Journal of Industrial and Management Optimization

JIMO, 22(8): 3926–3967.
DOI: 10.3934/jimo.2026140
Received: 07 April 2026
Revised: 17 June 2026
Accepted: 06 July 2026
Published: 23 July 2026

https://www.aimspress.com/journal/jimo

Research article

Energy-efficient distributed flexible job shop scheduling with variable processing speeds: a Q-learning hyper-heuristic algorithm

Haojie Li[1], Qifang Luo[1,2] and Yongquan Zhou[1,2,*]

[1] College of Artificial Intelligence, Guangxi Minzu University, Nanning, 530006 China
[2] Guangxi Key Laboratories of Hybrid Computation and IC Design Analysis, Nanning, 530006 China

* **Correspondence:** Email: yongquanzhou@126.com.

Abstract: This paper addresses the energy-efficient distributed flexible job shop scheduling problem (EEDFJSP) with the dual objectives of minimizing makespan and total energy consumption (TEC). In this problem, jobs are assigned to multiple factories, each containing several machines that can operate at variable speed levels, leading to a trade-off between productivity and energy efficiency. To solve this complex multi-objective optimization problem, a Q-learning-based hyper-heuristic algorithm (QL-HH) is proposed. The algorithm incorporates a hybrid initialization strategy combining four heuristic rules (NR2, longest remaining workload, minimum completion time, and speed-level heuristics) to generate a high-quality and diverse initial population. A Q-learning framework with an improved dynamic ε-greedy mechanism is developed to adaptively select eight low-level heuristic (LLH) operators according to the current state of each individual, which is characterized by normalized objective values and crowding distance. Additionally, critical path-based speed adjustment operators are designed to reduce energy consumption without increasing makespan. Extensive experiments are conducted on 23 benchmark instances, including high-flexibility and low-flexibility datasets derived from classical FJSP instances. The results demonstrate that QL-HH consistently outperforms four state-of-the-art algorithms (QMOEA/D-AWA, MOEA/D, NSGA-II, and KBEA) in terms of hypervolume (HV) and inverted generational distance (IGD), with statistical significance confirmed by Wilcoxon and Friedman tests.

Keywords: distributed flexible job shop scheduling; energy-efficient scheduling; multi-objective optimization; Q-learning; hyper-heuristic
Mathematics Subject Classification: 90B35, 90C29, 90C59, 68T05

1. Introduction

Flexible job shop scheduling is a classic problem in the industrial field. At present, related research has formed a comprehensive system, and many effective scheduling algorithms have been developed [1,2]. The practical characteristics of this problem vary significantly across different application scenarios. Additionally, as industry standards for simulation accuracy continue to improve, the academic community has gradually shifted its research focus toward derived and extended models of flexible job shop scheduling [3]. The energy-efficient distributed flexible job shop scheduling problem (EEDFJSP) is one such novel extension that has attracted considerable attention. This model adopts a multi-factory cooperative operation mode and, building on the traditional objective of minimizing the makespan, further incorporates minimizing energy consumption into the optimization scope [4]. Constrained by factors such as the multi-objective optimization nature, massive data volumes, and a sharp increase in problem complexity, traditional algorithms often fail to obtain high-quality scheduling solutions [5,6].

For conventional medium- to large-scale FJSP, exact algorithms are limited by their enormous computational requirements and cannot meet practical application needs [7]. Consequently, approximate algorithms have become the mainstream solution approach. In terms of classification, approximate algorithms can be divided into traditional heuristics and metaheuristics. Heuristics adjust a single solution according to predefined rules, where the design of these rules fully incorporates problem-specific characteristics and mathematical principles. Metaheuristics, on the other hand, operate on a population of multiple solutions and achieve global optimization through stochastic search.

Most early academic studies on the FJSP adopted heuristic methods for problem-solving. Whether these algorithms are centered on scheduling rules or supported by industry prior knowledge, they can effectively avoid redundant search behaviors and eliminate a substantial amount of ineffective search space. Such algorithms typically employ greedy strategies to iteratively update the current solution, gradually improving its performance, thereby achieving high-efficiency convergence. Perez and Raupp [8] proposed a novel hierarchical heuristic algorithm for solving the multi-objective FJSP, which is a modification of Newton's method for continuous multi-objective unconstrained optimization problems and belongs to the class of multi-objective descent methods. Ziaee [9] employed a multi-factor criterion to schedule job operations, in which each factor is assigned two weights. Wang et al. [10] introduced a heuristic algorithm named HFBS (filtered beam search), which successfully identifies near-optimal schedules for the FJSP while maintaining reasonable computational time. Thanks to their clear search logic, heuristic methods often yield high-quality scheduling solutions. However, such algorithms are prone to premature convergence and struggle to escape local optima.

In terms of practical solving performance, metaheuristic methods generally exhibit superior capability for solving highly complex FJSPs [11]. In multi-objective flexible job shop scheduling problems, conflicts of interest exist among different optimization objectives, making it impossible to rely on a single solution to satisfy all requirements [12]. Therefore, it is necessary to generate a diverse set of solutions. Against this backdrop, the application value of metaheuristic methods becomes prominent, and related algorithms centered on multi-objective evolutionary optimization have become the mainstream choice for solving such problems [13,14]. Frutos et al. [15] introduced a memory algorithm based on NSGAII that operates on two chromosomes and incorporates simulated annealing as a local search in the genetic stage to solve the FJSP. Jiang and Wang [16] proposed a hybrid multi-objective evolutionary algorithm based on decomposition (HMOEA/D) for the FJSP under time-of-

use electricity prices (FJSP-TOUEP). They introduced a cooperative search operator, two local intensification operators, an adaptive selection strategy, and an adjustment strategy tailored to time-of-use tariffs.

Despite the widespread adoption of metaheuristics, they often suffer from high computational cost, sensitive parameter tuning, and difficulty in generalizing across different problem instances [17]. To overcome these limitations, hyper-heuristics have emerged as a higher-level search methodology that manages a set of low-level heuristics, aiming to select or generate effective heuristics adaptively [18]. Unlike traditional heuristics that operate directly on the problem domain, a hyper-heuristic algorithm operates on a pool of existing heuristics, separating the high-level strategy from the specific problem context [19]. This separation allows algorithm designers to focus on the design of high-level strategies without being entangled in domain-specific details. Consequently, hyper-heuristics typically feature simple operations, few parameters, and strong generalization capability. Similar to ensemble learning, hyper-heuristics organically combine the advantages of multiple individual heuristics, thereby improving robustness and solution quality. For example, Fu et al. [20] proposed a mixed integer programming model for integrated production and distribution in a distributed flexible job shop environment, aiming to minimize the maximum completion time. They developed a Q-learning-based evolutionary algorithm with hybrid search strategies, incorporating three heuristics for initialization, problem-specific operators, and an adaptive strategy to select effective search strategies. Likewise, a genetic programming hyper-heuristic (GPHH) was developed for DAPFSP with sequence-dependent setup times, using genetic programming to generate effective LLH sequences [21]. These successes demonstrate the feasibility and effectiveness of hyper-heuristics for complex scheduling problems.

Meanwhile, Q-learning, as a classic reinforcement learning technique, enables agents to learn optimal policies through trial-and-error interactions with the environment, showing strong adaptability for complex scheduling decisions [22]. Compared with traditional meta-optimization procedures that are time-consuming, Q-learning offers unique advantages in intelligently guiding the selection of LLHs without requiring expensive offline tuning. The integration of Q-learning with hyper-heuristics has proven successful in various domains. For instance, Lin et al. [23] proposed a Q-learning-based hyper-heuristic (QHH) for the capacitated electric vehicle routing problem (CEVRP). The method includes a novel initialization, a repair method for charging station insertion, and Q-learning as a high-level heuristic to dynamically select low-level heuristics (LLHs). A Q-learning-based hyper-heuristic (QHH) was also developed for the semiconductor final testing scheduling problem (SFTSP), using Q-learning as the high-level strategy to select heuristics from an LLH set [24]. These insights motivate the design of a Q-learning-based hyper-heuristic (QL-HH) to address the EEDFJSP. The synergy of hyper-heuristics and Q-learning not only enhances solution quality and algorithmic efficiency but also provides adaptive and generalized scheduling support, which is particularly beneficial for energy-aware distributed manufacturing environments.

With the continuous enrichment of production control standards and the increasing number of influencing factors in manufacturing sites, the limitations of traditional FJSP models have gradually become apparent, as they can no longer accurately reflect the actual operating status of industrial workshops [25,26]. Driven by the practical demands of industrial development, a vast amount of research has been conducted on the FJSP in related fields [5,27]. Researchers have continuously proposed new scheduling models that are more aligned with real production conditions and involve more complex problem dimensions, along with corresponding algorithm design and solution analysis [28]. Meanwhile, to adapt to the trend of large-scale production and comprehensively improve the operational

efficiency of manufacturing processes, industrial production has gradually shifted toward an organizational model of multi-factory collaborative operation [29]. This cross-factory production structure breaks the operational boundaries of a single workshop and poses new challenges to traditional scheduling theory. Against this backdrop, numerous researchers have turned their attention to the field of multi-factory collaborative scheduling and have conducted extensive and in-depth research on the distributed flexible job shop scheduling problem (DFJSP) [2,30]. De et al. [31] proposed an improved genetic algorithm to solve the DFJSP, aiming to minimize the global makespan, with extended gene encoding, greedy decoding, and a new local search-based operator. Chang et al. [32] proposed a hybrid genetic algorithm for the DFJSP, which optimizes GA parameters via the Taguchi method and adopts a novel encoding mechanism along with improved crossover and mutation operators. Marzouki et al. [33] proposed a decentralized model based on tabu search to solve the DFJSP with the objective of minimizing makespan. Wu et al. [34] proposed a machine degradation model, a dual-threshold preventive maintenance strategy, and a hybrid artificial bee colony algorithm for the integrated scheduling problem. Zhang et al. [35] proposed a DQN-aided variable neighborhood search algorithm (DQN-VNS) to solve the distributed heterogeneous flexible job-shop scheduling problem (DHFJSP) for minimizing makespan. Zhao et al. [36] proposed a Q-learning-based heterogeneous graph reinforcement learning framework (QIHGRL) for dynamic FJSS that uses problem-aware attention to encode features and inter-option attention to balance action selection.

With the continuous expansion of industrial production scales, issues such as pollutant emissions and excessive resource consumption have become increasingly prominent, posing significant challenges to the global ecological environment [37]. To implement the concept of green manufacturing and alleviate environmental pressure, how to reduce energy consumption and improve energy efficiency in production activities has gradually become a key issue in the transformation and upgrading of the manufacturing industry [28,38]. As a mainstream form of production organization, the flexible job shop's scheduling scheme directly determines equipment operation duration, process arrangement, and overall energy consumption [39]. Based on this practical need, energy-efficient flexible job shop scheduling (EEFJSP), which balances production objectives and energy consumption indicators, has begun to receive in-depth investigation by numerous scholars [40]. Wang et al. [41] applied Q-learning to adaptively select the number of neighborhood solutions in MOEA/D-AWA for low-carbon FJSP. Zhang et al. [42] proposed a distributed contrastive multi-agent deep reinforcement learning framework (DCMAD3QN) to solve dynamic FJSP for minimizing tardiness. Li et al. [43] proposed an NSGA-II-based method for energy-efficient FJSP under dynamic events (e.g., new job arrivals and machine failures). The method updates task and machine status, considers both machining and idle energy consumption, and minimizes total energy and makespan. As research on EFJSP continues to increase, Hassandokami et al. [44] reviewed environment-oriented shop scheduling problems, including energy-efficient scheduling models and their corresponding solution methods.

Combining the dual development trends of green and low-carbon transformation in manufacturing and large-scale collaborative production, both energy efficiency optimization and distributed multi-factory collaborative production have become indispensable core considerations in the field of modern intelligent manufacturing, possessing high engineering practical value and research significance. In addressing the resulting energy-efficient distributed flexible job shop scheduling problem (EEDFJSP), Pan et al. [45] proposed a bi-learning evolutionary algorithm (BLEA) for the problem with transportation constraints (DEFJSP-T). BLEA integrates statistical learning and evolutionary learning, searches for beneficial substructures on weight vectors, performs local exploitation, and adopts a

dynamic switching mechanism to allocate computational resources. Yu et al. [46] proposed a knowledge-guided bi-population evolutionary algorithm (KBEA) for the EEDFJSP, aiming to minimize makespan and total energy consumption (TEC). The algorithm features a problem-specific initialization strategy, five adaptive evolutionary operators, a knowledge-guided local search, and an elaborately designed energy-saving strategy. Pan et al. [47] proposed a knowledge-based two-population optimization (KTPO) algorithm to solve the distributed energy-efficient parallel-machines scheduling problem (DEPMSP), which integrates factory and machine assignment to minimize TEC and total tardiness. Wang et al. [48] proposed an EC-IIoT-based distributed and flexible job shop real-time scheduling (DFJS-RS) method to enhance real-time decision-making in intelligent manufacturing. The method integrates edge computing with IIoT, consisting of shop-floor task assignment and flexible manufacturing unit-level real-time scheduling solved via an evolutionary game-based approach.

Although several existing studies have addressed the energy-efficient distributed flexible job shop scheduling problem (EEDFJSP), most either assume fixed machine speeds or rely on predefined heuristic rules that lack adaptability to different search stages; conventional metaheuristics often suffer from high parameter sensitivity, premature convergence, and difficulty in balancing makespan with TEC. To overcome these limitations, this study further introduces machine processing speed as an additional decision variable; while this extension makes the scheduling model more realistic by reflecting adjustable machine speeds in actual shop-floor conditions, it also significantly increases problem complexity due to the expanded coupling between speed selection and task sequencing. Nevertheless, the proposed Q-learning-based hyper-heuristic is well-suited to handle such complexity, as its adaptive learning mechanism can effectively navigate the enlarged solution space. The main contributions are as follows:

(1) A hybrid initialization strategy combining four heuristic rules (NR2 for factory assignment, longest remaining workload for operation sequencing, minimum completion time for machine allocation, and speed-level heuristics) is designed to generate a diverse and high-quality initial population.

(2) A Q-learning framework with an improved dynamic ε-greedy mechanism is developed, where the agent adaptively selects among eight low-level heuristic operators based on each individual's normalized objective values and crowding distance, enabling context-aware search that balances exploration and exploitation.

(3) Eight problem-specific low-level heuristic operators are introduced, covering factory assignment, machine allocation, operation sequencing, and speed selection. Among them, critical path-based speed reduction and hybrid speed adjustment are specifically designed to balance makespan and TEC effectively.

(4) The contribution of each component (hybrid initialization, Q-learning mechanism, and critical path-based speed adjustment) is verified, followed by experimental comparisons with several state-of-the-art algorithms. The experimental results demonstrate the superior performance of the proposed QL-HH algorithm in solving the energy-efficient distributed flexible job shop scheduling problems.

The rest of this paper is structured as follows: Section 2 presents the mathematical model of the EEDFJSP. Section 3 details the proposed QL-HH algorithm, including encoding, initialization, Q-learning components, low-level heuristics, and overall procedure. Section 4 describes the experimental setup, parameter tuning, and comparative results. Finally, Section 5 concludes the paper and discusses future research directions.

2. Definition and model

2.1. Problem definition

The EEDFJSP can be described as follows: There are N independent jobs $I = \{1, 2, \ldots, N\}$ that need to be distributed across NF factories $\boldsymbol{F} = \{1, 2, \ldots, NF\}$, with each factory f equipped with NM machines. Each job J_i consists of N_i operations, and these operations must be processed sequentially in strict accordance with the precedence constraints of the process. Each operation $O_{i,j}$ of job J_i can be processed on a subset of machines across all factories, and its processing time varies depending on the specific machine selected for processing. The final schedule aims to simultaneously optimize multiple objectives, including minimizing the maximum completion time (C_{max}) of all jobs and reducing the TEC during the production process. In addition, the EEDFJSP involves several basic assumptions:

- All machines and jobs are available at time zero.
- Each machine can process at most one operation at a time.
- Each operation should be processed non-preemptively without interruption.
- Neither transportation time nor setup time is considered.

2.2. Mathematical model

To formulate the EEDFJSP with variable processing speed constraints, a mixed-integer linear programming (MILP) model is established [49]. The symbols and detailed descriptions are presented in Table 1.

Table 1. Symbols used in the model and descriptions.

Notation	Description
Indices	
f	Factory index; $f = 1, 2, \ldots, NF$.
i, i'	Job index; $i, i' = 1, 2, \ldots, N$.
j, j'	Operation index of a job; $j, j' = 1, 2, \ldots, N_i$.
k, k'	Machine index; $k, k' = 1, 2, \ldots, NM$.
v	Speed-level index; $v = 1, 2, \ldots, NS$.
Sets	
$\boldsymbol{I}$	Complete job set, $\boldsymbol{I} = \{1, 2, \ldots, N\}$.
$\boldsymbol{J_i}$	Operation set of job i, $\boldsymbol{J_i} = \{1, 2, \ldots, N_i\}$.
$\boldsymbol{F}$	Factory set, $\boldsymbol{F} = \{1, 2, \ldots, NF\}$.
$\boldsymbol{V}$	Available speed levels, $\boldsymbol{V} = \{1, 2, \ldots, NS\}$.
$\boldsymbol{K_{i,j}}$	Eligible machines for operation $O_{i,j}$.
$\boldsymbol{E}$	Set of energy consumption coefficient, $\boldsymbol{E} = \{1, 2, \ldots, NE\}$.
Parameters	
F_f	The fth factory.
M_k	The kth machine.
J_i	The ith job.
$O_{i,j}$	The jth operation of job i.

Continued on next page

Notation	Description
Vv	The vth speed level.
E_v	The vth energy consumption coefficient.
NF	Total count of factories.
N	Total count of jobs.
N_i	Operation count for job i.
NM	Number of machines per factory.
NS	Total available speed levels.
SP	Energy consumption per unit time during standby.
PP	Energy consumption per unit time during processing.
$t_{k,i,j}$	The nominal processing duration of $O_{i,j}$ on machine k in each factory.
$p_{k,i,j}$	The actual processing duration of $O_{i,j}$ on machine k in each factory.
$PEC_{f,k}$	Total energy consumption due to processing on machine k in factory f.
$SEC_{f,k}$	Total energy consumption during standby periods of machine k in factory f.
$C_{f,k,i,j}$	The complete time of operation $O_{i,j}$ on machine k in factory f.
C_f	The maximum completion time of factory f.
L	A sufficiently large positive constant.
Objective function	
$C_{\max}$	The makespan (maximum completion time of all factories).
TEC	The total energy consumption.
Binary decision variables	

$$X_{f,k,i,j} = \begin{cases} 1, \text{ if } O_{i,j} \text{ is assigned to machine } k \text{ in factory } f, \\ 0, \text{ other.} \end{cases}$$

$$Y_{i,j,i',j'} = \begin{cases} 1, \text{ if } O_{i,j} \text{ is scheduled before } O_{i',j'} \text{ on the same machine,} \\ 0, \text{ other.} \end{cases}$$

$$Z_{f,k,i,j,v} = \begin{cases} 1, \text{ if operation } O_{i,j} \text{ is processed on machine } k \text{ at speed } v \text{ in factory } f, \\ 0, \text{ other.} \end{cases}$$

$$U_{i,f} = \begin{cases} 1, \text{if job } i \text{ is assigned to factory } f, \\ 0, \text{other} \end{cases}$$

Table 2. Details of processing speed levels.

v	1	2	3	4	5
V_v	0.6	0.8	1	1.2	1.4
E_v	8	5	3	2	1

Referring to the work by Zhang et al. [50], the unit energy consumptions for processing and standby are set as $PP = 3$ and $SP = 1$, respectively. The speed coefficients and corresponding energy consumption coefficients for each level are shown in Table 2. The mathematical model is presented as follows:

$$f = \min\{f_1, f_2 \mid f_1 = C_{\max}, f_2 = TEC\} \tag{2.1}$$

$$C_{\max} = \max\left\{C_{f,k,i,j}\right\}, \forall f,k,i,j \tag{2.2}$$

$$\sum_{f=1}^{NF} U_{i,f} = 1, \forall i \tag{2.3}$$

$$\sum_{f=1}^{NF}\sum_{k=1}^{NM} X_{f,k,i,j} = 1, \forall i,j \tag{2.4}$$

$$\sum_{k=1}^{NM} X_{f,k,i,j} \le U_{i,f}, \forall f,i,j \tag{2.5}$$

$$X_{f,k,i,j} = 0, \forall f,i,j,k \notin K_{i,j} \tag{2.6}$$

$$\sum_{v=1}^{NS} Z_{f,k,i,j,v} = X_{f,k,i,j}, \forall f,k,i,j \tag{2.7}$$

$$p_{k,i,j} = t_{k,i,j} \times \left(\sum_{v=1}^{NS} Z_{f,k,i,j,v} \times V_v\right), \forall f,k,i,j \tag{2.8}$$

$$C_{f,k,i,j} \ge C_{f,k',i,j-1} + p_{k,i,j} - L \times (2 - X_{f,k',i,j-1} - X_{f,k,i,j}), \forall f,k,k',i,j \ge 2 \tag{2.9}$$

$$C_{f,k,i',j'} \ge C_{f,k,i,j} + p_{k,i',j'} - L \times (1 - Y_{i,j,i',j'}), \forall f,k,i,i',j,j' \tag{2.10}$$

$$C_{f,k,i,j} \ge C_{f,k,i',j'} + p_{k,i,j} + L \times Y_{i,j,i',j'}, \forall f,k,i,i',j,j' \tag{2.11}$$

$$C_{f,k,i,j} \ge 0, \forall f,k,i,j \tag{2.12}$$

$$PEC_{f,k} = \sum_{i=1}^{N}\sum_{j=1}^{N_i}\left(p_{k,i,j} \times \sum_{v=1}^{NS} Z_{f,k,i,j,v} \times E_v\right) \times PP \tag{2.13}$$

$$SEC_{f,k} = \left(C_f - \sum_{i=1}^{N}\sum_{j=1}^{N_i} p_{k,i,j} \times X_{f,k,i,j}\right) \times SP \tag{2.14}$$

$$TEC = \sum_{f=1}^{NF}\sum_{k=1}^{NM}\left(PEC_{f,k} + SEC_{f,k}\right) \tag{2.15}$$

Eq.(2.1) aims to minimize C_{max} and TEC. Eq.(2.2) provides the formula for calculating C_{max}. Eq.(2.3) indicates that each job can only be assigned to one factory, and all its operations must be processed within the same factory. Eq.(2.4) represents the unique assignment constraint for operations. Each operation must be assigned to exactly one machine. Eq.(2.5) is the consistency constraint between jobs and factories. If job i is assigned to factory f (i.e., $Z_{i,f} = 1$), all its operations can only be allocated to machines in this factory. Eq.(2.6) restricts that each operation can only be processed on its eligible machine set $K_{i,j}$. Eq.(2.7) is the speed selection constraint. If an operation is assigned to a machine, a corresponding speed must be selected. Eq.(2.8) calculates the actual processing time, which equals the standard processing time multiplied by the speed coefficient. Eq.(2.9) denotes the precedence constraint for the same job, meaning the subsequent operation can only start after the preceding one is completed. Eq.(2.10)–Eq.(2.11) are mutual exclusion constraints for different operations on the same machine. Eq.(2.12) indicates that the completion time of all operations is non-negative. Eq.(2.13)–Eq.(2.15) define the calculation formulas for $PEC_{f,k}$, $SEC_{f,k}$, and TEC, respectively.

3. QL-HH for EEDFJSP

3.1. Solution representation

For the EEDFJSP, considering variable processing speed constraints, the solution representation can be decomposed into four interdependent components, denoted as $\pi = [\pi^F, \pi^O, \pi^M, \pi^V]$. Specifically, π^F represents the job assignment vector, π^O denotes the operation sequencing vector, π^M corresponds to the machine allocation vector, and π^V stands for the processing speed assignment vector. As illustrated in Table 3, an illustrative dataset is provided, where the symbol '-' indicates that the corresponding operation is not allowed to be processed on the designated machine. A feasible encoding scheme derived from this dataset is depicted in Figure 1, and the associated scheduling Gantt chart is presented in Figure 2.

Table 3. Sample dataset.

Jobs	Operations	Available machines and standard processing times		
		M_1	M_2	M_3
$\boldsymbol{J_1}$	$O_{1,1}$	5	3	-
	$O_{1,2}$	-	4	6
	$O_{1,3}$	2	-	4
$\boldsymbol{J_2}$	$O_{2,1}$	6	5	-
	$O_{2,2}$	-	7	4
$\boldsymbol{J_3}$	$O_{3,1}$	-	5	8
	$O_{3,2}$	3	-	6
	$O_{3,3}$	4	-	9
	$O_{3,4}$	-	7	5
$\boldsymbol{J_4}$	$O_{4,1}$	7	-	5
	$O_{4,2}$	8	6	-
	$O_{4,3}$	-	4	9
$\boldsymbol{J_5}$	$O_{5,1}$	6	-	4
	$O_{5.2}$	-	5	9

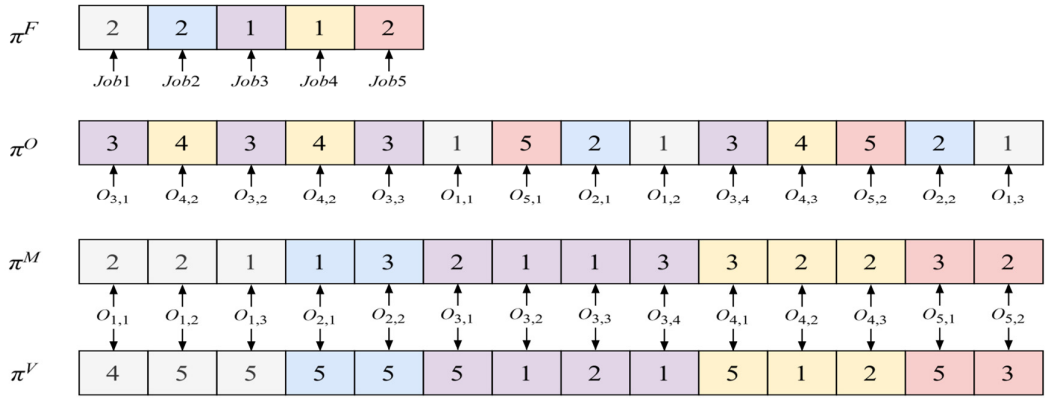


Figure 1. A feasible coding scheme.

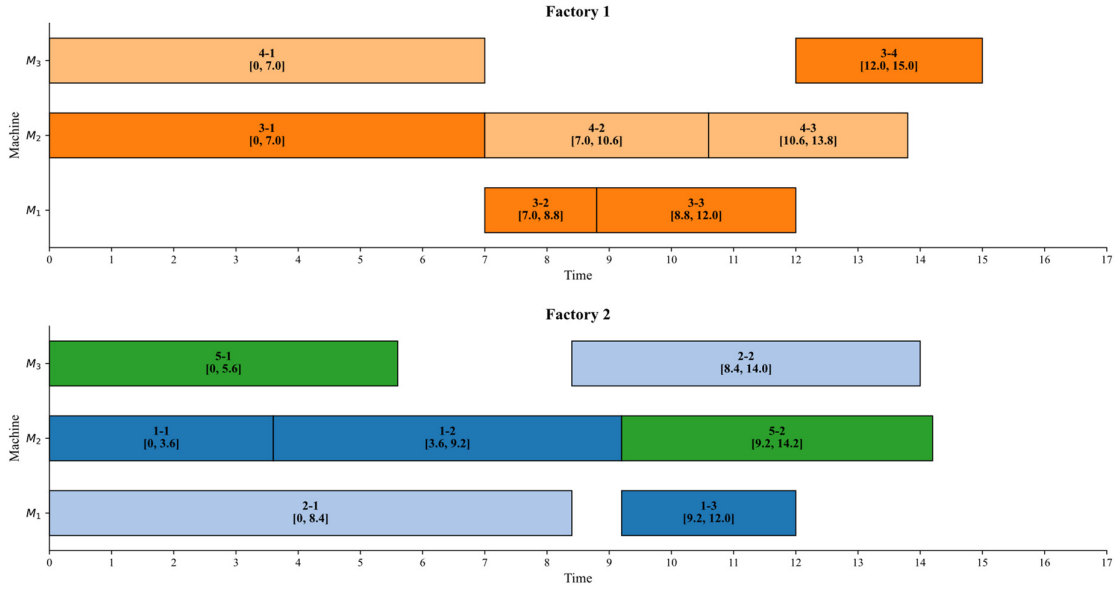


Figure 2. Sample Gantt chart.

3.2. Population initialization

Effective population initialization is a cornerstone practice for algorithms. It serves to populate the search space with diverse and promising candidate solutions, thereby laying a solid foundation for effective exploration and exploitation. A robust initialization method can significantly enhance the algorithm's capacity to escape local optima and accelerate its convergence to the global optimum, ultimately leading to higher quality optimization outcomes. We complete the population initialization by combining heuristic methods and random methods. For the job assignment vector π^F, we adopt the NR2 [51] and random assignment methods, where the random assignment method attempts to distribute jobs as evenly as possible across each factory. For the operation sequence vector π^O, the

longest remaining workload rule (LRW) [52], the maximum remaining operations rule (MRO) [53], and the random rule are employed. For the machine assignment vector π^M, the minimum completion time rule (MCT) [53], the shortest processing time of operation rule (SPT) [53], and the random rule are used. Here, the random rule randomly selects one machine from the set of available machines for the operation, ensuring the feasibility of the encoding. For the speed assignment vector π^V, the minimum speed, maximum speed, and random rules are applied. The detailed steps are shown in Algorithm 1.

Algorithm 1: Hybrid initialization

Input: Population size N
Output: Initial population P_0
$P_0 \leftarrow \varnothing$
for $i = 1$ to N **do**
    // 1. Factory assignment π^F
    randomly choose NR2 or uniform random assignment;
    // 2. Operation sequencing π^O
    randomly choose LRW, MRO, or random order;
    // 3. Machine allocation π^M
    randomly choose MCT, SPT, or random eligible machine;
    // 4. Speed selection π^V
    randomly choose minimum speed, maximum speed, or random speed;
    Evaluate the individual;
    Add the generated individual to P_0;
end for
return P_0

3.3. Q-learning methods

Reinforcement learning has been widely applied to various complex multi-objective optimization problems (MOPs), such as path planning, resource allocation, and robot control [54]. As a typical model-free reinforcement learning method, Q-learning has become one of the most widely used algorithms in this field due to its advantages, including the simple structure and stable performance. By iteratively updating the Q-value function of state-action pairs, Q-learning achieves continuous optimization of decision-making behaviors, thereby learning the optimal decision strategy in the corresponding environment.

Hyper-heuristic approaches generally operate at two hierarchical levels: a high-level strategy and a set of low-level heuristics (LLHs). Two core components constitute the high-level strategy, namely the heuristic selection mechanism and the acceptance criterion [55]. The heuristic selection mechanism is responsible for determining which low-level heuristic to execute, while the acceptance criterion is subsequently employed to judge whether the newly generated solution should be accepted. Low-level heuristics represent a collection of problem-specific heuristic operators tailored to address the target problem. In this study, the proposed QL-HH adopts a Q-learning-based high-level strategy, which dynamically selects the most appropriate low-level heuristic according to the problem state during the evolutionary process. Its framework is shown as follows.

3.3.1. Agent and state

The agent is the decision-making entity that learns to select appropriate heuristic operators according to the current search situation. The state must capture the status of an individual with respect to both convergence and diversity within the population, thereby enabling the agent to distinguish between different optimization phases. In this work, the state of an individual is constructed from three complementary indicators: the normalized objective values of the two scheduling objectives ($C_{\max}$ and TEC) and the crowding distance. These three measures are discretized into distinct levels to form a finite state space.

For a given population P with size N, denote the two objective values of individual i as $f_1(i)$ ($C_{\max}$) and $f_2(i)$ (TEC). To eliminate scale differences, each objective is normalized as:

$$\tilde{f}_k(i) = \frac{f_k(i) - \min_{j \in P} f_k(j)}{\max_{j \in P} f_k(j) - \min_{j \in P} f_k(j)},\ k \in \{1, 2\} \tag{3.1}$$

A small value of $\tilde{f}_k(i)$ indicates a better performance in that objective. The normalized value of each objective is divided into three intervals, corresponding to good, medium, and poor performance. The division is performed according to fixed thresholds τ_1 and τ_2 (here set to 1/3 and 2/3).

To reflect the density distribution of an individual along the current Pareto front, the crowding distance $d(i)$ is computed in the standard manner: after sorting the population by each objective, the distance to the nearest neighbors is summed over both objectives. The crowding distances of all individuals are then sorted, and the population is split into three groups based on quartiles:

$$lev_{cd}(i) = \begin{cases} 1,\ d(i) < Q_1 \\ 2,\ Q_1 < d(i) < Q_2 \\ 3,\ d(i) < Q_2 \end{cases} \tag{3.2}$$

where Q_1 and Q_2 denote the lower and upper quartiles of the crowding distance distribution. These three levels correspond to dense, moderate, and sparse distribution regions, respectively.

The total number of possible states is 3×3×3=27, which is sufficient to characterize individual states in a sufficiently detailed and concise manner. Through this representation, the Q-learning agent can distinguish different individual conditions, thereby achieving context-based operator selection throughout the evolutionary process.

3.3.2. Action

In the Q-learning framework, the action set defines the low-level heuristic (LLH) operators that the agent can select for decision-making at each step, which are used to perturb the current scheduling solution to generate neighborhood candidate solutions. The design of the action set determines the search space coverage capability and adaptability of the hyper-heuristic algorithm.

For the EEDFJSP studied in this paper, eight low-level heuristic operators are designed, which act on different dimensions of the scheduling solution, including factory assignment, operation sequencing, machine allocation, and processing speed selection. These operators perturb the solution from different perspectives, covering various granularities such as fine-tuning and structural adjustment, aiming to

balance the conflicts between the two optimization objectives and enhance search diversity. The formal definition of the action set is given as $A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8\}$, where each action ai corresponds to a specific heuristic operator. The detailed definitions of the eight operators will be elaborated in Section 3.4. By maintaining a diverse action set, the Q-learning agent can adaptively select the most appropriate operator according to the current search state, thereby achieving a dynamic balance between global exploration and local exploitation.

3.3.3. Reward function

The reward function is the core feedback mechanism in Q-learning, providing the agent with a quantitative evaluation of the action taken. A properly designed reward function enables the agent to distinguish beneficial actions from detrimental ones, thereby guiding the learning process toward an optimal decision policy. In this work, the reward function is formulated to incorporate both the Pareto dominance relationship between the new solution and the original solution, as well as the changes in objective values and crowding distance.

Let x_{old} denote the incumbent individual before applying an action, and x_{new} denote the newly generated individual after executing the selected heuristic operator. The crowding distances of these two solutions within the current population are denoted as $d(x_{old})$ and $d(x_{new})$, respectively. The reward R is defined according to the following piecewise function:

$$R = \begin{cases} +2, \ x_{new} \prec x_{old} \\ +0.6 \cdot \Delta_{obj} + 0.4 \cdot \Delta_{cd}, \ x_{new} \parallel x_{old} \\ -0.5, \ x_{old} \prec x_{new} \\ 0, \text{ otherwise} \end{cases} \tag{3.3}$$

where $x_{new} \prec x_{old}$ indicates that the new solution Pareto dominates the old one, $x_{new} \ // \ x_{old}$ indicates that the two solutions are non-dominated with respect to each other, and $x_{old} \prec x_{new}$ indicates that the new solution is dominated by the old one.

When the two solutions are non-dominated, the reward is composed of two complementary terms. The first term Δ_{obj} reflects the improvement in objective values, defined as:

$$\Delta_{obj} = \frac{1}{2} \sum_{k \in \{1,2\}} \left(\tilde{f}_k(x_{old}) - \tilde{f}_k(x_{new}) \right) \tag{3.4}$$

where $\tilde{f}_k(x)$ denotes the normalized value of objective k for solution x within the current population. This term encourages actions that move the solution closer to the Pareto front.

The second term Δ_{cd} captures the change in crowding distance:

$$\Delta_{cd} = d(x_{new}) - d(x_{old}) \tag{3.5}$$

This term is specifically designed to promote actions that increase the diversity of the solution set. Even when a new solution does not dominate the old one, if it moves into a sparser region of the objective space (i.e., increases its crowding distance), it receives a positive contribution to the reward. Conversely, if the new solution becomes more crowded, the reward is reduced. This design aligns with

the state representation introduced earlier, where crowding distance is a key component of the state, thereby closing the feedback loop between state characterization and reward evaluation.

By incorporating both objective improvement and diversity enhancement into the reward signal, the proposed reward function provides the Q-learning agent with nuanced feedback. This design enables the agent to learn action selection strategies that effectively balance convergence toward the Pareto front and maintenance of population diversity, thereby enhancing the overall performance of the hyper-heuristic algorithm.

3.3.4. Improved ε-greedy strategy

In Q-learning, the balance between exploration and exploitation is critical for effective learning [36]. Exploration encourages the agent to try different actions to discover potentially superior strategies, while exploitation leverages the current knowledge to select actions with the highest estimated value. A standard ε-greedy strategy selects a random action with probability ε and the greedy action (the one with the highest Q-value) with probability 1 - ε. However, a fixed ε value often fails to adapt to the changing needs of the search process, leading to either excessive randomness in later stages or insufficient exploration in early stages.

To address this limitation, this work adopts an improved dynamic ε-greedy strategy that adjusts the exploration probability over the course of the evolutionary process. The core idea is to maintain a high level of exploration in the early generations to broadly cover the solution space and gradually shift toward exploitation as the search progresses to refine promising regions. The exploration probability ε at iteration t is defined as:

$$\varepsilon(t) = 0.1 + \frac{0.8}{1 + e^{8\times(t/T_{\max} - 0.5)}} \tag{3.6}$$

$$a_t = \begin{cases} \text{random action from } \mathcal{A},\ \text{rand}(0,\ 1) < \varepsilon(t) \\ \arg\max_{a\in\mathcal{A}} Q(s_t, a),\ \text{otherwise} \end{cases} \tag{3.7}$$

where rand (0,1) denotes a uniformly distributed random number in the interval (0,1). When exploration is triggered, any of the eight low-level heuristic operators may be selected with equal probability, allowing the agent to explore diverse search directions. When exploitation is triggered, the agent selects the action with the highest Q-value for the current state, thereby applying the operator that has historically yielded the best outcomes under similar search conditions.

The dynamic adjustment of ε is tightly coupled with the state representation and reward function described earlier. As the search evolves and the agent accumulates experience, the decreasing exploration probability allows the learning process to transition from broad exploration of the solution space to focused exploitation of promising regions identified by the Q-table. This adaptive mechanism helps the algorithm avoid premature convergence while ensuring efficient local refinement in later stages, ultimately contributing to the overall effectiveness of the hyper-heuristic framework.

3.3.5. Q-table update

The Q-table is the core memory structure in Q-learning, storing the estimated value of taking each action in each possible state. Through iterative updates, the agent gradually refines these estimates to

approximate the optimal action-value function, thereby learning a policy that maximizes cumulative rewards over the long term.

In this work, the Q-table is implemented as a two-dimensional matrix of size $|S| \times |A|$, where S denotes the state space, and A denotes the action set. As established in Section 3.4, the state space contains 27 discrete states, and the action set contains 8 low-level heuristic operators. The Q-value $Q(s, a)$ represents the expected cumulative reward when taking action a in state s and following the current policy thereafter. At each decision step, after executing an action a_t in state s_t and receiving an immediate reward R_t, the agent updates the corresponding Q-value according to equation Eq. (3.8).

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[R_t + \gamma \cdot \max_{a \in \mathcal{A}} Q(s_{t+1}, a_t) - Q(s_t, a_t) \right] \tag{3.8}$$

where α is the learning rate, controlling the step size of each update. A larger α places more weight on recent experiences, while a smaller α leads to slower but more stable learning. γ is the discount factor, determining the importance of future rewards relative to immediate rewards. A higher γ encourages the agent to consider long-term consequences of its actions, which is particularly important in scheduling problems where the impact of an operator may not be immediately apparent. R_t is the immediate reward obtained after executing action a_t. $\max_{a \in A} Q(s_{t+1}, a)$ represents the maximum estimated value for the next state s_{t+1}, capturing the optimal future reward that the agent expects to obtain from that state onward.

3.4. Critical path analysis

Critical path analysis is a fundamental tool for identifying bottleneck operations that determine the overall makespan of a schedule. In the context of EEDFJSP, the critical path is defined as the longest chain of operations from the start of the first operation to the completion of the last operation, considering precedence constraints and processing times. Any delay on a critical operation directly increases the maximum completion time $C_{\max}$, whereas non-critical operations possess positive slack (float) that allows them to be delayed without affecting $C_{\max}$.

As illustrated in Figure 3, a typical schedule can be decomposed into a critical path (highlighted in red) and several parallel non-critical paths. Operations lying on the critical path are termed critical operations; all remaining operations are non-critical operations. The key insight for energy saving is that reducing the processing speed (i.e., increasing the processing time) of a non-critical operation does not postpone the overall makespan as long as its slack is not exhausted; for example, appropriately slowing down the operations $O_{2,1}$, $O_{2,2}$, and $O_{3,3}$ in the figure. Conversely, decelerating a critical operation would lengthen the critical path and thus increase $C_{\max}$, which is usually undesirable.

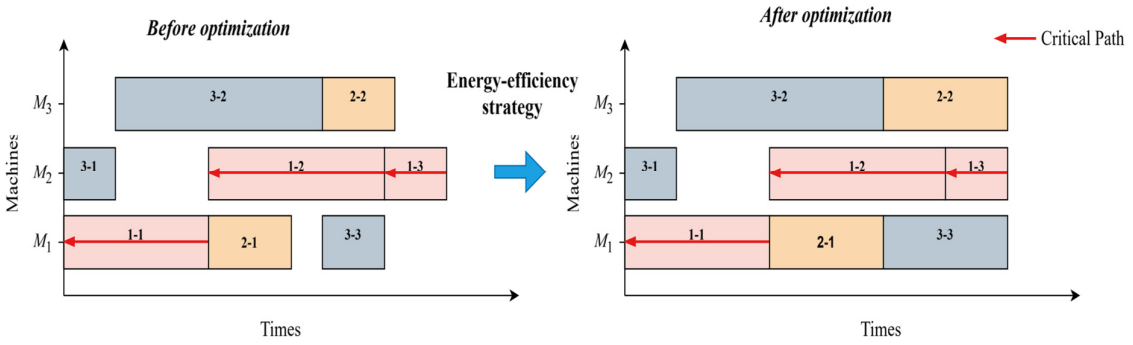


Figure 3. Energy efficiency based on the critical path.

3.5. Low-level heuristics

The low-level heuristics (LLH1-LLH8) constitute the operational core of the hyper-heuristic framework, providing a diverse set of perturbation mechanisms that modify different components of a scheduling solution. Each LLH is designed to target a specific aspect of the EEDFJSP factory assignment, machine allocation, operation sequencing, or speed selection, thereby enabling the Q-learning agent to explore distinct regions of the solution space. In this context, the critical factory is defined as the factory with the maximum makespan among all factories, and the critical machine is defined as the machine with the maximum completion time within a factory. For clarity of presentation, the eight LLHs are grouped according to their primary focus, as summarized below.

(1) Factory-level operators

LLH1: Factory reassignment. A job is randomly selected from the factory with the largest makespan and reassigned to another eligible factory. This operator helps balance workload distribution across distributed facilities.

LLH2: Factory-level job exchange. Two jobs belonging to different factories are randomly selected, and their factory assignments are swapped. This operator performs reciprocal exchange, preserving the total number of jobs per factory while redistributing workload.

(2) Machine-level operators

LLH3: Machine reassignment. A random operation is selected, and its assigned machine is changed to a different eligible machine. This operator exploits the flexibility of machine allocation to reduce processing times or balance machine utilization.

LLH4: Within the critical factory, the machine with the highest load is identified, and a random operation from that machine is reassigned to another available machine. This operator targets bottleneck resources that directly influence the overall makespan.

(3) Operation sequencing operators

LLH5: Operation sequence swap. Two operations belonging to different jobs are randomly selected, and their positions in the operation sequence are exchanged. This operator explores alternative processing orders while respecting job precedence constraints.

LLH6: Operation sequence segment reversal. Within the critical factory, a contiguous segment of the operation sequence is randomly selected and reversed. This operator introduces structural diversity to escape local optima.

(4) Speed adjustment operators

LLH7: Non-critical speed reduction. Operations not located on the critical path have their processing speed reduced by one level. Since these operations do not affect the overall makespan, lowering their speed reduces energy consumption.

LLH8: Hybrid speed adjustment. For a randomly selected job, operations on the critical path are accelerated, while operations on non-critical paths are decelerated. This operator explicitly targets the trade-off between makespan and energy consumption.

(5) Operator summary

The eight operators are carefully constructed to balance the trade-off between the two optimization objectives and to leverage the structural properties of the problem, particularly the critical path concept introduced in Section 3.4. Among them, LLH7 and LLH8 explicitly incorporate critical path analysis to adjust processing speeds. By focusing speed adjustments on operations that do not affect the total makespan, the algorithm effectively reduces TEC while maintaining or even optimizing the makespan. The eight low-level heuristics can be categorized by their operational scope, as shown in Table 4.

Table 4. Main optimization objectives of LLHs.

Operator	Scope	Primary focus
LLH1	Factory (critical)	Workload balance
LLH2	Factory	Workload redistribution
LLH3	Machine	Processing efficiency
LLH4	Machine (critical)	Bottleneck alleviation
LLH5	Sequence	Schedule structure
LLH6	Sequence (critical factory)	Structural diversity
LLH7	Speed (non-critical)	Energy reduction
LLH8	Speed (hybrid)	Balanced optimization

This diverse operator pool provides the Q-learning agent with a comprehensive set of search actions, each suited to different search states. By learning to select the most appropriate operator based on the current state of an individual, the agent effectively balances convergence toward the Pareto front with the maintenance of population diversity throughout the evolutionary process.

3.6. QL-HH procedure

Algorithm 2 outlines the overall procedure of QL-HH. First, an initial population P_0 of size N is generated using hybrid heuristic strategies. The Q-table is initialized with zeros for all state-action pairs. In each generation, for every individual x in the current parent population, the algorithm extracts its state s_t, selects an action a_t via the dynamic ε-greedy strategy, applies the corresponding low-level heuristic to produce an offspring x', and computes the reward R_t. The Q-table is then updated. After

processing all individuals, parent and offspring populations are merged, followed by fast non-dominated sorting and crowding distance assignment. The best N individuals are selected as the next generation. The algorithm terminates after $T_{\max}$ generations and returns the non-dominated solution set. Furthermore, the flowchart of the QL-HH algorithm is shown in Figure 4.

Algorithm 2: QL-HH procedure

Input: Population size N, maximum iterations Tmax, learning rate α, discount factor γ, instance data.

Output: Non-dominated solution set.

// Initialization
Generate initial population P_0 of size N using hybrid heuristic strategies (Algorithm 1);
Evaluate all individuals in P_0 (compute $C_{\max}$ and TEC);
Initialize Q-table with $Q(s, a) = 0$ for all $s \in$ S, $a \in$ A;
for $t = 1$ to $T_{\max}$ **do**
    for each individual $x \in P_{t-1}$ **do**
        Extract current state $s_t = (lev_1(x), lev_2(x), lev_{cd}(x))$;
        Compute exploration probability $\varepsilon(t)$ using dynamic ε-greedy strategy;
        Select action a_t using Eq.(3.7);
        Apply operator a_t to x to generate offspring x';
        Evaluate x';
        Compute reward R_t based on comparison between x' and x;
        Extract next state s_{t+1} for x';
        Update Q-table using Eq.(3.8) ;
        Add x' to offspring population O_t;
    end for
    Merge parent and offspring: $P_t = P_{t-1} \cup O_t$;
    Perform fast non-dominated sorting on P_t;
    Compute crowding distance for individuals in the same front;
    Select the best N individuals to form P_t (higher rank first, then larger crowding distance);
end for
Return: all individuals in the first non-dominated front.

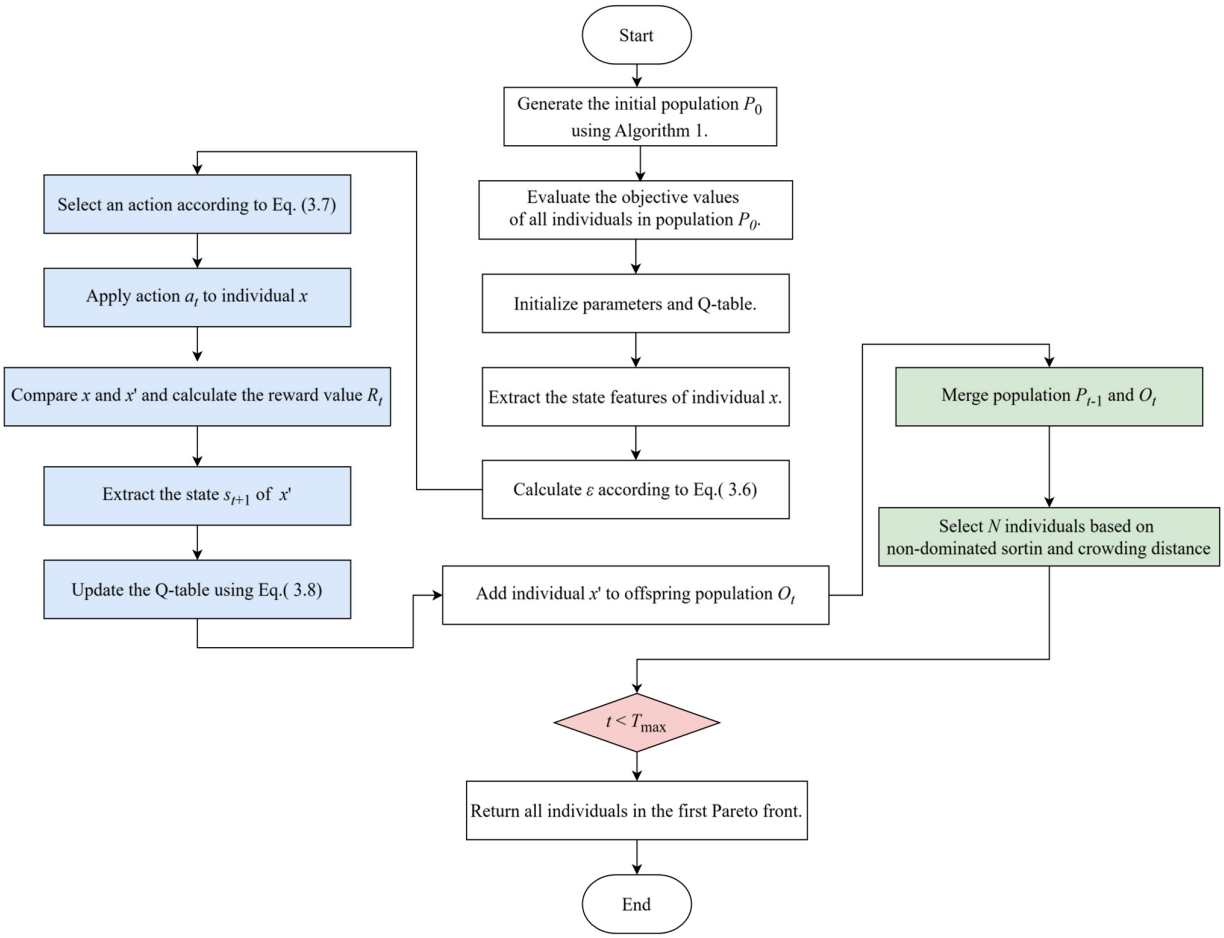


Figure 4. Flowchart of the QL-HH.

3.7. Time complexity analysis

This section provides a concise analysis of the time complexity of the proposed QL-HH. The analysis focuses on the following parameters: N: population size; P: total number of operations across all jobs; M: number of machines per factory; F: number of factories; and T: maximum number of generations.

During population initialization, generating N individuals using hybrid heuristic strategies incurs a cost of $O(N \cdot P \cdot M)$. Within the main evolutionary loop, each generation requires evaluating all individuals to compute makespan and energy consumption, which costs $O(N \cdot P)$. Subsequently, for each individual, the Q-learning operations, including state extraction, action selection, application of a low-level heuristic [bounded by $O(P)$ in the worst case], reward calculation, and Q-table update, collectively contribute $O(N \cdot P)$ per generation. Following offspring generation, environmental selection merges the parent and offspring populations (size $2N$) and performs fast non-dominated sorting, which dominates with a complexity of $O(N^2)$ per generation. Summing the dominant terms across all generations, the overall time complexity is $O(T \cdot (N \cdot P+N^2)$.

4. Experimental results

4.1. Datasets and test environments

To comprehensively evaluate the performance of the proposed QL-HH, a diverse set of benchmark instances is adopted from the literature. The test suite consists of three classical datasets widely used for the FJSP and DFJSP, namely those proposed by Brandimarte [56] (MK01-MK08), Dauzère-Pérès and Paulli [54] (DP01-DP18), and Hurink et al. [57] (*vdata* and *rdata*). These instances cover various problem scales in terms of number of jobs, operations per job, and machine flexibility. Following the methodology of Xiong et al. [58], the original single factory instances (especially those from Hurink et al.) are extended to a distributed setting by assuming that all factories are identical and each factory contains the same set of machines as the original instance. Two flexibility levels are generated:

- **High-flexibility dataset**: For each operation, the set of eligible machines contains approximately half of the total machines in a factory. Specifically, the flexibility size is set to $\lfloor M/2 \rfloor$, where M is the number of machines per factory.
- **Low-flexibility dataset**: For each operation, exactly two machines are eligible, which is the minimum flexibility setting that still maintains the flexible nature of the problem.

The number of jobs J is selected from {10, 15, 20, 30, 40, 50, 60, 75, 80, 100}, the number of machines per factory M from {5, 10}, and the number of factories F from {2, 3, 4}. To keep computational complexity manageable, only combinations satisfying a machine load ratio $\rho = J/(M \times F) \leq 3$ are retained. This threshold refers to the research results of Xiong et al. [56], aiming to keep the computational workload reasonable and prevent excessive load on the equipment. This yields 23 valid (J, M, F) combinations, and five instances are generated for each combination, resulting in 115 instances for each flexibility level. All algorithms are implemented in Python. The experiments are conducted on a personal computer equipped with an Intel(R) Core (TM) i7-8550U processor (1.80 GHz) and 8.0 GB of RAM, running the Windows 10 operating system. For each benchmark instance, every algorithm is executed independently 10 times to ensure statistical reliability. The maximum number of iterations per run is set to 200. The average values of the performance metrics over the 10 runs are reported for comparison.

4.2. Evaluation metrics

Two widely adopted performance metrics are employed to assess the quality of the Pareto front approximations obtained by the compared algorithms.

(1) Hypervolume (HV) [59]

The hypervolume metric quantifies the volume of the objective space dominated by a set of non-dominated solutions with respect to a reference point. Let PF denote the set of Pareto front solutions obtained by an algorithm, and let $r = (r_1, r_2, \ldots, r_k)$ be a reference point that is dominated by all solutions in PF. The HV is defined as:

$$HV(PF) = \bigcup_{x \in PF} \prod_{i=1}^{m} | x_i - r_i | \tag{4.1}$$

where m is the number of objectives, and r_i is the reference point for the ith objective. A larger HV value indicates better overall performance.

(2) Inverted generational distance (IGD) [60]

The inverted generational distance measures both the convergence and coverage of an approximated Pareto front PF relative to a reference set PF^*. Since the true Pareto front of the problem is unknown, PF^* is constructed by collecting all non-dominated solutions obtained by all compared algorithms across multiple runs and then extracting the final Pareto front from this union. This reference set serves as an approximation of the true Pareto front. The IGD is then computed as the average Euclidean distance from each point in PF^* to its nearest neighbor in PF:

$$IGD(PF, PF^*) = \frac{1}{|PF^*|} \sum_{z \in PF} \min_{x \in PF} \| z - x \|_2 \tag{4.2}$$

where $\| \|_2$ denotes the Euclidean distance in the objective space. A smaller IGD value indicates that PF is closer to the reference front and covers it more uniformly.

4.3. Parameter setting

Three parameters play crucial roles in the proposed QL-HH: the population size (PS), the learning rate (α), and the discount factor (γ). To determine their appropriate values, we carry out an orthogonal experimental design. Each parameter is tested at four levels, as summarized in Table 5. An L_{16} (4^3) orthogonal array is adopted, leading to 16 distinct combinations (see Table **6**). The experiments are performed on three representative instances with different numbers of factories (f = 2, 3, 4). For each parameter combination, the algorithm runs five independent times per instance. The average HV and IGD values across the three instances are recorded.

Figure 5 shows how each parameter influences HV and IGD. As the population size (PS) increases from 40 to 100, HV exhibits a continuous upward trend, while IGD decreases steadily, indicating that a larger population size consistently improves both convergence and diversity. Regarding the learning rate (α), the best performance is achieved at α = 0.3, where HV reaches its maximum and IGD its minimum. For the discount factor (γ), the optimal value is γ = 0.8, yielding the best balance between immediate and future rewards. Based on the above analysis, the optimal parameter combination is determined as PS = 100, α = 0.3, γ = 0.8.

Table 5. Four levels of parameters.

Parameters	Level			
	1	2	3	4
PS	40	60	80	100
α	0.1	0.3	0.5	0.7
γ	0.6	0.7	0.8	0.9

Table 6. Orthogonal table of parameter combinations.

Combination	Factor			Average	
	PS	α	γ	*HV*	*IGD*
1	1	1	1	0.6930	1.1267e-1
2	1	2	2	0.7143	1.0925e-1
3	1	3	3	0.7191	1.1168e-1
4	1	4	4	0.7019	1.1993e-1
5	2	1	2	0.7144	8.4417e-2
6	2	2	1	0.7279	8.1317e-2
7	2	3	4	0.7278	7.9767e-2
8	2	4	3	0.7331	7.9367e-2
9	3	1	3	0.7290	6.1717e-2
10	3	2	4	0.7319	6.5233e-2
11	3	3	1	0.7167	6.8150e-2
12	3	4	2	0.7368	6.4683e-2
13	4	1	4	0.7382	5.1450e-2
14	4	2	3	0.7486	5.0217e-2
15	4	3	2	0.7254	5.4533e-2
16	4	4	1	0.7264	5.4033e-2

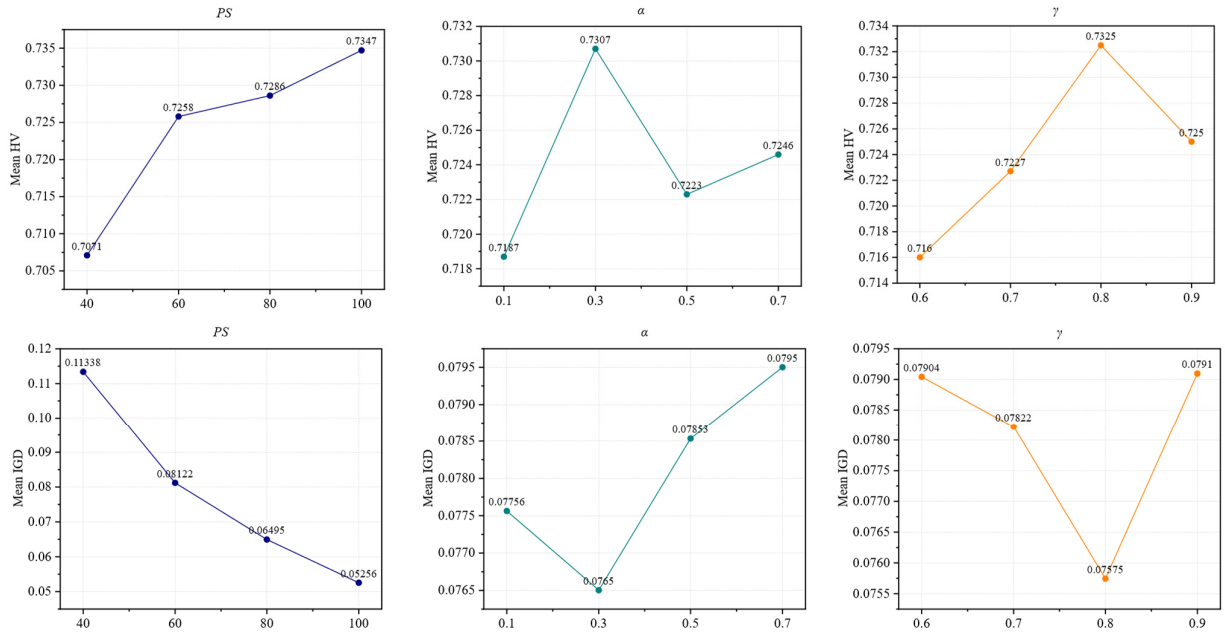


Figure 5. Variation trend of each level parameter on HV and IGD.

Tables 7 and 8 present the ANOVA results for HV and IGD, respectively. For HV, the population size is the most influential factor ($p < 0.05$); the effects of the learning rate and discount factor are also present, though not as pronounced as that of *PS*. For IGD, *PS* is highly significant ($p < 0.001$), while the influence of α and γ is considerably smaller. These results confirm that population size plays a

dominant role, whereas the learning rate and discount factor have comparatively weaker effects. A similar pattern is observed for both performance metrics.

Table 7. ANOVA of parameters on HV.

Source	Sum Sq.	d. f.	Mean Sq.	F	*p-value*
PS	0.001503	3	0.000501	5.2827	0.040302
α	0.000123	3	0.000041	0.4135	0.738179
γ	0.000298	3	0.000099	1.0493	0.436727
Error	0.000569	6	0.000095		
Total	0.002494	15			

Table 8. ANOVA of parameters on IGD.

Source	Sum Sq.	d. f.	Mean Sq.	F	*p-value*
PS	0.010924	3	0.003641	189.1998	0.000002
α	0.000041	3	0.000014	0.7185	0.639494
γ	0.000044	3	0.000015	0.7684	0.183954
Error	0.000115	6	0.000019		
Total	0.011126	15			

4.4. Effectiveness of initialization

To validate the contribution of the proposed hybrid initialization scheme described in Section 3.3, a variant of the algorithm, denoted as QL-HH_ni, is constructed for comparison. In this variant, the heuristic strategies are removed, and each individual in the initial population is generated purely at random across all four encoding dimensions (factory assignment, operation sequence, machine allocation, and speed level). All other components (Q-learning mechanism, low-level heuristics) remain identical to those in the standard QL-HH.

Table 9 and Table 10 summarize the results. Across all tested cases, QL-HH consistently outperforms QL-HH_ni, achieving higher HV values and lower IGD values. The improvement is more pronounced when the number of factories increases, indicating that the hybrid initialization becomes increasingly beneficial for larger solution spaces. In summary, the hybrid heuristic initialization provides a well-structured and diverse starting population, which effectively guides the subsequent Q-learning search and accelerates convergence. The removal of this strategy leads to a noticeable degradation in both solution quality and dominance, thereby validating its necessity in the proposed algorithm. Figures 6 and 7 present the box plots of HV and IGD, respectively, for QL-HH and QL-HH_ni. It can be observed that QL-HH exhibits not only higher median HV values but also a more concentrated distribution, indicating both superior solution quality and robustness. In terms of IGD, QL-HH achieves consistently lower values with less variance, further confirming the effectiveness of the hybrid initialization strategy.

Table 9. Comparison of QL-HH and its variants on the high-flexibility dataset.

Instance	Scale	*HV*		*IGD*	
		QL-HH	QL-HH_ni	QL-HH	QL-HH_ni
HF01	10×5 - 2	0.7662	0.7120	3.5320e-3	1.8986e-1
HF02	15×5 - 2	0.7508	0.7018	7.1900e-4	3.0295e-1
HF03	20×5 - 2	0.7127	0.7116	8.9000e-5	3.0521e-1
HF04	10×10 - 2	0.7885	0.7105	5.1700e-4	2.2986e-1
HF05	20×10 - 2	0.7450	0.6951	0.0000e+0	4.0509e-1
HF06	30×10 - 2	0.7192	0.6667	0.0000e+0	5.2450e-1
HF07	40×10 - 2	0.7384	0.6893	0.0000e+0	5.4114e-1
HF08	50×10 - 2	0.7716	0.7188	0.0000e+0	5.5343e-1
HF09	60×10 - 2	0.7343	0.6813	0.0000e+0	5.6127e-1
HF10	15×5 - 3	0.7602	0.6717	1.8400e-4	2.8705e-1
HF11	20×5- 3	0.8069	0.5943	1.8700e-4	3.5357e-1
HF12	30×5- 3	0.7181	0.6755	0.0000e+0	4.0453e-1
HF13	30×10 - 3	0.7354	0.6508	0.0000e+0	4.4882e-1
HF14	45×10 - 3	0.7840	0.6553	0.0000e+0	5.4996e-1
HF15	60×10 - 3	0.7381	0.6753	0.0000e+0	5.6524e-1
HF16	75×10 - 3	0.6980	0.6037	0.0000e+0	5.4523e-1
HF17	20×5 - 4	0.7102	0.6813	4.1610e-3	2.7652e-1
HF18	40×5- 4	0.7216	0.6592	0.0000e+0	4.3919e-1
HF19	20×10- 4	0.7161	0.6741	5.1000e-5	3.0941e-1
HF20	40×10- 4	0.7425	0.6767	0.0000e+0	5.5055e-1
HF21	60×10- 4	0.7404	0.6453	0.0000e+0	6.0187e-1
HF22	80×10- 4	0.7857	0.6370	0.0000e+0	6.2149e-1
HF23	100×10- 4	0.7082	0.6252	0.0000e+0	6.1876e-1

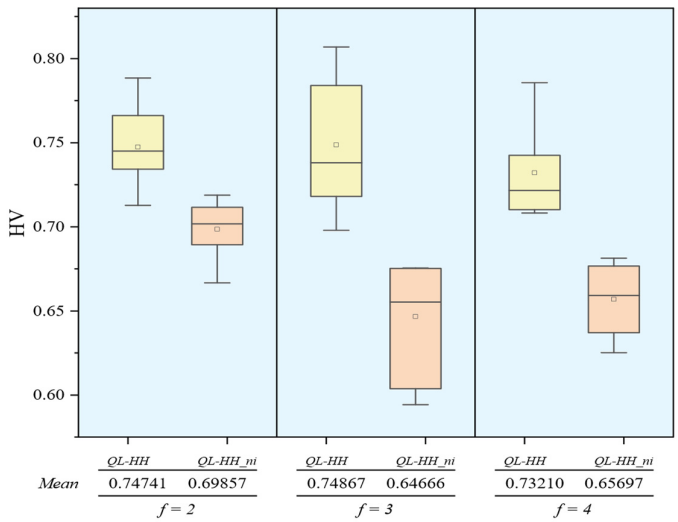


Figure 6. Box plots of HV comparison between QL-HH and its variants.

Table 10. Comparison of QL-HH and its variants on the low-flexibility dataset.

Instance	Scale	*HV*		*IGD*	
		QL-HH	QL-HH_ni	QL-HH	QL-HH_ni
LF01	10×5 - 2	0.7954	0.7404	1.5260e-3	2.0343e-1
LF02	15×5 - 2	0.7551	0.7238	3.6800e-4	3.2578e-1
LF03	20×5 - 2	0.7767	0.7072	0.0000e+0	3.1272e-1
LF04	10×10 - 2	0.8021	0.7480	8.7402e-4	2.7531e-1
LF05	20×10 - 2	0.7598	0.7226	0.0000e+0	4.4177e-1
LF06	30×10 - 2	0.7416	0.7097	0.0000e+0	4.5493e-1
LF07	40×10 - 2	0.7276	0.6949	0.0000e+0	5.5085e-1
LF08	50×10 - 2	0.7324	0.7133	0.0000e+0	5.4892e-1
LF09	60×10 - 2	0.7051	0.6739	0.0000e+0	5.8348e-1
LF10	15×5 - 3	0.7637	0.7092	6.3037e-4	2.7936e-1
LF11	20×5- 3	0.7207	0.6395	0.0000e+0	3.5448e-1
LF12	30×5- 3	0.7370	0.7090	0.0000e+0	4.1674e-1
LF13	30×10 - 3	0.7629	0.6547	0.0000e+0	4.6360e-1
LF14	45×10 - 3	0.7121	0.6761	0.0000e+0	5.4428e-1
LF15	60×10 - 3	0.7309	0.6421	0.0000e+0	5.7313e-1
LF16	75×10 - 3	0.7179	0.6660	0.0000e+0	5.8730e-1
LF17	20×5 - 4	0.7156	0.6179	5.2903e-4	3.1993e-1
LF18	40×5- 4	0.7456	0.6380	0.0000e+0	4.8021e-1
LF19	20×10- 4	0.7730	0.7095	2.4615e-4	3.3867e-1
LF20	40×10- 4	0.7228	0.6248	0.0000e+0	5.6378e-1
LF21	60×10- 4	0.7127	0.6467	0.0000e+0	6.0795e-1
LF22	80×10- 4	0.7002	0.6011	0.0000e+0	5.9930e-1
LF23	100×10- 4	0.6847	0.5994	0.0000e+0	6.4670e-1

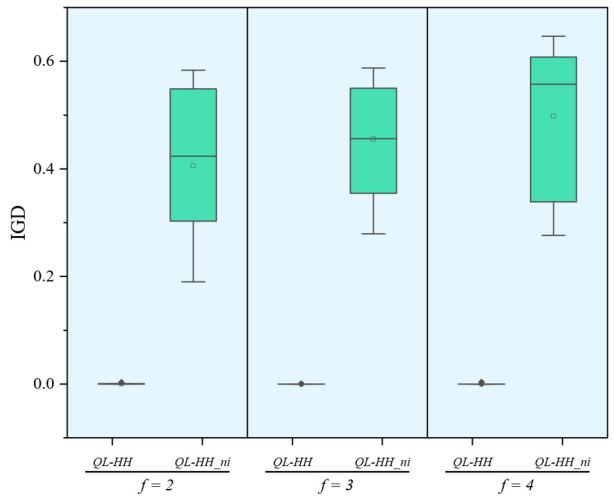


Figure 7. Box plots of IGD comparison between QL-HH and its variants.

4.5. Evaluating each core component of the proposed QL-HH

To identify the contribution of each core component of QL-HH, we conduct a systematic ablation study focusing on four key components: Q-learning mechanism, reward function design, dynamic ε-greedy strategy, and critical path-based speed adjustment operators (LLH7 and LLH8). Four variants of QL-HH are constructed, each disabling one specific component while keeping all other settings identical to the full QL-HH, as shown in Table 11.

Table 11. QL-HH and its variants.

Variant	Removed component	Purpose
QL-HH_nQL	Q-learning (replace with random operator selection)	Test Q-learning mechanism
QL-HH_nRd	Crowding distance term in reward (only Pareto dominance)	Test reward design
QL-HH_nEps	Dynamic ε-greedy (use fixed $\varepsilon = 0.5$)	Test adaptive exploration
QL-HH_nCP	Critical-path speed adjustment (LLH7 and LLH8 removed)	Test energy-saving operators

All variants are evaluated on 10 representative instances (five from the high-flexibility dataset: HF01, HF06, HF11, HF16, HF21; and five from the low-flexibility dataset: LF01, LF06, LF11, LF16, LF21) covering different factory numbers (2, 3, 4) and problem scales. Each algorithm runs 10 independent times per instance under the same parameter configuration ($PS = 100$, $\alpha = 0.3$, $\gamma = 0.8$, $T_{\max} = 200$). The average HV and IGD values over the 10 runs are reported. To measure the impact of each component, we calculate the relative degradation in HV and IGD compared to the full QL-HH:

$$\Delta HV = \frac{HV_{full} - HV_{varient}}{HV_{full}} \times 100\%, \Delta IGD = \frac{IGD_{varient} - IGD_{full}}{IGD_{full}} \times 100\% \tag{4.3}$$

Table 12. Comparison of HV of QL-HH and its variants on partial test instances.

Instances	Scale	QL-HH and its variants				
		QL-HH	*QL-HH_nQL*	*QL-HH_nRd*	*QL-HH_nEps*	*QL-HH_nCP*
HF01	10×5 - 2	0.7662	0.6716	0.7447	0.7401	0.6998
HF06	30×10 - 2	0.7192	0.6331	0.6990	0.6947	0.6568
HF11	20×5 - 3	0.8069	0.7168	0.7884	0.7795	0.7371
HF16	75×10 - 3	0.6980	0.6232	0.6785	0.6743	0.6376
HF21	60×10 - 4	0.7401	0.6485	0.7194	0.7150	0.6761
LF01	10×5 - 2	0.7954	0.6973	0.7732	0.7684	0.7266
LF06	30×10 - 2	0.7416	0.6580	0.7208	0.7163	0.6774
LF11	20×5 - 3	0.7207	0.6359	0.7005	0.6962	0.6583
LF16	75×10 - 3	0.7179	0.6218	0.6978	0.6935	0.6558
LF21	60×10 - 4	0.7127	0.6204	0.6927	0.6885	0.6510
Average	-	0.74187	0.65266	0.72150	0.71665	0.67765

Table 13. Comparison of HV of QL-HH and its variants on partial test instances.

Instances	Scale	QL-HH and its variants				
		QL-HH	*QL-HH_nQL*	*QL-HH_nRd*	*QL-HH_nEps*	*QL-HH_nCP*
HF01	10×5 - 2	1.3753e-1	1.7742e-1	1.4716e-1	1.5134e-1	1.9537e-1
HF06	30×10 - 2	1.3657e-1	1.7471e-1	1.4470e-1	1.4785e-1	1.9106e-1
HF11	20×5 - 3	1.1125e-1	1.4549e-1	1.2021e-1	1.2356e-1	1.7035e-1
HF16	75×10 - 3	1.5636e-1	2.0412e-1	1.6805e-1	1.7273e-1	2.2457e-1
HF21	60×10 - 4	1.5471e-1	1.9535e-1	1.6334e-1	1.6760e-1	2.1679e-1
LF01	10×5 - 2	1.1043e-1	1.4533e-1	1.2034e-1	1.2368e-1	1.7013e-1
LF06	30×10 - 2	1.4359e-1	1.8120e-1	1.5095e-1	1.5723e-1	1.9672e-1
LF11	20×5 - 3	1.5867e-1	2.0478e-1	1.6987e-1	1.7627e-1	2.2348e-1
LF16	75×10 - 3	1.2470e-1	1.6316e-1	1.3228e-1	1.3725e-1	1.8063e-1
LF21	60×10 - 4	1.6602e-1	2.2049e-1	1.8481e-1	1.9186e-1	2.4282e-1
Average	-	1.39983e-1	1.81205e-1	1.50171e-01	1.54937e-01	2.01192e-01

Tables 12 and 13 present the detailed HV and IGD results for all 10 instances, respectively. Figure 8 shows the average HV and IGD of the full QL-HH and its four variants. The bar chart clearly shows that the full QL-HH achieves the highest HV (0.74187) and the lowest IGD (0.13998). Removing Q-learning (*QL-HH_nQL*) causes the largest performance degradation, while removing critical-path speed adjustment (*QL-HH_nCP*) also leads to a substantial increase in IGD, indicating poor Pareto front quality.

Based on the data from Tables 12 and 13, we compute the relative degradation in HV (ΔHV) and IGD (ΔIGD) for each variant compared to the full QL-HH. The results are summarized in Table 14, which also ranks the impact of each component. The Q-learning mechanism is thus identified as the most critical contributor (rank 1). The critical-path speed adjustment operators rank second, with an extremely high IGD degradation (43.73%), confirming their essential role in maintaining solution quality. The dynamic ε-greedy strategy and the crowding-distance term in the reward also provide positive but secondary contributions (ranks 3 and 4).

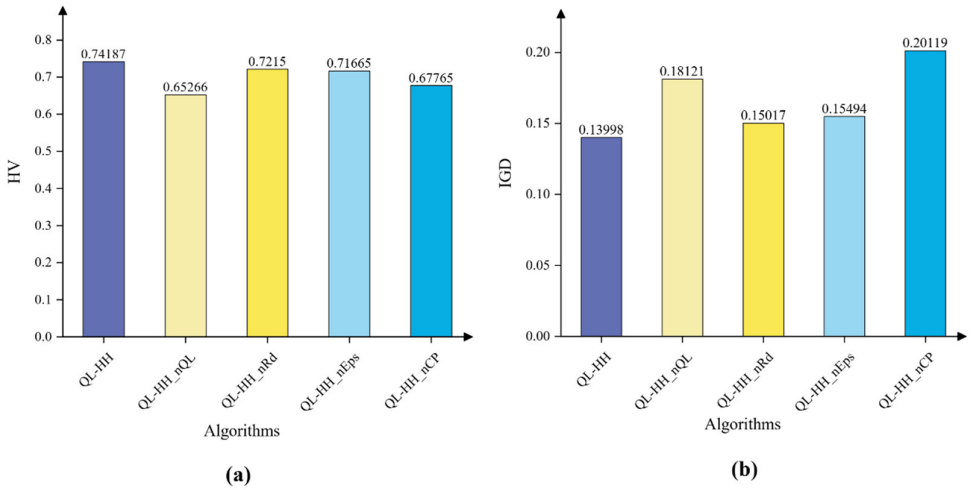


Figure 8. Average HV and IGD values of full QL-HH and its four variants.

Table 14. Performance degradation and impact ranking of each variant relative to full QL-HH.

Variant	ΔHV (↓)	ΔIGD (↑)	Rank of impact
QL-HH_nQL	12.03%	29.45%	1
QL-HH_nRd	2.75%	7.28%	4
QL-HH_nEps	3.40%	10.69%	3
QL-HH_nCP	8.66%	43.73%	2

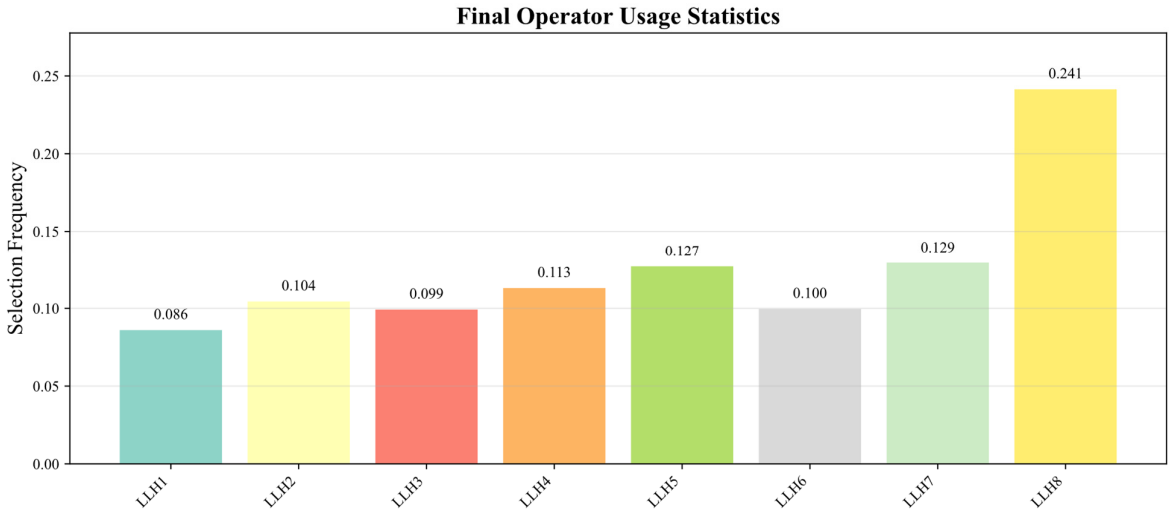


Figure 9. Selection frequency of each low-level heuristic (LLH1-LLH8) on instance HF06.

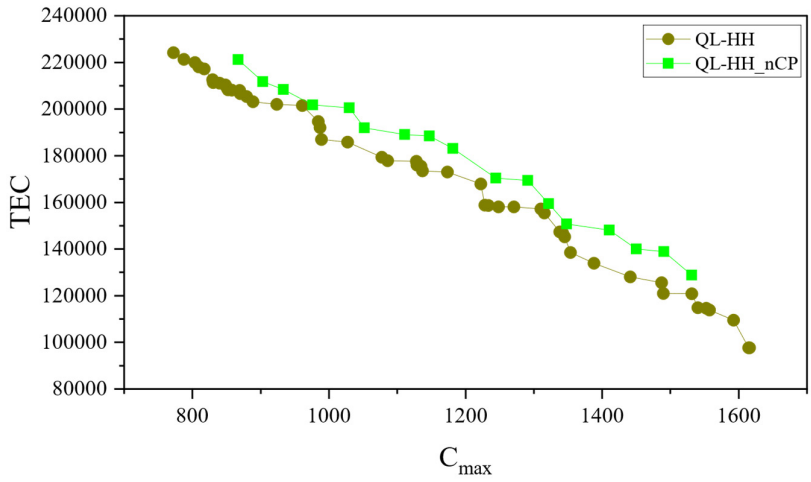


Figure 10. Pareto fronts obtained by full QL-HH and QL-HH_nCP on instance HF06.

Figure 9 presents the operator selection frequency of the Q-learning agent on a representative instance (HF06). Notably, LLH8 (hybrid speed adjustment) is selected most frequently (0.241), followed by LLH7 (non-critical speed reduction, 0.129). Both frequencies are significantly higher than

those of the other six LLHs (0.086–0.127). This indicates that the agent learns to favor operators that directly balance makespan and energy consumption, validating the effectiveness of the reward function and dynamic ε-greedy strategy.

Figure 10 compares the Pareto solution distributions of full QL-HH and *QL-HH_nCP* on instance HF06 in the makespan dimension. The full QL-HH produces solutions across a wide $C_{\max}$ range (780-1600) with good density, whereas *QL-HH_nCP* concentrates its solutions in higher makespan regions with a sparse distribution. This demonstrates that, although *QL-HH_nCP* maintains a moderate HV, its Pareto set is significantly worse in both convergence and diversity, highlighting the indispensable role of critical-path speed adjustment in achieving a high-quality Pareto front.

In summary, the ablation study confirms that every core component of QL-HH contributes positively, with the Q-learning mechanism and critical-path speed adjustment operators being the two most influential factors.

4.6. Comparison between QL-HH and its peers

To further validate the effectiveness of the proposed QL-HH, we compare it against four state-of-the-art multi-objective evolutionary algorithms: QMOEA/D-AWA [41], MOEA/D [37], NSGA-II [61], and KBEA [46]. The parameter configurations for each algorithm follow the recommendations in the literature, and for QL-HH, the settings determined in Section 4.3 are used.

Table 15–Table 19 list the average HV and IGD values for all instances, grouped by the number of factories and flexibility level. The best result in each row is highlighted in bold. According to the experimental results, QL-HH achieves the optimal performance in the vast majority of test instances, and is only slightly inferior to the comparison algorithms in a few individual cases. This advantage is particularly pronounced in high-flexibility, large-scale instances, where the adaptive Q-learning-based low-level heuristic selection strategy can effectively explore the expanded solution space. Figure 11 and Figure 12 present boxplots comparing QL-HH with five benchmark algorithms in terms of HV and IGD, respectively. It can be clearly observed that the distribution of HV values obtained by QL-HH is generally higher than that of the other algorithms, while its IGD values are significantly lower, indicating that the obtained Pareto front exhibits better convergence and diversity.

To assess the statistical significance of the observed differences, the Wilcoxon signed-rank test is applied pairwise between QL-HH and each competitor, based on the average HV values obtained from 10 independent runs per instance. The null hypothesis that the two algorithms perform equally is tested at a significance level of $\alpha = 0.05$. The results are summarized in Table 18. All comparisons yield $p < 0.05$, indicating that the superiority of QL-HH is statistically significant. Moreover, the consistently high R^+ values (sum of ranks where QL-HH outperforms the competitor) and near-zero R^- values further confirm that QL-HH outperforms the four state-of-the-art algorithms on almost all instances.

Table 15. HV comparison between QL-HH and comparison algorithms on the high-flexibility dataset.

Instances	Scale	Algorithms				
		QL-HH	QMOEA/D-AWA	MOEA/D	NSGA-II	KBEA
HF01	10×5 - 2	0.7662	0.7228	0.6834	0.6912	0.7427
HF02	15×5 - 2	0.7508	0.7395	0.6589	0.6643	0.7281
HF03	20×5 - 2	0.7127	0.6882	0.6415	0.6372	0.6956
HF04	10×10 - 2	0.7885	0.7383	0.6127	0.6235	0.7319
HF05	20×10 - 2	0.7450	0.7179	0.6742	0.6816	0.7253
HF06	30×10 - 2	0.7192	0.6899	0.6638	0.6571	0.7084
HF07	40×10 - 2	0.7384	0.7156	0.6329	0.6447	0.7012
HF08	50×10 - 2	0.7716	0.7099	0.6925	0.7048	0.7493
HF09	60×10 - 2	0.7343	0.6947	0.6213	0.6356	0.6938
HF10	15×5 - 3	0.7602	0.6744	0.6128	0.6074	0.7051
HF11	20×5- 3	0.8069	0.7012	0.6775	0.6839	0.7427
HF12	30×5- 3	0.7181	0.6774	0.6775	0.6839	0.7116
HF13	30×10 - 3	0.7354	0.6851	0.6521	0.6645	0.7123
HF14	45×10 - 3	0.7840	0.7101	0.6468	0.6532	0.7347
HF15	60×10 - 3	0.7381	0.6996	0.5843	0.5926	0.6825
HF16	75×10 - 3	0.6980	0.6308	0.6115	0.6039	0.6541
HF17	20×5 - 4	0.7102	0.6303	0.5427	0.5513	0.6539
HF18	40×5- 4	0.7216	0.6981	0.6045	0.6128	0.6826
HF19	20×10- 4	0.7161	0.7245	0.6324	0.6417	0.6932
HF20	40×10- 4	0.7425	0.6841	0.6239	0.6352	0.7048
HF21	60×10- 4	0.7404	0.6857	0.5628	0.5715	0.6723
HF22	80×10- 4	0.7857	0.7210	0.6032	0.5946	0.7159
HF23	100×10- 4	0.7082	0.6682	0.5917	0.6023	0.6738

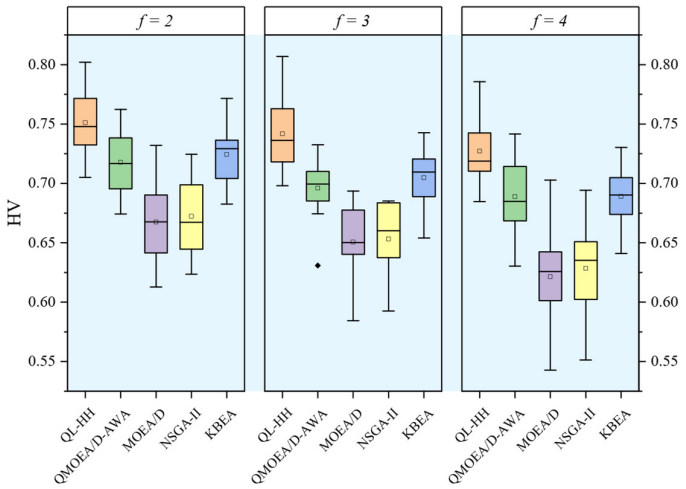


Figure 11. Box plots of HV comparison between QL-HH and comparison algorithms.

Table 16. IGD comparison between QL-HH and comparison algorithms on the high-flexibility dataset.

Instances	Scale	Algorithms				
		QL-HH-MOEA	QMOEA/D-AWA	MOEA/D	NSGA-II	KBEA
HF01	10×5 - 2	1.3753e-1	2.0969e-1	3.6647e-1	2.1067e-1	2.0521e-1
HF02	15×5 - 2	1.2026e-1	2.0381e-1	4.6499e-1	2.9980e-1	2.1143e-1
HF03	20×5 - 2	1.5634e-1	1.8973e-1	4.0923e-1	3.1344e-1	1.9356e-1
HF04	10×10 - 2	1.4017e-1	2.5580e-1	4.9419e-1	2.9684e-1	2.4892e-1
HF05	20×10 - 2	1.0875e-1	2.1372e-1	5.7705e-1	4.2163e-1	2.3018e-1
HF06	30×10 - 2	1.3653e-1	2.0819e-1	5.9559e-1	4.3120e-1	2.1457e-1
HF07	40×10 - 2	1.0136e-1	2.1920e-1	6.5573e-1	4.9893e-1	2.2684e-1
HF08	50×10 - 2	1.5429e-1	1.9641e-1	6.3060e-1	5.2644e-1	2.0055e-1
HF09	60×10 - 2	1.2315e-1	2.0552e-1	6.1592e-1	5.2959e-1	2.1137e-1
HF10	15×5 - 3	1.4762e-1	2.0340e-1	3.0340e-1	2.6950e-1	2.0013e-1
HF11	20×5 - 3	1.1125e-1	2.0681e-1	3.0681e-1	3.0704e-1	2.0458e-1
HF12	30×5 - 3	1.0147e-1	2.1533e-1	3.3672e-1	4.1116e-1	2.0458e-1
HF13	30×10 - 3	1.3039e-1	2.0494e-1	2.0494e-1	4.3976e-1	2.0271e-1
HF14	45×10 - 3	1.9285e-1	1.7738e-1	2.0738e-1	4.8522e-1	2.0642e-1
HF15	60×10 - 3	1.6426e-1	1.9751e-1	3.9751e-1	5.5146e-1	1.9533e-1
HF16	75×10 - 3	1.5697e-1	2.0032e-1	3.0032e-1	5.0209e-1	1.9875e-1
HF17	20×5 - 4	1.0874e-1	2.3227e-1	4.5837e-1	3.1864e-1	2.1945e-1
HF18	40×5 - 4	1.6012e-1	2.4345e-1	5.3277e-1	3.7318e-1	2.1589e-1
HF19	20×10 - 4	1.5319e-1	2.1621e-1	5.6832e-1	3.7515e-1	2.3236e-1
HF20	40×10 - 4	1.4468e-1	2.0761e-1	6.2436e-1	4.6027e-1	2.2515e-1
HF21	60×10 - 4	1.5382e-1	2.8635e-1	6.3614e-1	5.3292e-1	2.6974e-1
HF22	80×10 - 4	2.1675e-1	3.7867e-1	6.1791e-1	5.3230e-1	3.8102e-1
HF23	100×10 - 4	2.6003e-1	3.7145e-1	6.0622e-1	5.1989e-1	3.7358e-1

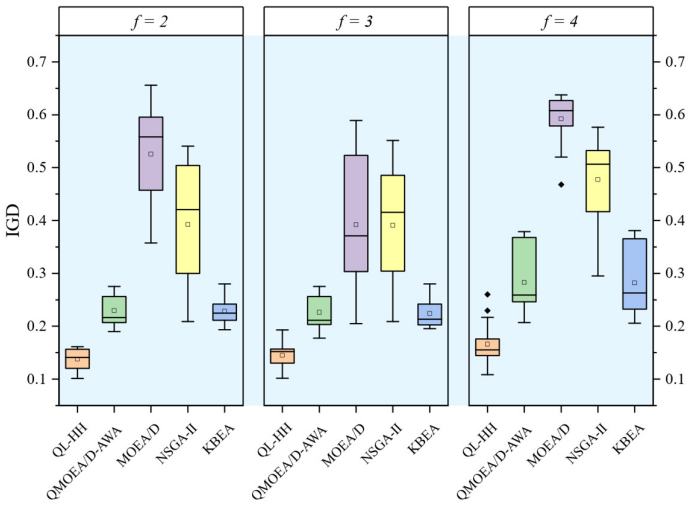


Figure 12. Box plots of IGD comparison between QL-HH and comparison algorithms.

Table 17. HV comparison between QL-HH and comparison algorithms on the high-flexibility dataset.

Instances	Scale	Algorithms				
		QL-HH-MOEA	QMOEA/D-AWA	MOEA/D	NSGA-II	KBEA
LF01	10×5 - 2	0.7954	0.7623	0.7034	0.7156	0.7715
LF02	15×5 - 2	0.7551	0.7218	0.6845	0.6723	0.7302
LF03	20×5 - 2	0.7767	0.7456	0.7012	0.7134	0.7321
LF04	10×10 - 2	0.8021	0.7613	0.7321	0.7245	0.7535
LF05	20×10 - 2	0.7598	0.7289	0.6902	0.6987	0.7364
LF06	30×10 - 2	0.7416	0.7123	0.6715	0.6638	0.7041
LF07	40×10 - 2	0.7276	0.6954	0.6547	0.6612	0.7332
LF08	50×10 - 2	0.7324	0.7018	0.6623	0.6701	0.7195
LF09	60×10 - 2	0.7051	0.6742	0.6315	0.6428	0.6826
LF10	15×5 - 3	0.7637	0.7325	0.6936	0.6852	0.7213
LF11	20×5 - 3	0.7207	0.6891	0.6482	0.6556	0.6974
LF12	30×5 - 3	0.7370	0.7056	0.6673	0.6745	0.7139
LF13	30×10 - 3	0.7629	0.7318	0.6921	0.6837	0.7205
LF14	45×10 - 3	0.7121	0.7204	0.6402	0.6515	0.6889
LF15	60×10 - 3	0.7309	0.6993	0.6584	0.6669	0.7076
LF16	75×10 - 3	0.7179	0.6862	0.6451	0.6374	0.6748
LF17	20×5 - 4	0.7156	0.6835	0.6423	0.6508	0.6920
LF18	40×5- 4	0.7456	0.7142	0.6745	0.6821	0.7227
LF19	20×10 - 4	0.7730	0.7416	0.7028	0.6942	0.7303
LF20	40×10 - 4	0.7228	0.6915	0.6507	0.6593	0.6992
LF21	60×10 - 4	0.7127	0.6801	0.6395	0.6472	0.6886
LF22	80×10 - 4	0.7002	0.6685	0.6278	0.6354	0.6763
LF23	100×10 - 4	0.6847	0.6524	0.6012	0.6205	0.6409

Table 18. Wilcoxon test of QL-HH and comparison algorithms on HV and IGD.

QL-HH vs	Metric	R^+	R^-	Z	*p-value*	$\alpha = 0.05$
QMOEA/D-AWA	*HV*	1078	3	–5.872388	1.421085e-13	YES
	IGD	1080	1	–5.894239	5.684342e-14	YES
MOEA/D	*HV*	1081	0	–5.905164	2.842171e-14	YES
	IGD	1081	0	–5.905164	2.842171e-14	YES
NSGA-II	*HV*	1081	0	–5.905164	2.842171e-14	YES
	IGD	1081	0	–5.905164	2.842171e-14	YES
KBEA	*HV*	1080	1	–5.894239	5.684342e-14	YES
	IGD	1081	0	–5.905164	2.842171e-14	YES

Table 19. IGD comparison between QL-HH and comparison algorithms on the high-flexibility dataset.

Instances	Scale	Algorithms				
		QL-HH	QMOEA/D-AWA	MOEA/D	NSGA-II	KBEA
LF01	10×5 - 2	1.1043e-1	2.3086e-1	3.5732e-1	2.0899e-1	2.1975e-1
LF02	15×5 - 2	1.4156e-1	2.5448e-1	3.8506e-1	2.8847e-1	2.8021e-1
LF03	20×5 - 2	1.5638e-1	2.6975e-1	4.5711e-1	3.1772e-1	2.6489e-1
LF04	10×10 - 2	1.6124e-1	2.7510e-1	5.2315e-1	3.0442e-1	2.3036e-1
LF05	20×10 - 2	1.5701e-1	2.7049e-1	5.8895e-1	4.1992e-1	2.6612e-1
LF06	30×10 - 2	1.4372e-1	2.5637e-1	5.3923e-1	4.2779e-1	2.4189e-1
LF07	40×10 - 2	1.5655e-1	2.0690e-1	5.7998e-1	5.4048e-1	2.2314e-1
LF08	50×10 - 2	1.1662e-1	2.0695e-1	5.7990e-1	5.0404e-1	2.3437e-1
LF09	60×10 - 2	1.5700e-1	2.5737e-1	6.3740e-1	5.2666e-1	2.0589e-1
LF10	15×5 - 3	1.4716e-1	2.6012e-1	3.9450e-1	2.6141e-1	2.5583e-1
LF11	20×5- 3	1.5867e-1	2.7228e-1	5.1132e-1	3.4946e-1	2.5794e-1
LF12	30×5- 3	1.2939e-1	2.0988e-1	5.1057e-1	3.6448e-1	2.1735e-1
LF13	30×10 - 3	1.1587e-1	2.2723e-1	5.9973e-1	4.4711e-1	2.5351e-1
LF14	45×10 - 3	1.3945e-1	2.6995e-1	5.9022e-1	4.8296e-1	2.0678e-1
LF15	60×10 - 3	1.4400e-1	2.5421e-1	6.0015e-1	5.0733e-1	2.0112e-1
LF16	75×10 - 3	1.2473e-1	1.9445e-1	6.3548e-1	5.3198e-1	2.1126e-1
LF17	20×5 - 4	1.4755e-1	2.6058e-1	4.6801e-1	2.9531e-1	2.5631e-1
LF18	40×5- 4	1.0855e-1	2.4953e-1	5.2010e-1	4.1673e-1	2.1589e-1
LF19	20×10- 4	1.2581e-1	2.7769e-1	5.7892e-1	3.8816e-1	2.3412e-1
LF20	40×10- 4	1.6647e-1	2.4628e-1	5.9003e-1	5.0820e-1	2.9305e-1
LF21	60×10- 4	1.7601e-1	2.5527e-1	6.2710e-1	5.4341e-1	2.8226e-1
LF22	80×10- 4	1.7005e-1	3.7900e-1	6.2688e-1	5.0478e-1	3.7683e-1
LF23	100×10- 4	2.2948e-1	3.6787e-1	6.0911e-1	5.7643e-1	3.6532e-1

Furthermore, the Friedman test is employed to rank all algorithms comprehensively across all instances. As shown in Table 20 and Figure 13, QL-HH achieves the highest rank in both HV and IGD metrics, with a combined ranking of 1.04348, far ahead of the second-ranked KBEA (2.41304). The corresponding p-values are well below 0.05, demonstrating that the performance advantage of QL-HH is not due to chance.

Figure 14 shows the Pareto front plots obtained by each algorithm on different instances. It can be seen that the non-dominated solutions obtained by QL-HH are more uniformly distributed and lie closer to the approximated true front. Figure 15 and 16 present the Gantt charts of the schedules obtained for the high-flexibility instance HF03 ($f = 2$) and the low-flexibility instance LF10 ($f = 3$), respectively. These charts intuitively illustrate the rationality and energy-saving effectiveness of the schedules generated by QL-HH in terms of machine assignment, job sequencing, and speed selection.

Table 20. Friedman ranking of QL-HH and comparison algorithms.

Algorithms	*HV*	*IGD*	Combined
QL-HH	1.065217	1.021739	1.043478
QMOEA/D-AWA	2.608696	2.576087	2.592391
MOEA/D	4.717391	4.858696	4.788043
NSGA-II	4.239130	4.086957	4.163043
KBEA	2.369565	2.456522	2.413043
p-value	6.865629e-34	6.713355e-35	

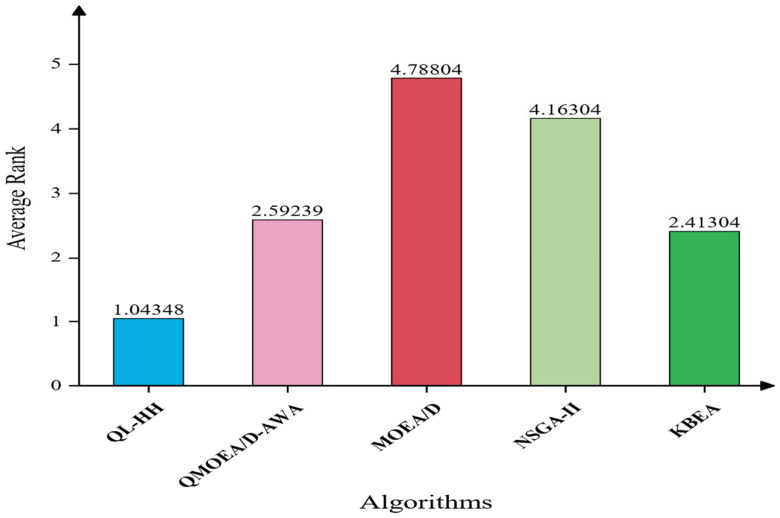
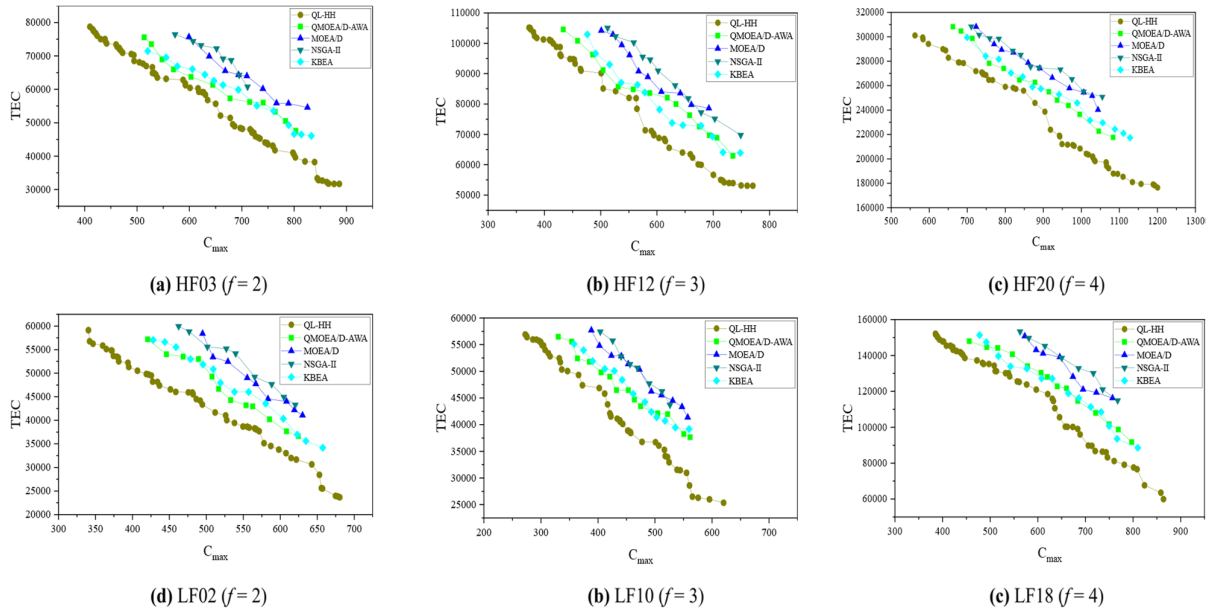


Figure 13. Friedman ranking plot of each algorithm.



(a) HF03 ($f = 2$) **(b)** HF12 ($f = 3$) **(c)** HF20 ($f = 4$)

(d) LF02 ($f = 2$) **(b)** LF10 ($f = 3$) **(c)** LF18 ($f = 4$)

Figure 14. Pareto front plots obtained by each algorithm under different instances.

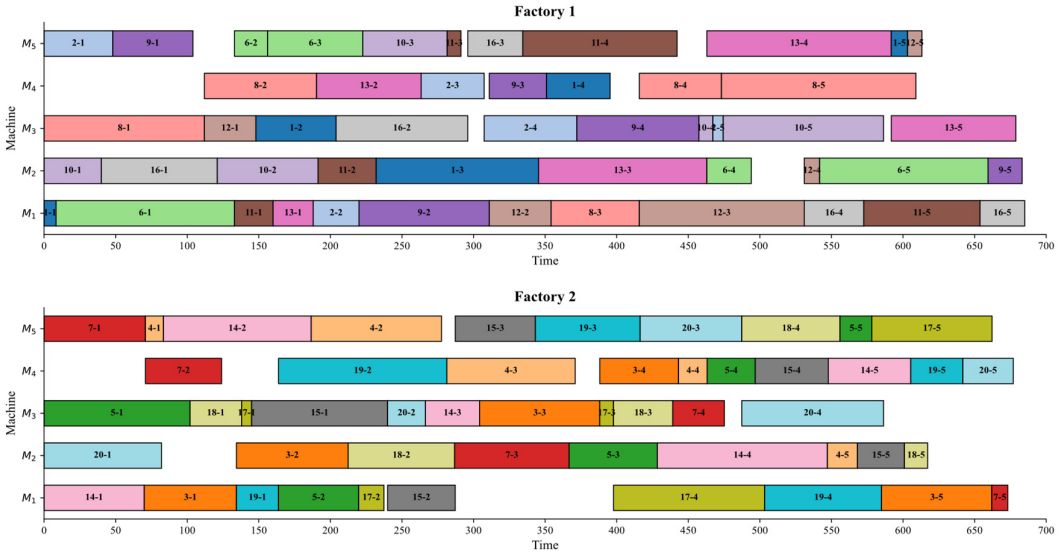


Figure 15. Gantt chart under HF03 (No. 15) in the high-Flexibility dataset ($f = 2$).

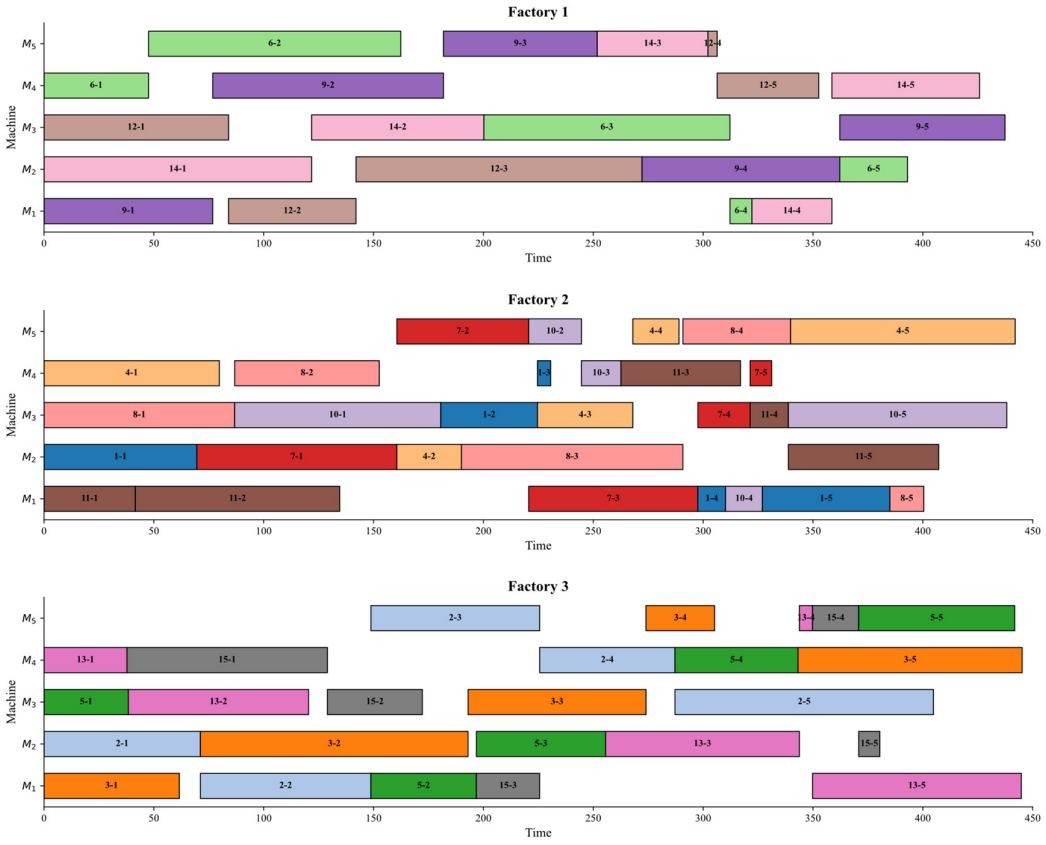


Figure 16. Gantt chart under LF10 (No. 50) in the low-flexibility dataset ($f = 3$).

4.7. Summary

The effectiveness of QL-HH is analyzed from three perspectives. First, the ablation studies in Sections 4.3 and 4.5 quantify the contribution of each component. The hybrid initialization (Section 4.3) provides a well-structured and diverse starting population, with random initialization leading to noticeable degradation in both HV and IGD. Among the four core components evaluated in Section 4.5, removing Q-learning (*QL-HH_nQL*) causes the largest performance drop (HV, −12.03%, IGD, 29.45%), identifying it as the most critical factor. The critical-path speed adjustment operators (*QL-HH_nCP*) rank second, with an IGD degradation of 43.73%, confirming their essential role in maintaining solution quality. The dynamic ε-greedy strategy and the crowding-distance term in the reward also provide moderate but consistent improvements. Second, the operator selection frequencies in Figure 9 show that LLH8 and LLH7 are selected most often (0.241 and 0.129), indicating that the Q-learning agent successfully learns to favor operators that directly balance makespan and energy consumption. Third, statistical tests (Wilcoxon and Friedman) confirm that QL-HH significantly outperforms four state-of-the-art algorithms, with the highest combined ranking of 1.04348. Therefore, the effectiveness of QL-HH stems from the synergy of its core components—hybrid initialization, Q-learning mechanism, dynamic ε-greedy strategy, and critical-path speed adjustment operators—which together enable adaptive and energy-efficient scheduling.

5. Conclusion and future work

5.1. Conclusion

This paper addresses the EEDFJSP with the dual objectives of minimizing makespan and TEC. A multi-objective Q-learning hyper-heuristic algorithm (QL-HH) is proposed to solve this problem efficiently. The algorithm incorporates four hybrid heuristic initialization strategies to construct a high-quality and diverse starting population. An improved ε-greedy mechanism is developed to adaptively select eight low-level heuristic operators, enabling the algorithm to balance exploration and exploitation throughout the search process. Furthermore, critical-path analysis is integrated into the local search phase, allowing the algorithm to identify bottleneck operations and refine promising solutions in a targeted manner. Extensive experiments are conducted on 230 benchmark instances, including high-flexibility and low-flexibility datasets. The results demonstrate that QL-HH consistently outperforms four state-of-the-art algorithms in terms of HV and IGD. Statistical tests (Wilcoxon signed-rank and Friedman) confirm the significance of the performance advantage.

5.2. Future work

Despite its success, several limitations remain. The computational cost grows with population size and number of operations, which may hinder scalability to extremely large instances. Moreover, the current model assumes identical factories and deterministic processing times, which may not fully capture real-world complexities. Future research will focus on the following directions:

(1) Extend the problem formulation to heterogeneous factories, where each factory may have different machine types and capabilities, and incorporate dynamic events such as machine breakdowns and rush order insertions.

(2) Investigate more advanced reinforcement learning techniques, such as deep Q-networks (DQN) or policy gradient methods, to further improve the adaptability of operator selection.

(3) Design parallel or distributed implementations of the hyper-heuristic framework to reduce computation time for large-scale industrial applications.

(4) Integrate additional practical objectives, such as machine utilization, tardiness, and carbon emission constraints, to align with green manufacturing requirements.

In summary, this work provides an effective Q-learning-based hyper-heuristic for the EEDFJSP, and the promising results open up several avenues for further improvement and real-world deployment.

Author contributions

Haojie Li designed the algorithm, conducted the experiments, and wrote the original draft. Qifang Luo contributed to algorithm design, data analysis, and manuscript revision. Yongquan Zhou supervised the research, provided resources, and reviewed the manuscript. All authors have read and agreed to the published version of the manuscript.

Use of Generative-AI tools declaration

The authors acknowledge the use of generative artificial intelligence (AI) tools (ChatGPT and DeepSeek) strictly for technical support during manuscript preparation, limited to language refinement (grammar, clarity). All intellectual content, including study design, data analysis, and scientific conclusions, originated solely from the authors, who reviewed and verified all AI-assisted technical suggestions for accuracy.

Conflict of interest

The authors declare that they have no known competing financial interests or personal relationships

References

1. S. Kumar, P. Kumar, Challenges and Opportunities for Lean 4.0 in Indian SMEs: A Case Study of Jharkhand, in *Industry 4.0 Technologies: Sustainable Manufacturing Supply Chains*, Springer Nature Singapore, Singapore, 2024, 77–98. https://doi.org/10.1007/978-981-99-4894-9_6
2. C. Wang, J. Chen, B. Xu, S. Liu, A Discrete Improved Gray Wolf Optimization Algorithm for Dynamic Distributed Flexible Job Shop Scheduling Considering Random Job Arrivals and Machine Breakdowns, *Processes*, **13** (2025), 1987. https://doi.org/10.3390/pr13071987
3. J. Xie, X. Li, L. Gao, L. Gui, A hybrid genetic tabu search algorithm for distributed flexible job shop scheduling problems, *J. Manuf. Syst.*, **71** (2023), 82–94. https://doi.org/10.1016/j.jmsy.2023.09.002
4. S. Cao, R. Li, W. Gong, C. Lu, Inverse model and adaptive neighborhood search based cooperative optimizer for energy-efficient distributed flexible job shop scheduling, *Swarm Evol. Comput.*, **83** (2023), 101419. https://doi.org/10.1016/j.swevo.2023.101419

5. S. Zhang, T. Hou, Q. Qu, A. Glowacz, S. M. Alqhtani, M. Irfan, et al., An Improved Mayfly Method to Solve Distributed Flexible Job Shop Scheduling Problem under Dual Resource Constraints, *Sustainability*, **14** (2022), 12120. https://doi.org/10.3390/su141912120
6. T. Jamrus, C. F. Chien, M. Gen, K. Sethanan, Hybrid Particle Swarm Optimization Combined With Genetic Operators for Flexible Job-Shop Scheduling Under Uncertain Processing Time for Semiconductor Manufacturing, *IEEE Trans. Semicond. Manuf.*, **31** (2018), 32–41. https://doi.org/10.1109/TSM.2017.2758380
7. C. Y. Zhang, P. G. Li, Y. Q. Rao, S. Z. Li, A new hybrid GA/SA algorithm for the job shop scheduling problem, in *Evolutionary Computation in Combinatorial Optimization, Proceedings*, Springer-Verlag Berlin, Berlin, 2005, 246–259. https://doi.org/10.1007/978-3-540-31996-2_23
8. M. A. Fernandez Perez, F. M. P. Raupp, A Newton-based heuristic algorithm for multi-objective flexible job-shop scheduling problem, *J. Intell. Manuf.*, **27** (2016), 409–416. https://doi.org/10.1007/s10845-014-0872-0
9. M. Ziaee, A heuristic algorithm for solving flexible job shop scheduling problem, *Int. J. Adv. Manuf. Technol.*, **71** (2014), 519–528. https://doi.org/10.1007/s00170-013-5510-z
10. S. J. Wang, B. H. Zhou, L. F. Xi, A filtered-beam-search-based heuristic algorithm for flexible job-shop scheduling problem, *Int. J. Prod. Res.*, **46** (2008), 3027–3058. https://doi.org/10.1080/00207540600988105
11. J. Li, Y. Han, K. Gao, X. Xiao, P. Duan, Bi-Population Balancing Multi-Objective Algorithm for Fuzzy Flexible Job Shop With Energy and Transportation, *IEEE Trans. Autom. Sci. Eng.*, **21** (2024), 4686–4702. https://doi.org/10.1109/TASE.2023.3300922
12. J. Jose Palacios, I. Gonzalez-Rodriguez, C. R. Vela, J. Puente, Robust multiobjective optimisation for fuzzy job shop problems, *Appl. Soft. Comput.*, **56** (2017), 604–616. https://doi.org/10.1016/j.asoc.2016.07.004
13. Z. Liu, J. Wang, C. Zhang, H. Chu, G. Ding, L. Zhang, A hybrid genetic-particle swarm algorithm based on multilevel neighbourhood structure for flexible job shop scheduling problem, *Comput. Oper. Res.*, **135** (2021), 105431. https://doi.org/10.1016/j.cor.2021.105431
14. G. Zhang, L. Gao, Y. Shi, An effective genetic algorithm for the flexible job-shop scheduling problem, *Expert Syst. Appl.*, **38** (2011), 3563–3573. https://doi.org/10.1016/j.eswa.2010.08.145
15. M. Frutos, A. Carolina Olivera, F. Tohme, A memetic algorithm based on a NSGAII scheme for the flexible job-shop scheduling problem, *Ann. Oper. Res.*, **181** (2010), 745–765. https://doi.org/10.1007/s10479-010-0751-9
16. E. Jiang, L. Wang, Multi-objective optimization based on decomposition for flexible job shop scheduling under time-of-use electricity prices, *Knowledge-Based Syst.*, **204** (2020), 106177. https://doi.org/10.1016/j.knosys.2020.106177
17. J. H. Drake, A. Kheiri, E. Ozcan, E. K. Burke, Recent advances in selection hyper-heuristics, *Eur. J. Oper. Res.*, **285** (2020), 405–428. https://doi.org/10.1016/j.ejor.2019.07.073
18. B. Dong, L. Jiao, J. Wu, A two-phase knowledge based hyper-heuristic scheduling algorithm in cellular system, *Knowledge-Based Syst.*, **88** (2015), 244–252. https://doi.org/10.1016/j.knosys.2015.07.028
19. S. S. Choong, L. P. Wong, C. P. Lim, Automatic design of hyper-heuristic based on reinforcement learning, *Inf. Sci.*, **436** (2018), 89–107. https://doi.org/10.1016/j.ins.2018.01.005

20. Y. Fu, Z. Zhang, K. Gao, Q. Pan, H. F. Rahman, Integrated distributed flexible job shop scheduling and vehicle routing problem via Q-learning-based evolutionary algorithms, *Inf. Sci.*, **713** (2025), 122169. https://doi.org/10.1016/j.ins.2025.122169
21. H. B. Song, J. Lin, A genetic programming hyper-heuristic for the distributed assembly permutation flow-shop scheduling problem with sequence dependent setup times, *Swarm and Evolutionary Computation*, **60** (2021), 100807. https://doi.org/10.1016/j.swevo.2020.100807
22. I. Golcuk, F. B. Ozsoydan, Q-learning and hyper-heuristic based algorithm recommendation for changing environments, *Eng. Appl. Artif. Intell.*, **102** (2021), 104284. https://doi.org/10.1016/j.engappai.2021.104284
23. J. Lin, X. Wang, R. Niu, Y. He, A Q-Learning-Based Hyper-Heuristic for Capacitated Electric Vehicle Routing Problem, *IEEE Trans. Intell. Transp. Syst.*, **26** (2025), 15746–15757. https://doi.org/10.1109/TITS.2025.3594393
24. J. Lin, Y. Y. Li, H. B. Song, Semiconductor final testing scheduling using Q-learning based hyper-heuristic, *Expert Syst. Appl.*, **187** (2022), 115978. https://doi.org/10.1016/j.eswa.2021.115978
25. G. Gong, Q. Deng, X. Gong, W. Liu, Q. Ren, A new double flexible job-shop scheduling problem integrating processing time, green production, and human factor indicators, *J. Clean Prod.*, **174** (2018), 560–576. https://doi.org/10.1016/j.jclepro.2017.10.188
26. M. S. Zadeh, Y. Katebi, A. Doniavi, A heuristic model for dynamic flexible job shop scheduling problem considering variable processing times, *Int. J. Prod. Res.*, **57** (2019), 3020–3035. https://doi.org/10.1080/00207543.2018.1524165
27. H. Wang, B. R. Sarker, J. Li, J. Li, Adaptive scheduling for assembly job shop with uncertain assembly times based on dual Q-learning, *Int. J. Prod. Res.*, **59** (2021), 5867–5883. https://doi.org/10.1080/00207543.2020.1794075
28. K. Geng, L. Liu, S. Wu, A reinforcement learning based memetic algorithm for energy-efficient distributed two-stage flexible job shop scheduling problem, *Sci. Rep.*, **14** (2024), 30816. https://doi.org/10.1038/s41598-024-81064-z
29. J. Lin, Z. J. Wang, X. Li, A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem, *Swarm Evol. Comput.*, **36** (2017), 124–135. https://doi.org/10.1016/j.swevo.2017.04.007
30. B. H. Abed-alguni, N. A. Alawad, Distributed Grey Wolf Optimizer for scheduling of workflow applications in cloud environments, *Appl. Soft. Comput.*, **102** (2021), 107113. https://doi.org/10.1016/j.asoc.2021.107113
31. L. De Giovanni, F. Pezzella, An Improved Genetic Algorithm for the Distributed and Flexible Job-shop Scheduling problem, *Eur. J. Oper. Res.*, **200** (2010), 395–408. https://doi.org/10.1016/j.ejor.2009.01.008
32. H. C. Chang, T. K. Liu, Optimisation of distributed manufacturing flexible job shop scheduling by using hybrid genetic algorithms, *J. Intell. Manuf.*, **28** (2017), 1973–1986. https://doi.org/10.1007/s10845-015-1084-y
33. B. Marzouki, O. B. Driss, K. Ghedira, Decentralized Tabu Searches in Multi Agent system for Distributed and Flexible Job shop Scheduling Problem, in *2017 Ieee/Acs 14th International Conference on Computer Systems and Applications (aiccsa)*, IEEE, New York, 2017, 1019–1026. https://doi.org/10.1109/AICCSA.2017.133

34. R. Wu, E. Luo, X. Li, H. Tang, Y. Li, Hybrid artificial bee colony algorithm with Q-learning for distributed heterogeneous flexible job shop scheduling problem considering machine preventive maintenance, *Swarm Evol. Comput.*, **98** (2025), 102134. https://doi.org/10.1016/j.swevo.2025.102134
35. Q. Zhang, W. Shao, Z. Shao, D. Pi, J. Gao, Deep reinforcement learning driven trajectory-based meta-heuristic for distributed heterogeneous flexible job shop scheduling problem, *Swarm Evol. Comput.*, **91** (2024), 101753. https://doi.org/10.1016/j.swevo.2024.101753
36. F. Zhao, Z. Fu, L. Wang, H. Sang, A Heterogeneous Graph Reinforcement Learning Framework With Question-Aware Neighborhood Aggregation and Interoption Prompt Attention for Dynamic Flexible Job Shop Scheduling Problem, *IEEE Trans. Ind. Inform.*, **22** (2026), 2863–2874. https://doi.org/10.1109/TII.2025.3646962
37. E. Jiang, L. Wang, Z. Peng, Solving energy-efficient distributed job shop scheduling via multi-objective evolutionary algorithm with decomposition, *Swarm Evol. Comput.*, **58** (2020), 100745. https://doi.org/10.1016/j.swevo.2020.100745
38. R. Li, W. Gong, L. Wang, C. Lu, X. Zhuang, Surprisingly Popular-Based Adaptive Memetic Algorithm for Energy-Efficient Distributed Flexible Job Shop Scheduling, *IEEE T. Cybern.*, **53** (2023), 8013–8023. https://doi.org/10.1109/TCYB.2023.3280175
39. F. Zhao, M. Li, N. Zhu, T. Xu, Jonrinaldi, Multi-objective fitness landscape-based estimation of distribution algorithm for distributed heterogeneous flexible job shop scheduling problem, *Appl. Soft. Comput.*, **171** (2025), 112780. https://doi.org/10.1016/j.asoc.2025.112780
40. F. C. Wu, B. Qian, R. Hu, Z. Q. Zhang, B. Wang, A Q-Learning-Based Hyper-Heuristic Evolutionary Algorithm for the Distributed Flexible Job-Shop Scheduling Problem, in *Advanced Intelligent Computing Technology and Applications, Icic 2023, Pt I*, Springer-Verlag Singapore Pte Ltd, Singapore, 2023, 251–261. https://doi.org/10.1007/978-981-99-4755-3_22
41. Z. Wang, M. He, H. Chen, Y. Hu, Y. Xia, A Q-learning-based evolutionary algorithm for solving the low-carbon multi-objective flexible job shop scheduling problem, *Comput. Oper. Res.*, **185** (2026), 107266. https://doi.org/10.1016/j.cor.2025.107266
42. Z. Q. Zhang, Z. M. Wu, B. Qian, R. Hu, A reward-shaping dueling distributed multi-agent deep reinforcement learning framework for dynamic flexible job shop scheduling with random job arrivals, *Expert Syst. Appl.*, **297** (2026), 128951. https://doi.org/10.1016/j.eswa.2025.128951
43. Y. Li, Y. He, Y. Wang, F. Tao, J. W. Sutherland, An optimization method for energy-conscious production in flexible machining job shops with dynamic job arrivals and machine breakdowns, *J. Clean Prod.*, **254** (2020), 120009. https://doi.org/10.1016/j.jclepro.2020.120009
44. M. Hassanchokami, A. Vital-Soto, J. Olivares-Aguila, The Role of Environmental Factors in the Flexible Job-Shop Scheduling Problem: A Literature Review, in *Ifac Papersonline*, Elsevier, Amsterdam, 2022, 175–180. https://doi.org/10.1016/j.ifacol.2022.09.386
45. Z. Pan, L. Wang, J. Wang, Q. Zhang, A Bi-Learning Evolutionary Algorithm for Transportation-Constrained and Distributed Energy-Efficient Flexible Scheduling, *IEEE Trans. Evol. Comput.*, **29** (2025), 232–246. https://doi.org/10.1109/TEVC.2024.3354850
46. F. Yu, C. Lu, J. Zhou, L. Yin, K. Wang, A knowledge-guided bi-population evolutionary algorithm for energy-efficient scheduling of distributed flexible job shop problem, *Eng. Appl. Artif. Intell.*, **128** (2024), 107458. https://doi.org/10.1016/j.engappai.2023.107458

47. Z. Pan, D. Lei, L. Wang, A Knowledge-Based Two-Population Optimization Algorithm for Distributed Energy-Efficient Parallel Machines Scheduling, *IEEE T. Cybern.*, **52** (2022), 5051–5063. https://doi.org/10.1109/TCYB.2020.3026571
48. J. Wang, Y. Liu, S. Ren, C. Wang, W. Wang, Evolutionary game based real-time scheduling for energy-efficient distributed and flexible job shop, *J. Clean Prod.*, **293** (2021), 126093. https://doi.org/10.1016/j.jclepro.2021.126093
49. L. Chen, C. Yang, D. Zhu, T. Li, A population diffusion algorithm for energy-efficient distributed flexible job shop scheduling problem, *Appl. Soft. Comput.*, **195** (2026), 115056. https://doi.org/10.1016/j.asoc.2026.115056
50. Z. Q. Zhang, X. P. Zhu, Y. X. Xu, B. Qian, B. Yang, R. Hu, MEDHEA: Multidimensional estimation of distribution based hyper-heuristic evolutionary algorithm for energy-efficient distributed assembly no-wait flow-shop scheduling problem, *Expert Syst. Appl.*, **271** (2025), 126526. https://doi.org/10.1016/j.eswa.2025.126526
51. B. Naderi, R. Ruiz, The distributed permutation flowshop scheduling problem, *Comput. Oper. Res.*, **37** (2010), 754–768. https://doi.org/10.1016/j.cor.2009.06.019
52. Z. Zhang, Y. Fu, K. Gao, Q. Pan, M. Huang, A learning-driven multi-objective cooperative artificial bee colony algorithm for distributed flexible job shop scheduling problems with preventive maintenance and transportation operations, *Comput. Ind. Eng.*, **196** (2024), 110484. https://doi.org/10.1016/j.cie.2024.110484
53. K. Z. Gao, P. N. Suganthan, M. F. Tasgetiren, Q. K. Pan, Q. Q. Sun, Effective ensembles of heuristics for scheduling flexible job shop problem with new job insertion, *Comput. Ind. Eng.*, **90** (2015), 107–117. https://doi.org/10.1016/j.cie.2015.09.005
54. J. Deng, J. Zhang, S. Yang, A cooperative Q-learning-based memetic algorithm for distributed assembly heterogeneous flexible flowshop scheduling, *Expert Syst. Appl.*, **288** (2025), 128198. https://doi.org/10.1016/j.eswa.2025.128198
55. K. Lei, P. Guo, W. Zhao, Y. Wang, L. Qian, X. Meng, et al., A multi-action deep reinforcement learning framework for flexible Job-shop scheduling problem, *Expert Syst. Appl.*, **205** (2022), 117796. https://doi.org/10.1016/j.eswa.2022.117796
56. P. Brandimarte, Routing and scheduling in a flexible job shop by tabu search, *Ann Oper Res*, **41** (1993), 157–183. https://doi.org/10.1007/BF02023073
57. J. Hurink, B. Jurisch, M. Thole, Tabu search for the job-shop scheduling problem with multi-purpose machines, *OR Spektrum*, **15** (1994), 205–215. https://doi.org/10.1007/BF01719451
58. F. Xiong, H. Liu, Logic-based benders decomposition methods for the distributed flexible job shop scheduling problem, *Eur. J. Oper. Res.*, **329** (2026), 778–797. https://doi.org/10.1016/j.ejor.2025.08.039
59. S. Du, W. Zhou, D. Wu, M. Fei, An effective discrete monarch butterfly optimization algorithm for distributed blocking flow shop scheduling with an assembly machine, *Expert Syst. Appl.*, **225** (2023), 120113. https://doi.org/10.1016/j.eswa.2023.120113
60. J. Wang, H. Han, L. Wang, A Feedback Learning-Based Memetic Algorithm for Energy-Aware Distributed Flexible Job-Shop Scheduling With Transportation Constraints, *IEEE Trans. Evol. Comput.*, **29** (2025), 1085–1099. https://doi.org/10.1109/TEVC.2024.3388527

61. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Trans. Evol. Comput.*, **6** (2002), 182–197. https://doi.org/10.1109/4235.996017

AIMS Press