations are designed as neighborhood search operators. Figure 5 provides an illustrative example.

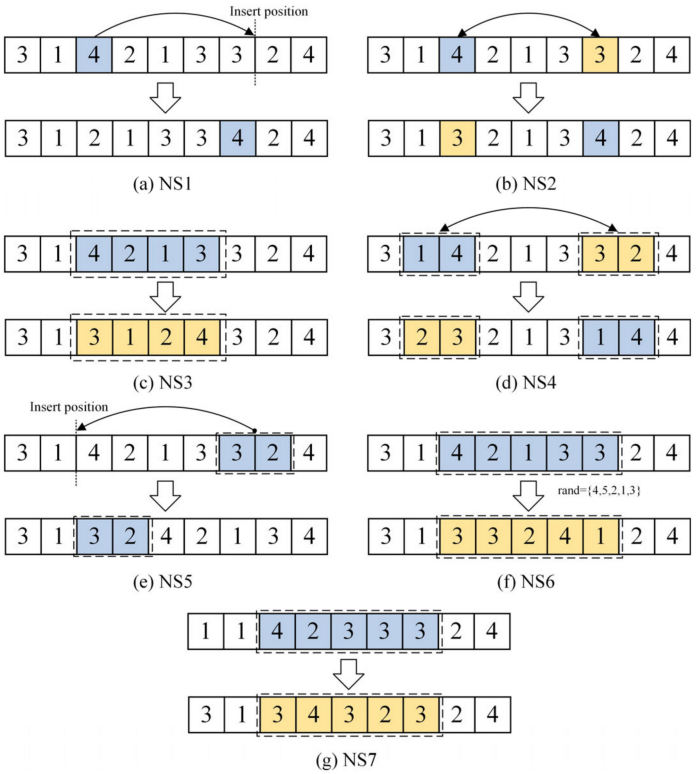


Figure 5. Schematic diagram of neighborhood structure.

NS1: Single-operation insertion operation for individual encoding. A random operation is selected from the individual encoding and inserted into another position.

NS2: Single-operation exchange operation for individual encoding. Two different operations are randomly selected, and their positions are swapped.

NS3: Reversal operation for individual encoding. Two positions are randomly selected, and the segment of operations between these positions is reversed.

NS4: Segment exchange operation for individual encoding. Two segments of operations are randomly selected, and their positions are exchanged.

NS5: Segment insertion operation for individual encoding. Two segments of operations are randomly selected, and one segment is inserted into another position within the encoding.

NS6: Segment shuffling operation for individual encoding. A segment of operations is randomly selected, and operations within segment are randomly shuffled.

NS7: Segment gene dispersal operation for individual encoding. The individual encoding is divided into several segments, and the consecutive repeated genes within each segment are dispersed to different positions.

5.3.2. Two-Dimensional Probability Model

Define $\mathbf{NS} = \left\{ NS^{G,k}(1), NS^{G,k}(2), \cdots, NS^{G,k}(l) \right\}$ as the neighborhood operations sequence executed by the k-th individual in generation G, where l represents the length of variable neighborhood search. The two-dimensional probability model $\mathbf{P^G(x,y)}$, shown in Equation (28), is used as the high-level strategy to guide the selection of the low-level variable neighborhood search operation set.

$$\mathbf{P^G(x,y)} = \begin{bmatrix} P^G_{l\times l}(1,1) & \cdots & P^G_{l\times l}(1,l) \\ \vdots & \ddots & \vdots \\ P^G_{l\times l}(l,1) & \cdots & P^G_{l\times l}(l,l) \end{bmatrix} \tag{28}$$

The initialization of the two-dimensional probability model and its update in the G-th generation are shown in Equations (29) and (30), respectively.

$$P^0(x,y) = 1/l,\ x,y = 1,2,\cdots,l \tag{29}$$

$$P^G(x,y) = (1+reward)\times P^{G-1}(x,y), x = 1,2,\cdots,l, y = NS(1), NS(2),\cdots,NS(l) \tag{30}$$

$$reward = \begin{cases} 0.1, \text{ new solution is not dominated by old solution} \\ -0.1, \text{ otherwise} \end{cases} \tag{31}$$

After completing the revolution operations, the cost of each country within the empire is calculated, and the country with the highest cost is selected as the new imperialist, while the original imperialist becomes a colony. $\mathbf{P^G(x,y)}$ is updated according to Equation (31).

5.4. Joint Imperialist Invasion

Dividing the population into multiple subpopulations is helpful for individuals to extensively search different areas and balances the exploration and exploitation capabilities of the algorithm through co-evolution [33]. To address the issues of poor diversity and premature convergence in the basic imperialist competitive algorithm, a joint imperialist competition mechanism is designed to replace the standard imperialist competition operations. Each empire is regarded as a population that collaboratively searches different regions of the solution space. This mechanism ensures that the collaborative evolutionary search is maintained within each empire and that information sharing and interaction are enhanced between empires, thereby improving diversity and the search efficiency of the algorithm.

The specific operations of the joint imperialist invasion are as follows: The total cost of each imperial group is calculated and ranked according to Equation (32), where Emp_k represents the set of colonies owned by Imp_k, and ξ is the proportional coefficient of the total cost of the empire accounted for by the colonies, which is set to 0.1. The imperialists of the top $N_{im}-1$ empires perform POX crossover [18] with random solutions, and the corresponding numbers of the worst colonies in the worst-ranked empire are replaced by the newly generated solutions. Through joint imperialist competition, the collaborative evolution within each empire can be maintained, while information sharing and interaction between empires can be enhanced, thus improving diversity and increasing the search efficiency of the algorithm. The pseudocode is shown in Algorithm 4.

$$TC^g_k = c^g_i + \xi \sum_{\sigma\in Emp_k} c^g_\sigma / NC_k \tag{32}$$

Algorithm 4. Joint imperialist invasion

Input: empires *Emp*, number of imperialists N_{im}

Output: updated *Emp* and new population $NewPop$

For $k = 1$ to N_{im} **do**

  Calculate the total cost of each empire and rank them in non-ascending order

End For

For $i = 1$ to $N_{im}-1$ **do**

  Perform POX between the imperialist and a random solution; the newly generated solution replaces the worst colony of the worst empire

End For

Merge all imperialist countries into $NewPop$

5.5. Algorithm Description

A flowchart of the proposed IICA-QL is shown in Figure 6. The process of IICA-QL is described as follows.

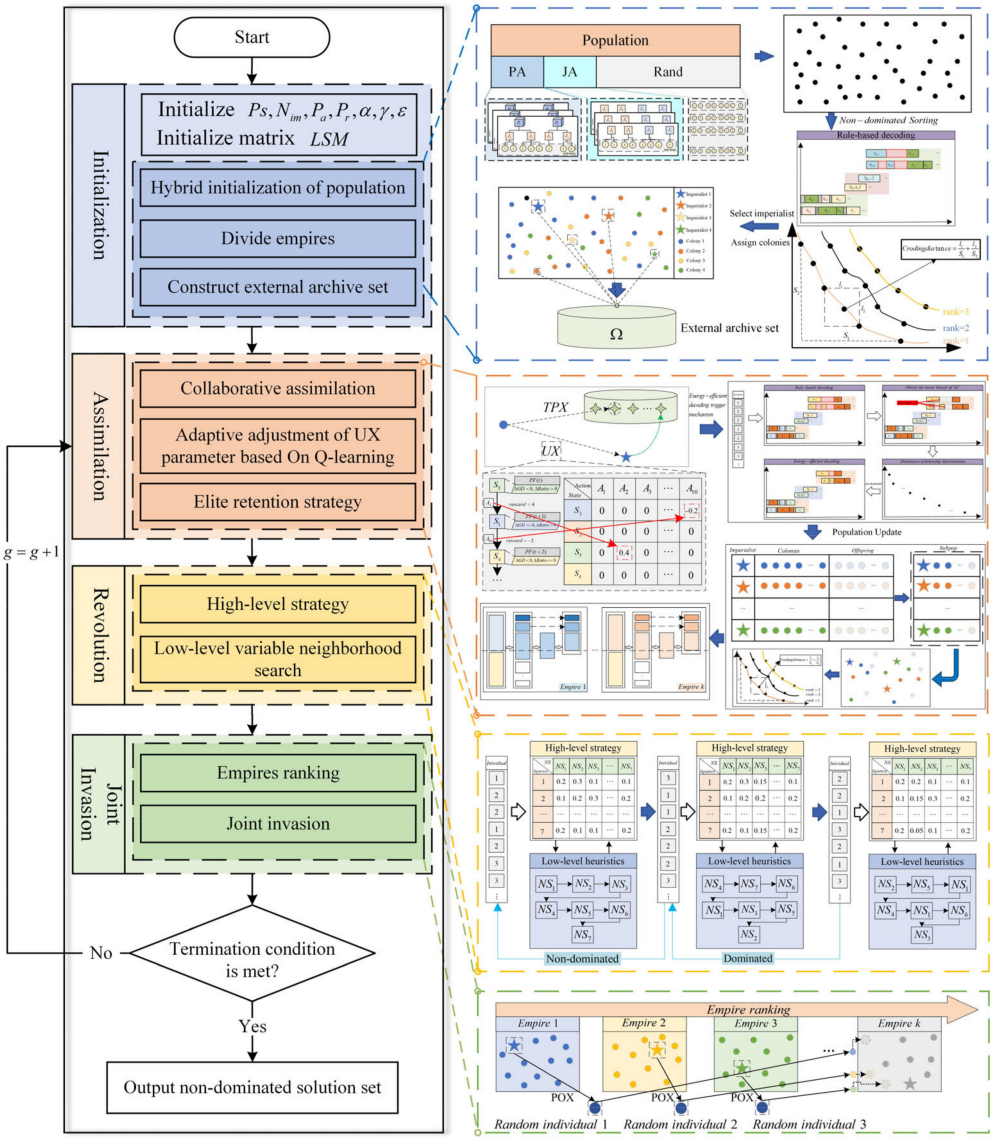


Figure 6. Framework of IICA-QL.

Step 1: Generate the initial population using heuristic rules and random generation methods. Select N_{im} imperialist countries to form the initial empires.

Step 2: Construct the external archive set and update it based on non-dominated solutions.

Step 3: Select the UX operator parameter according to the Q-table, then perform assimilation in each empire and update Q-table. Meanwhile, retain colonies based on the elite retention strategy.

Step 4: Execute high-dimensional strategies constructed by the two-dimensional probability model to select low-level neighborhood search operation sequences. Update the probability model based on feedback information.

Step 5: Perform joint imperialist competition to achieve the collaborative evolution of empires.

Step 6: If the termination condition is met, stop the search; otherwise, return to Step 3.

6. Experimental Results and Discussion

The proposed IICA-QL is implemented in MATLAB R2021a. The experimental configuration is an Intel Core i7-13700K and CPU @3.40 GHz PC with 32 GB memory (Intel, Santa Clara, CA, USA). Since no prior research has been conducted on the EEMsMlAJSP, the test data are generated randomly. The problem scale is denoted by O/J/A_M/T/K, where O/J/A represent the number of processing operations, jobs, and assembly operations, respectively. M/T/K represent the total number of processing machines, transport vehicles, and assembly machines, respectively. Detailed information about the product structure, order information, and production parameter data used in the experiments can be found from https://github.com/captain-dyq/EEMsMlAJSP.git accessed on 22 August 2024. The experimental time $T_{(i)}$ for each problem scale is determined based on the specific problem scale, where $T_{(i)} = O \times M + J \times T + A \times K$.

6.1. Performance Metrics

To comprehensively evaluate the convergence and diversity of the approximation Pareto fronts set obtained by the algorithm, the following three commonly used metrics are employed to measure the proposed IICA-QL.

(1) Solution set Coverage (C-metric). This metric measures the dominance relationship between two solution sets and is defined as follows:

$$C(A,B) = \frac{|x \in B | \exists y \in A : y \prec x|}{|B|} \tag{33}$$

where A and B are two non-dominated solution sets obtained by the algorithm. C(A,B) represents the proportion of solutions in set B that are dominated by at least one solution in set A. The larger the value of C(A,B), the better the performance of A.

(2) Generational Distance (GD).

The proximity of the Pareto front *PF* obtained by the algorithm to the optimal Pareto front *PF** is evaluated, where *PF** is the true non-dominated solution set among the non-dominated sets obtained by all algorithms. For each solution x in *PF*, the nearest solution y in *PF** is found, and their Euclidean distance is calculated. GD is the average of the shortest Euclidean distances, which is described as Equation (34). A smaller value of GD indicates better convergence and that the found *PF* is closer to the optimal Pareto front.

$$GD = \frac{\sqrt{\sum_{x \in PF} \min_{y \in PF^*} d(x,y)^2}}{|PF|} \tag{34}$$

(3) Hypervolume (HV)

The HV metric is a univariate metric that represents the volume of the region in the objective space enclosed by the non-dominated solution set obtained by the algorithm and the reference point, which is set to (1.01,1.01) [34]. The hypervolume metric is a comprehensive metric for evaluating the convergence and diversity of the non-dominated solution set obtained by the algorithm. The larger the HV value, the better the overall performance of the algorithm.

$$HV(E) = Leb(\cup_{x \in E}[f_1(x), r_1] \times \ldots \times [f_m(x), r_m]) \tag{35}$$

where $Leb(\cdot)$ represents the Lebesgue measure, and m represents the number of objectives in the problem.

To eliminate the impact of different dimensions, the solutions in the solution set are normalized according to the following formula [30]:

$$f_i' = \frac{f_i - f_i^{\min}}{f_i^{\max} - f_i^{\min}},\ i = 1,2 \tag{36}$$

6.2. Parameter Settings

To determine the optimal values for the parameters in IICA-QL, the Design of Experiment (DOE) method is used for parameter setting. IICA-QL has a total of seven parameters, each with three levels to choose from. The orthogonal array L27 (3^7) is selected, and the meanings and optional parameter levels for each parameter are shown in Table 2. Experiments are conducted using a medium-sized experimental scale (82/42/71_3/3/3), with each parameter combination run 10 times and the runtime being $T_{(i)} = O \times M + J \times T + A \times K$. The hypervolume (HV) metric is calculated after each run for each parameter combination, and the average of the ten HV values is used as the response value (RV). Additionally, the average RV and influence of each parameter level are shown in Table 3. According to Table 3, the optimal parameter set is $P_s = 10$, $N_{im} = 3$, $P_a = 0.8$, $P_r = 0.1$, $\alpha = 0.2$, $\gamma = 0.9$, and $\varepsilon = 0.6$.

Table 2. Parameter levels.

Parameter	Notation	Factor Level		
		1	**2**	**3**
Ps	population size	50	100	150
N_{im}	number of imperialists	3	5	7
P_a	assimilation probability	0.7	0.8	0.9
P_r	revolutionary probability	0.1	0.2	0.3
α	learning rate	0.1	0.2	0.3
γ	discount factor	0.7	0.8	0.9
ε	greed factor	0.3	0.6	0.9

Table 3. Average RVs and ranks of each parameter.

Level	P_s	N_{im}	P_a	P_r	α	γ	ε
1	0.7855	**0.7943**	0.7786	**0.7912**	0.7868	0.7844	0.7851
2	**0.7986**	0.7814	**0.7923**	0.7838	**0.7926**	0.7856	**0.7974**
3	0.7786	0.7869	0.7917	0.7877	0.7832	**0.7926**	0.7801
Delta	0.0200	0.0130	0.0138	0.0074	0.0094	0.0083	0.0174
Rank	1	4	3	7	5	6	2

6.3. The Effectiveness of the Adaptive Trigger Mechanism for the Energy-Efficient Strategy

To verify the effectiveness of the adaptive trigger mechanism for the energy-efficient strategy, IICA-QL (denoted as A) is compared with IICA-QL executing the energy-efficient strategy on Pareto solutions (IICA-QL-PFE, denoted as B) and IICA-QL executing the energy-efficient strategy on all individuals (IICA-QL-AE, denoted as C). Using the same computation time as the termination condition, each algorithm runs independently, 10 times on 18 test instances. The average values of C-metric, GD, and HV obtained are shown in Table 4, with the best value for each metric highlighted in bold. The runtime of each algorithm on each scale is recorded in Table 5.

Table 4. Comparison results between IICA-QL and IICA-QL-PFE/IICA-QL-AE.

O/J/A_M/T/K	C-Metric				GD			HV		
	C(A,B)	C(B,A)	C(A,C)	C(C,A)	A	B	C	A	B	C
25/13/22_2/2/2	**0.3839**	0.0400	**0.2750**	0.1600	**0.0020**	0.1062	0.0373	0.5436	0.5218	**0.6807**
25/13/22_3/2/2	**0.6563**	0.1050	**0.2900**	0.1409	**0.0339**	0.1394	0.0548	**0.6450**	0.5321	0.6181
25/13/22_3/3/3	**0.4100**	0.2100	**0.3000**	0.2000	**0.0717**	0.1468	0.1030	**0.6490**	0.5082	0.5965
44/23/39_2/2/2	**0.5657**	0.1820	**0.3150**	0.3000	**0.0574**	0.1509	0.0588	0.5470	0.3884	**0.6212**
44/23/39_3/2/2	**0.4458**	0.1650	**0.3700**	0.3450	**0.0731**	0.2145	0.1015	**0.7129**	0.3520	0.6705
44/23/39_3/3/3	**0.5582**	0.1350	**0.3400**	0.3133	0.1082	0.2129	**0.0442**	0.7045	0.3352	**0.7143**
82/42/71_2/2/2	**0.4906**	0.3996	**0.5576**	0.2904	0.1522	**0.1231**	0.1915	**0.6359**	0.5231	0.5631
82/42/71_3/2/2	**0.7106**	0.0514	**0.4873**	0.2845	**0.0939**	0.2222	0.0962	**0.7420**	0.4404	0.6939
82/42/71_3/3/3	**0.4310**	0.0681	0.2775	**0.5630**	0.1586	0.1963	**0.0408**	0.6667	0.2430	**0.7947**
135/70/120_3/3/3	**0.4000**	0.1630	**0.5083**	0.2593	**0.0631**	0.1542	0.0957	**0.6700**	0.3284	0.5469
135/70/120_4/3/3	0.2657	**0.3518**	**0.4769**	0.3038	0.1423	**0.0738**	0.1619	**0.7283**	0.6271	0.6485
135/70/120_4/3/4	**0.3667**	0.1672	**0.5594**	0.3510	**0.0932**	0.1349	0.1270	**0.7616**	0.4698	0.6264
173/88/151_3/3/3	**0.6350**	0.0258	**0.7278**	0.1113	**0.0119**	0.1721	0.1469	**0.7686**	0.1946	0.4786
173/88/151_4/3/3	**0.6333**	0.1595	**0.8169**	0.1060	**0.0669**	0.0912	0.2302	**0.7864**	0.4832	0.4958
173/88/151_4/3/4	**0.3667**	0.1445	**0.7105**	0.1496	**0.0328**	0.0415	0.1221	**0.8258**	0.5409	0.6232
202/104/175_3/3/3	**0.4300**	0.1385	**0.3846**	0.3389	0.0611	0.1018	**0.0579**	**0.7153**	0.2788	0.5539
202/104/175_4/3/3	**0.4333**	0.1494	**0.8427**	0.1018	**0.0283**	0.0734	0.1721	**0.7675**	0.4303	0.5742
202/104/175_4/3/4	**0.4500**	0.1076	**0.6800**	0.1413	**0.0248**	0.1309	0.0936	**0.7295**	0.2392	0.5453
Mean	**0.4796**	0.1535	**0.4955**	0.2478	**0.0709**	0.1381	0.1075	**0.7000**	0.4131	0.6137

Table 5. CPU time results of IICA-QL, IICA-QL-PFE, and IICA-QL-AE.

O/J/A_M/T/K	CPU Time/10 Generations (s)		
	IICA-QL	IICA-QL-PFE	IICA-QL-AE
25/13/22_2/2/2	10.1695	**8.3333**	12.7660
25/13/22_3/2/2	8.1006	**7.9235**	9.6667
25/13/22_3/3/3	8.3721	**8.1081**	9.7297
44/23/39_2/2/2	**9.7696**	10.1435	13.9474
44/23/39_3/2/2	10.5785	**10.1992**	14.0659
44/23/39_3/3/3	9.8452	**9.0857**	12.1839
82/42/71_2/2/2	**11.8182**	12.5000	21.3115
82/42/71_3/2/2	12.4211	**12.5199**	22.4762
82/42/71_3/3/3	12.8855	**12.8571**	19.9659
135/70/120_3/3/3	**15.5887**	15.8373	32.7723
135/70/120_4/3/3	**15.6354**	16.0340	32.6225
135/70/120_4/3/4	**16.1133**	16.1133	30.3883
173/88/151_3/3/3	**17.2385**	17.6320	42.9167
173/88/151_4/3/3	**17.0788**	17.6345	42.4398
173/88/151_4/3/4	17.1053	**16.7922**	36.7059
202/104/175_3/3/3	**18.2660**	18.4342	49.8577
202/104/175_4/3/3	18.0452	**17.8635**	49.1385
202/104/175_4/3/4	19.1857	**18.6786**	44.3970
Mean	13.7898	**13.7050**	27.6306

From the C-metric results in Table 4 and Figure 7, it can be seen that in all 18 test instances, C(A,B) is greater than C(B,A), and in 17 instances, C(A,C) is greater than C(C,A). This indicates that the non-dominated solutions obtained by IICA-QL generally dominate those obtained by IICA-QL-PFE and IICA-QL-AE. From the GD and HV metrics, it is evident that in most test instances, the values for IICA-QL are better than those for IICA-QL-PFE and IICA-QL-AE, and the values for IICA-QL-AE are better than those for IICA-QL-PFE. This shows that the adaptive trigger mechanism for the energy-efficient strategy improves the search quality of the algorithm by ensuring that it can more effectively explore and exploit the solution space, leading to better convergence speed and diversity. Furthermore, the superiority of IICA-QL is reinforced by the comparison of CPU times in

Table 5 and Figure 8, where the execution time of IICA-QL is comparable to that of IICA-QL-PFE and significantly faster than that of IICA-QL-AE. This demonstrates that while IICA-QL-PFE also applies the energy-efficient strategy, the adaptive trigger mechanism in IICA-QL ensures that potential solutions are not lost while simultaneously speeding up the convergence process by avoiding unnecessary evaluations and making more efficient use of computational resources. The significantly longer time required by IICA-QL-AE suggests that applying the energy-efficient strategy indiscriminately to all individuals results in an inefficient search. By contrast, the adaptive trigger mechanism in IICA-QL ensures a balance between search efficiency and solution quality, optimizing both the computational cost and the effectiveness of the search.

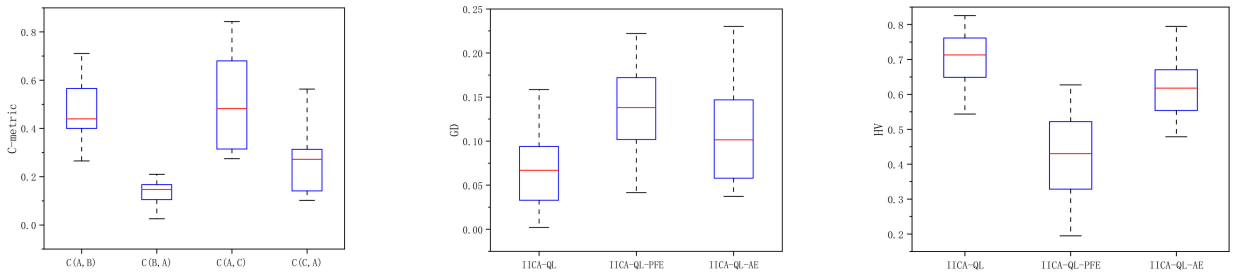


Figure 7. Box plots of three metrics between IICA-QL and IICA-QL-PFE/IICA-QL-AE.

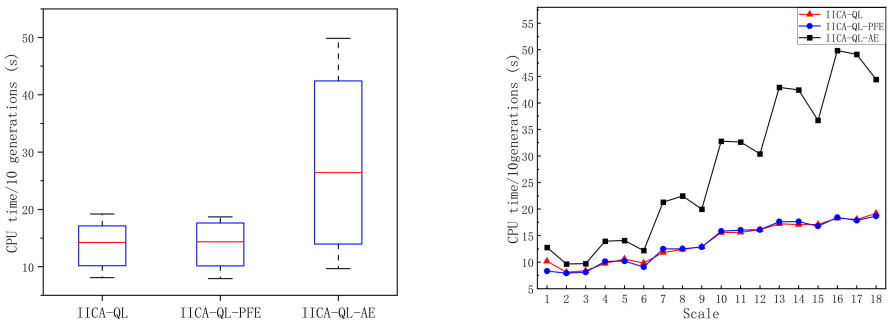


Figure 8. Box and line plots of CPU time between IICA-QL and IICA-QL-PFE/IICA-QL-AE.

6.4. Effectiveness of Operator Parameter Adaptive Adjustment Based on Q-Learning

To verify the effectiveness of the dynamic adaptive adjustment of the operator parameter based on Q-learning, IICA-QL (denoted as A) is compared with IICA-QL without this strategy (IICA-nQL, denoted as B), where the UX parameter is set to 0.55. Using the same computation time as the termination condition, each algorithm runs independently 10 times on 18 test instances. The average values of C-metric, GD, and HV obtained are shown in Table 6, with the best value for each metric highlighted in bold.

From the C-metric results in Table 6 and Figure 9, it can be seen that in 17 out of 18 test instances, C(A,B) is greater than C(B,A), and C(A,B) shows an increasing trend. This indicates that the non-dominated solutions obtained by IICA-QL generally dominate those obtained by IICA-nQL, and the advantage becomes more pronounced as the problem scale increases. From the GD and HV metrics, it is evident that in all test instances, the values for IICA-QL are better than those for IICA-nQL, indicating that IICA-QL has better convergence and diversity compared to IICA-nQL. This clearly shows that the dynamic adaptive adjustment strategy for the operator parameter based on Q-learning significantly improves the algorithm's performance, especially in larger and more complex problem instances. Additionally, this strategy ensures that the algorithm can maintain a robust balance between exploration and exploitation, further enhancing the quality of the Pareto front.

Table 6. Comparison results between IICA-QL and IICA-nQL.

O/J/A_M/T/K	C-Metric		GD		HV	
	IICA-QL(A)	IICA-nQL (B)	IICA-QL	IICA-nQL	IICA-QL	IICA-nQL
	C(A,B)	C(B,A)				
25/13/22_2/2/2	**0.3206**	0.3050	**0.0471**	0.0585	**0.6789**	0.6848
25/13/22_3/2/2	**0.3767**	0.2617	**0.0540**	0.0669	**0.6966**	0.6940
25/13/22_3/3/3	**0.3937**	0.2950	**0.0861**	0.1289	**0.7014**	0.6438
44/23/39_2/2/2	0.3283	**0.3679**	**0.0627**	0.0633	**0.7132**	0.6424
44/23/39_3/2/2	**0.3922**	0.3344	**0.0373**	0.1339	**0.7103**	0.6975
44/23/39_3/3/3	**0.5367**	0.3250	**0.0622**	0.1823	**0.7490**	0.6484
82/42/71_2/2/2	**0.5235**	0.2410	**0.0253**	0.1227	**0.7581**	0.6368
82/42/71_3/2/2	**0.4920**	0.3293	**0.0297**	0.0703	**0.7999**	0.7471
82/42/71_3/3/3	**0.4343**	0.3915	**0.0928**	0.1081	**0.7274**	0.7228
135/70/120_3/3/3	**0.6492**	0.3096	**0.0351**	0.0767	**0.7898**	0.7387
135/70/120_4/3/3	**0.5101**	0.3000	**0.0383**	0.0632	**0.7579**	0.7362
135/70/120_4/3/4	**0.5018**	0.4549	**0.0406**	0.0647	**0.7681**	0.7496
173/88/151_3/3/3	**0.5481**	0.3214	**0.0431**	0.0605	**0.7361**	0.7087
173/88/151_4/3/3	**0.6261**	0.3101	**0.0342**	0.1021	**0.7803**	0.7031
173/88/151_4/3/4	**0.4405**	0.4014	**0.0472**	0.1128	**0.8402**	0.7989
202/104/175_3/3/3	**0.5780**	0.2200	**0.0200**	0.1030	**0.7784**	0.6613
202/104/175_4/3/3	**0.6417**	0.2254	**0.0232**	0.1256	**0.8142**	0.6995
202/104/175_4/3/4	**0.6208**	0.3661	**0.0353**	0.0846	**0.8250**	0.7784
Mean	**0.4952**	0.3200	**0.0452**	0.0960	**0.7569**	0.7051

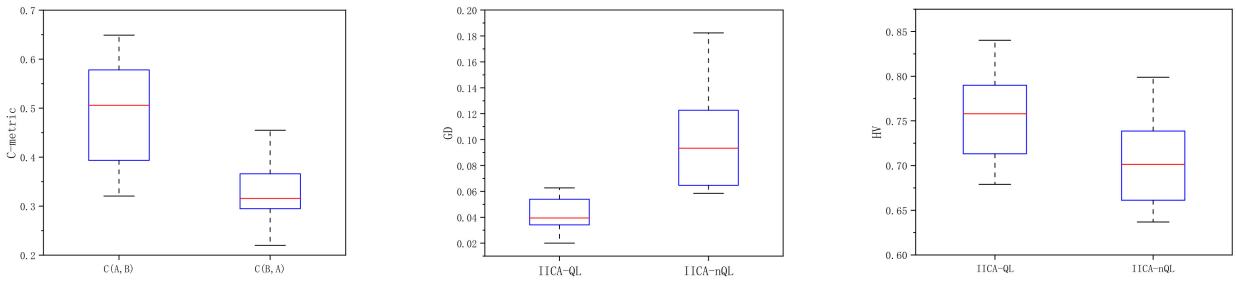


Figure 9. Box plots of three metrics between IICA-QL and IICA-nQL.

6.5. Comparison with Other Algorithms

Since no existing research has been conducted on solving the EEMsMlAJSP, to verify the effectiveness of the algorithm, IICA-QL is compared with the algorithms BPBMO [31], PSO-GA [35], and KBOA [28], which solve similar problems to the EEMsMlAJSP. The parameters for BPBMO, PSO-GA, and KBOA are chosen as specified in their respective references. Each of the four algorithms is independently run 10 times for the runtime determined by the problem scale, and the average values of the evaluation metrics are calculated. The C-metric, GD, and HV values for each algorithm across 18 problem scales are shown in Table 7, with the best value for each metric highlighted in bold.

As seen in Table 7 and Figure 10, the GD and HV values obtained by IICA-QL are superior to those of BPBMO, PSO-GA, and KBOA in all test instances. Specifically, the performance of IICA-QL in terms of the GD metric, demonstrating its ability to find solutions that are very close to the Pareto optimal front in the solution space, highlights its significant advantage in convergence. From the HV metric, it is evident that the values obtained by IICA-QL are much higher than those of the other three algorithms, indicating that IICA-QL not only finds high-quality solutions but also maintains solution diversity, covering a broader Pareto front region. These results clearly validate the effectiveness of IICA-QL in solving the EEMsMlAJSP, showcasing its superiority in both convergence and diversity compared to existing approaches.

Table 7. Comparison of GD and HV results for IICA-QL, BPBMO [31], PSO-GA [35], and KBOA [28].

O/J/A_M/T/K	GD				HV			
	IICA-QL	**BPBMO**	**PSO-GA**	**KBOA**	**IICA-QL**	**BPBMO**	**PSO-GA**	**BOA**
25/13/22_2/2/2	**0.0028**	0.2698	0.3198	0.2026	**0.7888**	0.4022	0.3688	0.5433
25/13/22_3/2/2	**0.0174**	0.1952	0.3166	0.2087	**0.8628**	0.5660	0.4059	0.5858
25/13/22_3/3/3	**0.0021**	0.4099	0.3654	0.2204	**0.8997**	0.3544	0.3958	0.5970
44/23/39_2/2/2	**0.0034**	0.3417	0.3449	0.2514	**0.8786**	0.3625	0.3508	0.4630
44/23/39_3/2/2	**0.0017**	0.4248	0.5155	0.3644	**0.9560**	0.4654	0.3790	0.5477
44/23/39_3/3/3	**0**	0.4992	0.3343	0.1943	**0.9531**	0.3060	0.3785	0.6688
82/42/71_2/2/2	**0.0013**	0.2599	0.4022	0.2959	**0.8988**	0.5376	0.3174	0.4242
82/42/71_3/2/2	**0.0127**	0.1507	0.3259	0.1707	**0.8421**	0.4690	0.3655	0.4644
82/42/71_3/3/3	**0.0075**	0.3547	0.3316	0.0941	**0.7589**	0.3434	0.3313	0.4765
135/70/120_3/3/3	**0.0003**	0.0777	0.3235	0.2528	**0.7709**	0.3298	0.2637	0.4445
135/70/120_4/3/3	**0.0005**	0.3859	0.4102	0.4588	**0.9723**	0.4449	0.3878	0.3356
135/70/120_4/3/4	**0.0010**	0.2352	0.4640	0.2639	**0.9323**	0.6075	0.3326	0.2695
173/88/151_3/3/3	**0.0031**	0.1326	0.2997	0.2859	**0.8063**	0.3339	0.3163	0.4845
173/88/151_4/3/3	**0**	0.0733	0.2559	0.1525	**0.7225**	0.3928	0.3541	0.4779
173/88/151_4/3/4	**0.0117**	0.0677	0.3560	0.2144	**0.7672**	0.6232	0.3032	0.3667
202/104/175_3/3/3	**0**	0.0676	0.2872	0.2337	**0.8153**	0.3370	0.2904	0.4662
202/104/175_4/3/3	**0.0049**	0.0878	0.2991	0.1570	**0.7052**	0.3648	0.2861	0.4667
202/104/175_4/3/4	**0.0217**	0.0575	0.3204	0.2630	**0.7515**	0.6474	0.3274	0.4378
Mean	**0.0051**	0.2273	0.3485	0.2380	**0.8379**	0.4382	0.3419	0.4733

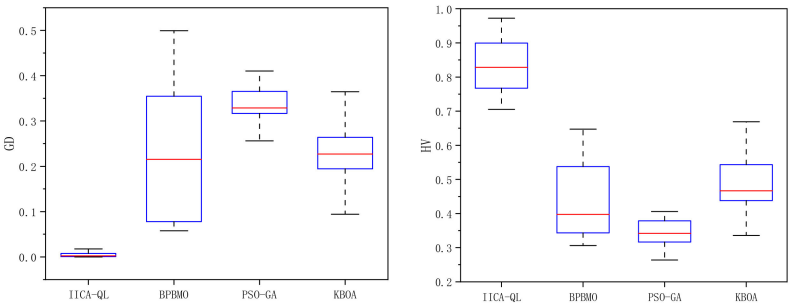


Figure 10. Box plots of two metrics between IICA-QL and BPBMO/PSO-GA/ KBOA.

To further illustrate the performance of the compared algorithms, Figure 11 presents the Pareto results obtained by four algorithms on small-scale (25/13/22_2/2/2), medium-scale (82/42/71_3/3/3), and large-scale (202/104/175_4/3/4) instances. As shown in the figure, the Pareto front obtained by IICA-QL outperforms those of the other three algorithms across different instances, indicating that IICA-QL is able to achieve non-dominated solutions with lower makespan (C_{max}) and total energy consumption (*TEC*) compared to BPBMO, PSO-GA, and KBOA. This further validates the effectiveness of IICA-QL in both convergence and diversity across different problem scales, as it consistently generates higher quality solutions with a broader coverage of the Pareto front.

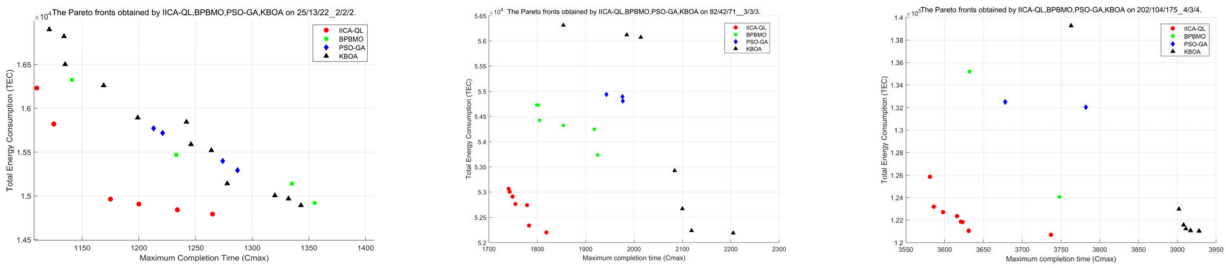


Figure 11. The Pareto fronts obtained by IICA-QL, BPBMO, PSO-GA, and KBOA with different scales.

6.6. Discussion

This paper presents a solution to the energy-efficient multi-stage multi-level assembly job shop scheduling problem (EEMsMlAJSP), a highly complex scheduling problem that arises in the manufacturing of multi-level products like ball valves and bogies. The multi-stage, multi-level structure of the EEMsMlAJSP involves not only the processing and assembly of various components but also intricate interdependencies between stages. These interdependencies often result in idle times, which contribute significantly to unnecessary energy consumption. The need to optimize both production efficiency and energy consumption makes the EEMsMlAJSP a challenging, yet important, problem in the manufacturing industry.

Through the research presented, the proposed IICA-QL has demonstrated its effectiveness in addressing the dual objectives of minimizing the maximum completion time and total energy consumption. The superiority of the proposed IICA-QL can be attributed to its two key designs:

(1) The energy-efficient decoding trigger mechanism is one of the key innovations of the IICA-QL. This mechanism ensures that, while maintaining high-quality solutions, the search process is significantly accelerated. This approach not only maintains the quality of the solutions but also expedites the convergence process, allowing the algorithm to explore the solution space more efficiently.
(2) IICA-QL achieves a balance between exploration and exploitation by using Q-learning to dynamically adjust operator parameters based on search feedback. This adaptive mechanism helps the algorithm explore new areas while refining high-quality solutions, ensuring efficient convergence, maintaining diversity, and avoiding premature stagnation.

7. Conclusions

This study addressed the multi-stage, multi-level assembly job shop scheduling problem, aiming to minimize the maximum completion time and total energy consumption. An improved imperialist competitive algorithm based on Q-learning (IICA-QL) was proposed. First, a mathematical model of the problem was established, and an energy-efficient strategy decoding trigger mechanism was designed to simultaneously enhance solution quality and search efficiency. Second, the IICA-QL for solving the EEMsMlAJSP was proposed. In IICA-QL, two heuristic rules and a random generation method are used to collaboratively construct an initial population with high quality and diversity. A dynamic adaptive adjustment strategy for the assimilation operator parameter was implemented based on Q-learning to enhance convergence speed while maintaining population diversity; thus, the exploration and exploitation capabilities of the algorithm are better balanced. A hyper-heuristic variable neighborhood search-guided revolution operation was designed to conduct detailed searches on identified high-quality colonies. The competitive phase was replaced with a joint imperialist invasion operation to achieve cooperative co-evolution and information exchange among multiple empires, resulting in more evenly distributed non-dominated solutions. Finally, the effectiveness of the proposed algorithm in solving the EEMsMlAJSP was verified through experimental results and comparison with other algorithms.

However, this study has certain limitations. First, the current model assumes static energy prices, but in real industrial environments, energy prices may fluctuate over time, which can significantly influence scheduling decisions. Therefore, future research could incorporate dynamic energy pricing models to better align with real production scenarios. Second, uncertainty is another important aspect not addressed in this study. We assumed a deterministic production environment, without considering uncertainties such as processing time fluctuations or machine breakdowns, which could greatly impact scheduling decisions in practice. Therefore, we will explore robust scheduling or scenario-based scheduling to address these uncertainties in future work.

Author Contributions: Conceptualization, Y.D. and W.L.; methodology, Y.D. and W.L.; software, Y.D.; formal analysis, Y.D., W.L. and G.X.; investigation, Y.D., W.L. and G.X.; resources, W.L.; data curation, Y.D., W.L. and G.X.; writing—original draft preparation, Y.D.; writing—review and editing, Y.D. and W.L.; visualization, G.X.; supervision, W.L.; project administration, Y.D. and W.L.; funding acquisition, W.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Chengdu Science and Technology Project of the Chengdu Science and Technology Bureau, Sichuan Province (NO: 2023-JB00-00020-GX).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are openly available in GitHub at https://github.com/captain-dyq/EEMsMlAJSP.git.

Acknowledgments: We would like to sincerely thank Bizhen Bao and Weigang Xu from Chengdu SIWI High-tech Industry Co., LTD. for their significant support in this research. They played crucial roles in investigation, data acquisition, and analysis. Additionally, they made important contributions to the review and refinement of the manuscript, providing valuable feedback that enhanced the quality of the final paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. He, L.; Chiong, R.; Li, W.; Dhakal, S.; Cao, Y.; Zhang, Y. Multiobjective Optimization of Energy-Efficient JOB-Shop Scheduling with Dynamic Reference Point-Based Fuzzy Relative Entropy. *IEEE Trans. Ind. Inform.* **2022**, *18*, 600–610. [CrossRef]
2. Duan, J. Energy-Efficient Collaborative Scheduling of Heterogeneous Multi-Stage Hybrid Flowshop for Large Metallic Component Manufacturing. *J. Clean. Prod.* **2022**, *375*, 134148. [CrossRef]
3. Zhu, Z.; Zhou, X.; Cao, D.; Li, M. A Shuffled Cellular Evolutionary Grey Wolf Optimizer for Flexible Job Shop Scheduling Problem with Tree-Structure Job Precedence Constraints. *Appl. Soft Comput.* **2022**, *125*, 109235. [CrossRef]
4. Zhao, F.; He, X.; Wang, L. A Two-Stage Cooperative Evolutionary Algorithm with Problem-Specific Knowledge for Energy-Efficient Scheduling of No-Wait Flow-Shop Problem. *IEEE Trans. Cybern.* **2021**, *51*, 5291–5303. [CrossRef]
5. Wang, W.; Tian, G.; Zhang, H.; Li, Z.; Zhang, L. A Hybrid Genetic Algorithm with Multiple Decoding Methods for Energy-Aware Remanufacturing System Scheduling Problem. *Robot. Comput.-Integr. Manuf.* **2023**, *81*, 102509. [CrossRef]
6. Cao, S.; Li, R.; Gong, W.; Lu, C. Inverse Model and Adaptive Neighborhood Search Based Cooperative Optimizer for Energy-Efficient Distributed Flexible Job Shop Scheduling. *Swarm Evol. Comput.* **2023**, *83*, 101419. [CrossRef]
7. Wang, Y.-J.; Li, J.; Wang, G.-G. Fuzzy Correlation Entropy-Based NSGA-II for Energy-Efficient Hybrid Flow-Shop Scheduling Problem. *Knowl.-Based Syst.* **2023**, *277*, 110808. [CrossRef]
8. Li, R.; Gong, W.; Wang, L.; Lu, C.; Zhuang, X. Surprisingly Popular-Based Adaptive Memetic Algorithm for Energy-Efficient Distributed Flexible Job Shop Scheduling. *IEEE Trans. Cybern.* **2023**, *53*, 8013–8023. [CrossRef]
9. Li, Y.; Yang, Z.; Wang, L.; Tang, H.; Sun, L.; Guo, S. A Hybrid Imperialist Competitive Algorithm for Energy-Efficient Flexible Job Shop Scheduling Problem with Variable-Size Sublots. *Comput. Ind. Eng.* **2022**, *172*, 108641. [CrossRef]
10. Lei, D.; Li, J. Distributed Energy-Efficient Assembly Scheduling Problem with Transportation Capacity. *Symmetry* **2022**, *14*, 2225. [CrossRef]
11. Hu, Y.; Zhang, L.; Zhang, Z.; Li, Z.; Tang, Q. Matheuristic and Learning-Oriented Multi-Objective Artificial Bee Colony Algorithm for Energy-Aware Flexible Assembly Job Shop Scheduling Problem. *Eng. Appl. Artif. Intell.* **2024**, *133*, 108634. [CrossRef]
12. Wang, J.; Tang, H.; Lei, D. A Q-Learning Artificial Bee Colony for Distributed Assembly Flow Shop Scheduling with Factory Eligibility, Transportation Capacity and Setup Time. *Eng. Appl. Artif. Intell.* **2023**, *123*, 106230. [CrossRef]
13. Wang, J.; Lei, D.; Cai, J. An Adaptive Artificial Bee Colony with Reinforcement Learning for Distributed Three-Stage Assembly Scheduling with Maintenance. *Appl. Soft Comput.* **2022**, *117*, 108371. [CrossRef]
14. Cheng, L.; Tang, Q.; Liu, S.; Zhang, L. Mathematical Model and Augmented Simulated Annealing Algorithm for Mixed-Model Assembly Job Shop Scheduling Problem with Batch Transfer. *Knowl.-Based Syst.* **2023**, *279*, 110968. [CrossRef]
15. Cheng, L.; Tang, Q.; Zhang, L. Production Costs and Total Completion Time Minimization for Three-Stage Mixed-Model Assembly Job Shop Scheduling with Lot Streaming and Batch Transfer. *Eng. Appl. Artif. Intell.* **2024**, *130*, 107729. [CrossRef]
16. Chen, J.; Wang, L.; Peng, Z. A Collaborative Optimization Algorithm for Energy-Efficient Multi-Objective Distributed No-Idle Flow-Shop Scheduling. *Swarm Evol. Comput.* **2019**, *50*, 100557. [CrossRef]
17. Fontes, D.B.M.M.; Homayouni, S.M.; Fernandes, J.C. Energy-Efficient Job Shop Scheduling Problem with Transport Resources Considering Speed Adjustable Resources. *Int. J. Prod. Res.* **2022**, *62*, 867–890. [CrossRef]
18. Gong, G.; Chiong, R.; Deng, Q.; Gong, X.; Lin, W.; Han, W.; Zhang, L. A Two-Stage Memetic Algorithm for Energy-Efficient Flexible Job Shop Scheduling by Means of Decreasing the Total Number of Machine Restarts. *Swarm Evol. Comput.* **2022**, *75*, 101131. [CrossRef]

19. Wang, J.-J.; Wang, L. A Cooperative Memetic Algorithm with Learning-Based Agent for Energy-Aware Distributed Hybrid Flow-Shop Scheduling. *IEEE Trans. Evol. Comput.* **2022**, *26*, 461–475. [CrossRef]
20. Meng, L.; Zhang, C.; Zhang, B.; Gao, K.; Ren, Y.; Sang, H. MILP Modeling and Optimization of Multi-Objective Flexible Job Shop Scheduling Problem with Controllable Processing Times. *Swarm Evol. Comput.* **2023**, *82*, 101374. [CrossRef]
21. Li, M.; Lei, D. An Imperialist Competitive Algorithm with Feedback for Energy-Efficient Flexible Job Shop Scheduling with Transportation and Sequence-Dependent Setup Times. *Eng. Appl. Artif. Intell.* **2021**, *103*, 104307. [CrossRef]
22. Lei, D.; Li, M.; Wang, L. A Two-Phase Meta-Heuristic for Multiobjective Flexible Job Shop Scheduling Problem with Total Energy Consumption Threshold. *IEEE Trans. Cybern.* **2019**, *49*, 1097–1109. [CrossRef] [PubMed]
23. Li, M.; Lei, D.; Cai, J. Two-Level Imperialist Competitive Algorithm for Energy-Efficient Hybrid Flow Shop Scheduling Problem with Relative Importance of Objectives. *Swarm Evol. Comput.* **2019**, *49*, 34–43. [CrossRef]
24. Zhou, R.; Lei, D.; Zhou, X. Multi-Objective Energy-Efficient Interval Scheduling in Hybrid Flow Shop Using Imperialist Competitive Algorithm. *IEEE Access* **2019**, *7*, 85029–85041. [CrossRef]
25. Chen, Y.; Zhong, J.; Mumtaz, J.; Zhou, S.; Zhu, L. An Improved Spider Monkey Optimization Algorithm for Multi-Objective Planning and Scheduling Problems of PCB Assembly Line. *Expert Syst. Appl.* **2023**, *229*, 120600. [CrossRef]
26. Chen, R.; Yang, B.; Li, S.; Wang, S. A Self-Learning Genetic Algorithm Based on Reinforcement Learning for Flexible Job-Shop Scheduling Problem. *Comput. Ind. Eng.* **2020**, *149*, 106778. [CrossRef]
27. Li, R.; Gong, W.; Lu, C.; Wang, L. A Learning-Based Memetic Algorithm for Energy-Efficient Flexible Job-Shop Scheduling with Type-2 Fuzzy Processing Time. *IEEE Trans. Evol. Comput.* **2023**, *27*, 610–620. [CrossRef]
28. Pan, Z.; Wang, L.; Wang, J.; Yu, Y. Distributed Energy-Efficient Flexible Manufacturing with Assembly and Transportation: A Knowledge-Based Bi-Hierarchical Optimization Approach. *IEEE Trans. Autom. Sci. Eng.* **2024**. [CrossRef]
29. Zou, P.; Rajora, M.; Liang, S.Y. A New Algorithm Based on Evolutionary Computation for Hierarchically Coupled Constraint Optimization: Methodology and Application to Assembly Job-Shop Scheduling. *J. Sched.* **2018**, *21*, 545–563. [CrossRef]
30. Zhu, Z.; Zhou, X. An Efficient Evolutionary Grey Wolf Optimizer for Multi-Objective Flexible Job Shop Scheduling Problem with Hierarchical Job Precedence Constraints. *Comput. Ind. Eng.* **2020**, *140*, 106280. [CrossRef]
31. Li, J.; Han, Y.; Gao, K.; Xiao, X.; Duan, P. Bi-Population Balancing Multi-Objective Algorithm for Fuzzy Flexible Job Shop with Energy and Transportation. *IEEE Trans. Autom. Sci. Eng.* **2023**, *21*, 4686–4702. [CrossRef]
32. Ming, M.; Trivedi, A.; Wang, R.; Srinivasan, D.; Zhang, T. A Dual-Population-Based Evolutionary Algorithm for Constrained Multiobjective Optimization. *IEEE Trans. Evol. Comput.* **2021**, *25*, 739–753. [CrossRef]
33. Cheng, L.; Tang, Q.; Zhang, L.; Meng, K. Mathematical Model and Enhanced Cooperative Co-Evolutionary Algorithm for Scheduling Energy-Efficient Manufacturing Cell. *J. Clean. Prod.* **2021**, *326*, 129248. [CrossRef]
34. Wang, J.; Wang, L.; Xiu, X. A Cooperative Memetic Algorithm for Energy-Aware Distributed Welding Shop Scheduling Problem. *Eng. Appl. Artif. Intell.* **2023**, *120*, 105877. [CrossRef]
35. Ren, W.; Wen, J.; Yan, Y.; Hu, Y.; Guan, Y.; Li, J. Multi-Objective Optimisation for Energy-Aware Flexible Job-Shop Scheduling Problem with Assembly Operations. *Int. J. Prod. Res.* **2021**, *59*, 7216–7231. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.