Tsinghua Science and Technology
Doi: 10.26599/TST.2026.9010050

Research Article

Automatic generation of heuristic dispatching rules for dynamic hybrid flow shop scheduling via personalized multi-island reflective evolution

Yuning Lei, Jin Huang, Xinyu Li, Qihao Liu (✉), and Liang Gao

Abstract Dynamic hybrid flow shop scheduling problems (DHFSP) are critical in modern manufacturing systems, where uncertainties such as order fluctuations and equipment failures pose significant challenges. Traditional exact methods and metaheuristics struggle to meet real-time decision-making requirements under such dynamic conditions. Heuristic dispatching rules (HDRs) have been widely adopted for their rapid response capabilities. In recent years, large language models (LLMs) have demonstrated remarkable capabilities in code generation and logical reasoning, showing promising potential for automated HDR design. However, existing LLM-based methods predominantly adopt single-population evolution strategies, which suffer from insufficient population diversity, limited semantic-level reasoning, and premature convergence, thereby frequently becoming trapped in local optima. To address these challenges, this paper proposes a personalized multi-island reflective evolution framework that assigns distinct exploration personalities to multiple parallel sub-populations and incorporates an LLM-driven semantic reflection mechanism to achieve efficient search space coverage and enhanced exploration depth. The framework employs a two-stage strategy: offline training constructs a robust rule library through diversified dynamic scenarios, while online application enables rapid real-time decision-making. Experimental results on 300 test instances demonstrate that the proposed method outperforms traditional HDRs, conventional evolutionary rule generation methods, and state-of-the-art LLM-based approaches, exhibiting superior stability and generalization capability.

Keywords Dynamic hybrid flow shop scheduling problem, Large language models, Multi-Island reflective evolution, Automatic heuristic dispatching rules.

• Yuning Lei is with the School of Naval Architecture and Ocean Engineering, Huazhong University of Science and Technology, Wuhan, 430074, China. E-mail: leiyuning@hust.edu.cn
• Jin Huang, Xinyu Li, Qihao Liu, and Liang Gao are with the State Key Laboratory of Intelligent Manufacturing Equipmentand Technology, Huazhong University of Science and Technology,Wuhan, 430074, China. E-mail: huangjin_@hust.edu.cn (J. Huang), lixinyu@hust.edu.cn (X. Li), lllqh@hust.edu.cn (Q. Liu), gaoliang@hust.edu.cn (L. Gao)
* Qihao Liu is the corresponding author. E-mail: lllqh@hust.edu.cn
Manuscript received: 2026-02-25; revised: 2026-04-20; accepted: 2026-05-12

1 Introduction

Against the backdrop of global manufacturing transformation toward intelligence and flexibility, production scheduling systems face increasingly complex challenges [1, 2]. Hybrid flow shops represent a production paradigm widely employed in semiconductor manufacturing [3], automobile assembly [4, 5], steel production [6], and other industrial sectors. This configuration features multiple serial processing stages, each equipped with parallel heterogeneous machines. Real-world production environments involve various uncertainties: stochastic customer demand requires continuous insertion of new orders [7], sudden equipment failures [8], and processing time variability adds further complexity [9]. These dynamic factors necessitate scheduling decision mechanisms capable of rapid adaptation.

The dynamic hybrid flow shop scheduling problem (DHFSP) constitutes a classic NP-hard combinatorial optimization problem involving multiple interdependent decisions: job sequencing at each stage, machine allocation for operations, and real-time disturbance response [10]. Traditional exact algorithms such as mixed integer linear programming (MILP) require frequent rescheduling in dynamic environments, with computational time scaling with problem size [11, 12]. Metaheuristic algorithms can identify near-optimal solutions for large-scale problems through iterative search [13]. In contrast, heuristic dispatching rules (HDRs) have been widely adopted in industrial practice due to their O(1) decision complexity and immediate response capability.

However, the design of HDRs is heavily dependent on domain expert knowledge, resulting in prolonged development cycles and substantial costs. More critically, these manually crafted rules exhibit limited generalization capability. No single rule demonstrates consistent superiority across diverse production scenarios. To achieve automated rule generation, evolutionary methods such as genetic programming (GP) [14, 15] and gene expression programming (GEP) [16] have emerged. These methods automatically construct rules through the evolution of symbolic expression trees or genetic encodings, achieving moderate success. Nevertheless, their

core mechanisms rely on stochastic mutation and crossover operations, lacking explicit exploitation of underlying problem structures. This results in low search efficiency and susceptibility to premature convergence, with particularly limited performance improvements when confronting complex scenarios involving multiple types of dynamic disturbances.

In recent years, large language models (LLMs) have demonstrated considerable potential in generating heuristic algorithms for combinatorial optimization problems including traveling salesman problem (TSP), online bin packing problem (BPP) [17, 18] and scheduling [19, 20]. The reflective evolution (ReEvo) framework [21] has incorporated verbal reflection mechanisms for algorithm generation. Within scheduling, LLMs have been successfully applied to dynamic job shop problems [22, 23]. However, existing LLM-based methods face several challenges when applied to DHFSP. First, single-population evolution frameworks are sensitive to initial conditions and may experience premature convergence due to limited population diversity. Second, traditional evolutionary operators do not fully leverage the semantic understanding capabilities of LLMs, affecting the logical coherence of generated rules. Third, the absence of training strategies tailored to dynamic scheduling characteristics limits rule robustness across multiple disturbance types.

To address these challenges, this paper proposes a personalized multi-island reflective evolution (Multi-Island-ReEvo) framework. The core innovation lies in assigning distinct exploration personalities to multiple parallel sub-populations while employing hierarchical reflection mechanisms to guide semantic-level rule optimization. The main contributions include as follows:

(1) **Personalized multi-island architecture:** Each subpopulation receives unique exploration characteristics through LLM-generated strategy personalities, enabling targeted exploration in distinct solution space regions and ensuring structural population diversity.

(2) **Dual-layer reflective evolution mechanism:** Short-term reflection integrated with crossover extracts performance differentials as verbal gradients, while long-term reflection integrated with mutation synthesizes historical experience into global guidance, enabling semantic-level reasoning beyond stochastic search.

(3) **Offline training and online application paradigm:** The offline phase constructs diverse case sets with multiple disturbance types for learning generalizable rules. The online phase leverages acquired knowledge for rapid adaptation to novel scenarios.

(4) **Systematic experimental validation:** Experiments on 300 test instances demonstrate that Multi-Island-ReEvo significantly outperforms GP, GEP, 12 classical HDRs, and state-of-the-art LLM methods including ReEvo [21] and HsEvo [24], with superior stability across disturbance types.

The remainder of this paper is organized as follows. Section 2 reviews relevant literature. Section 3 introduces the Multi-Island-ReEvo methodology. Section 4 presents the training and application framework. Section 5 reports experimental results and analysis. Section 6 summarizes contributions and discusses future directions.

2 Related Work

2.1 Traditional methods for solving DHFSP

The HFSP is a well-known NP-hard combinatorial optimization problem commonly found in manufacturing systems [25, 26]. The complexity of this problem increases significantly when dynamic elements like random order arrivals, machine breakdowns, and uncertain processing times are considered [27]. Chen et al. [28] introduced a neuro-evolution-based multi-agent system to address uncertain processing durations and flexible maintenance schedules. Similarly, Deng et al. [9] represented new job insertion times using triangular fuzzy numbers and developed a dynamic decision-driven memetic algorithm.

One approach to handling dynamic events is rescheduling by means of re-optimization. Tliba et al. [29] introduced a hybrid method combining digital twin technology with MILP simulation, although it still faces challenges with computational time when applied to large-scale problems. Metaheuristic algorithms offer a compromise between solution quality and computational speed. For example, Peng et al. [6] presented an enhanced imperialist competitive algorithm tailored for steelmaking and refining processes. Zuo and Zhao [30] created a dual-population cooperative scatter search method that incorporates predictive maintenance strategies. More recent developments have tackled increasingly complex scenarios. Li et al. [31] proposed an evolutionary greedy algorithm designed for dynamic cascaded flowshops, integrating both distributed permutation flowshop and hybrid flowshop stages. Similarly, Tao et al. [32] developed an iterated greedy algorithm enhanced with reinforcement learning to address distributed HFSP involving job merging and rework.

Deep reinforcement learning (DRL) has gained interest for its ability to perform end-to-end learning and make decisions in real time [33]. Liu et al. [34] combined DRL with multi-agent systems to address dynamic reentrant HFSP. Luo and Yan [35] utilized an evolution strategy-guided DRL

approach for dynamic HFSP involving unrelated parallel machines. Xu et al. [36] introduced a hierarchical multi-graph network framework that overcomes the dimensional limitations of neural networks. Although these DRL methods have shown promising results, their adoption in industrial settings is hindered by their black-box nature, which reduces interpretability and trust—key factors for explainability in practical production decision-making.

In contrast, HDRs continue to be extensively used in industry because of their key benefits: constant-time decision complexity, the ability to respond instantly, transparency, and straightforward implementation. Wang et al. [10] introduced a clustering scheduling model that integrates shortest processing time rules with simulated annealing. Shi et al. [37] developed an event-driven priority weight local search approach for sustainable HFSP. Nevertheless, manually crafted rules have drawbacks such as limited generalizability and high design costs. To overcome these challenges, methods for automatic rule generation have been developed. For example, Duan et al. [38] proposed a GP hyper-heuristic for dynamic, energy-efficient HFSP, while Luo and Yan [39] created a Q-learning based selection hyper-heuristic that adaptively chooses from a set of predefined heuristics.

2.2 Automatic algorithm design based on LLMs

Automatic algorithm design and configuration has emerged as a crucial research domain aimed at reducing the reliance on human domain expertise by automating the creation, selection, and tuning of optimization algorithms for specific problem classes. Within this broader field, hyper-heuristics represent a prominent and core methodology. Unlike traditional metaheuristics that search directly within a problem's solution space, hyper-heuristics operate at a higher level of abstraction—searching through a heuristic space to intelligently select, combine, or generate effective algorithms. Traditional generative hyper-heuristics, such as GP [14, 15] and gene expression programming GEP [16], automate this process by evolving new rules from a predefined set of components. While these methods significantly enhance the adaptability of dispatching rules across dynamic environments, they primarily rely on stochastic evolutionary operators. Consequently, they often lack the ability to explicitly exploit the underlying algorithmic logic, and their performance is fundamentally constrained by the limitations of their fixed component sets.

To overcome the semantic limitations of traditional automated algorithm design approaches, LLMs have recently introduced a revolutionary paradigm shift. By being pretrained on extensive code and text datasets [40], LLMs bring profound semantic understanding and logical reasoning capabilities to the automated generation of heuristics. Liu et al. [17] presented the evolution of heuristics (EoH) framework, which systematically showcases the capability of LLMs to automatically develop heuristic algorithms. Romera-Paredes et al. [18] published FunSearch in Nature, achieving new benchmarks in online bin packing problems by fine-tuning PaLM2 with island-based evolutionary strategies.

To improve the reliability of optimization, reflection mechanisms have become a central focus in developing closed-loop evolutionary systems. Ye et al. [21] introduced reflective evolution (ReEvo), which integrates problem comprehension and iterative reflection. Dat et al. [24] presented harmony search evolution (HSEvo), designed to dynamically balance exploration and exploitation. Guo et al. [41] created nested-refinement metamorphosis (NeRM), which enhances task descriptions through nested refinement. Zhong et al. [42] made a groundbreaking contribution with LLM assisted hyper-heuristic optimization algorithm (LLMOA) by cleverly utilizing LLMs as high-level components to intelligently organize low-level heuristics. This framework not only highlights the strong strategic planning abilities of LLMs but also successfully applies their linguistic skills to numerical optimization, achieving impressive performance compared to leading optimizers on complex engineering challenges.

These approaches have been further extended to address the intricate constraints of scheduling domains. More recently, Google DeepMind introduced AlphaEvolve [43], an evolutionary coding agent that extends FunSearch to evolve entire codebases, achieving notable success in data center scheduling optimization at industrial scale. Huang et al. [23] pioneered the application of population self-evolution to dynamic job shops, while Forniés et al. [44] and Li et al. [20] addressed multi-objective and hybrid batching constraints respectively. In the specific context of intelligent manufacturing, Zhao et al. [45] proposed a LLM-based multi-agent system that fundamentally reimagines shop floor management. Liao et al. [46] proposed LLM4EO, which leverages LLMs for online evolutionary operator design in flexible job shop scheduling, adaptively refining operator strategies when population evolution stagnates. By leveraging modular agents for interpretable negotiation, their approach overcomes the rigidity of traditional heuristic rules and the training costs of reinforcement learning. Extensive experiments confirmed that their system achieves superior adaptability and stability in physical environments without requiring pre-training. However, existing methods predominantly adopt single-population frameworks which are susceptible to premature convergence,

lacking systematic investigation into scenarios involving multiple simultaneous disturbances.

2.3 Population diversity maintenance mechanisms in evolutionary algorithms

Population diversity is critical for evolutionary algorithms to avoid premature convergence [47]. Traditional niching techniques such as fitness sharing and crowding distance encourage population diffusion to sparse regions by penalizing individuals in crowded areas [48]. Li et al. [49] proposed a multi-objective hybrid evolutionary algorithm based on variable weight strategy for distributed HFSP with batch processing machines, guiding search direction by dynamically adjusting weight vectors. These methods achieve diversity through fitness landscape modification, though they primarily operate at the individual level.

Multi-population approaches, particularly island models, provide a more natural diversity maintenance mechanism through geographic isolation and periodic migration. Cohoon et al. [50] first proposed this distributed evolutionary framework inspired by punctuated equilibria theory, demonstrating that geographic isolation combined with periodic migration yields superior solutions. Subsequent foundational studies established that migration interval and topology are the most critical design parameters governing the balance between exploration and exploitation [51, 52]. Alba and Tomassini [53] provided a comprehensive survey that unified the theoretical understanding of parallel evolutionary algorithms.

Building upon this general paradigm, island models have been increasingly applied to scheduling optimization with problem-specific adaptations. Zhang et al. [54] developed a migrating birds optimization algorithm adopting a V-shaped dual-line collaborative mechanism, where two evolutionary lines employ different genetic operators and exchange information through regular shuffling. This structure enables parallel exploration of distinct solution space regions. Zhu et al. [55] proposed a collaborative learning perception dynamic hierarchical hyper-heuristic with a hierarchical reinforcement learning framework, where the upper-level RL guides neighborhood structure selection through estimation of distribution algorithm, while the lower-level RL executes dynamic neighborhood switching. Such hierarchical architectures demonstrate the potential of intelligent coordination among multiple populations.

Recent research has explored integrating machine learning techniques to dynamically adjust diversity maintenance strategies. Qin et al. [56] proposed a Q-learning inspired selection mechanism for energy-efficient scheduling of distributed HFSP with blocking constraints, providing historical experience guidance for different factory scheduling scenarios. Pan et al. [57] developed dynamic sub-harmony memory mechanisms in harmony search algorithms, dynamically partitioning harmony memory into multiple small-scale sub-memories for independent evolution, achieving a balance between rapid convergence and diversity maintenance. In multi-objective optimization, Wang et al. [58] proposed a multi-population co-evolutionary algorithm setting different optimization objectives for sub-populations, enabling deep exploration of different solution space through objective-level differentiation.

Despite these advances, traditional island models exhibit limitations when combined with LLMs. The differences among sub-populations primarily originate from random initialization and geographic isolation, lacking mechanisms to endow sub-populations with differentiated exploration preferences at the strategic level. While reflection mechanisms have proven effective in single populations, achieving cross-population knowledge sharing and hierarchical reflection in multi-population frameworks remains underexplored. Addressing these gaps, this paper proposes leveraging LLMs' semantic understanding capabilities for multi-population personalized design, establishing distinct evolutionary objectives and reflection strategies for each island to achieve efficient coverage and deep exploration of the search space.

3 Generating HDRs with Multi-Island-ReEvo

To effectively search for high-quality HDRs in combinatorial optimization problems such as DHFSP with complex solution spaces, this paper proposes the Multi-Island-ReEvo, which is shown in Fig 1. All of the prompts are illustrated in supplementary material Fig. S1 and S2. This method introduces an innovative LLM-driven population diversity maintenance mechanism combined with semantic understanding-based reflective evolution operators, aiming to overcome the tendency of traditional single-population evolution algorithms toward premature convergence. Multi-Island-ReEvo not only maintains diversity through geographic isolation but also leverages LLMs to endow each sub-population with unique exploration personalities. These personalities guide targeted in-depth exploration in distinct regions of the solution space.

3.1 Individual encoding

In the Multi-Island-ReEvo method, each individual is represented as an executable heuristic code, specifically implemented as a Python function. This approach of directly using

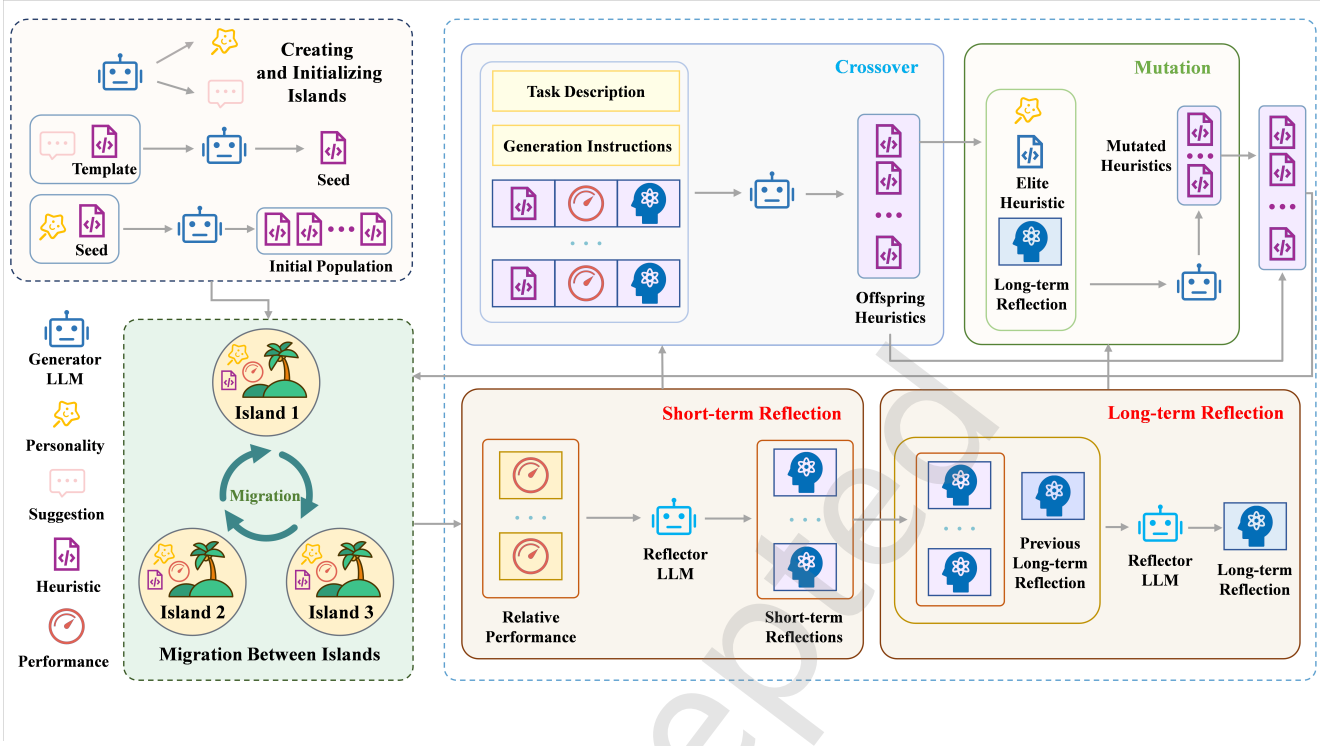

Fig. 1 The Multi-Island-ReEvo method.

code as genetic encoding enables LLMs to fully leverage their deep understanding of programming languages and perform evolutionary operations such as crossover and mutation directly at the semantic level. Unlike traditional GP, which encodes rules as symbolic expression trees and performs crossover and mutation through syntactic subtree operations, this direct code representation allows LLMs to fully understand the programming logic and generate structurally coherent modifications. This avoids issues such as semantic information loss and code bloat commonly encountered in GP using indirect encoding.

3.2 Creating and initializing islands

The objective of the initialization phase is to create multiple parallel populations with substantial diversity and unique evolutionary starting points. As described in Algorithm 1, this process primarily consists of three steps.

3.2.1 Generating personalized islands using LLM

The algorithm first creates N_{isl} parallel island instances, forming the set $S = \{S_1, S_2, \ldots, S_{N_{\text{isl}}}\}$. Unlike traditional multi-island genetic algorithms where all sub-populations follow identical evolution logic, Multi-Island-ReEvo assigns each island S_k a unique personality Π_k generated by the LLM. This personality is a natural language description that establishes a macro-level strategic direction for the island's

Algorithm 1 Multi-Island Initialization

Require: Number of islands N, population size M
Ensure: Initialized island set $\mathcal{I}$

Step 1: Generate island personalities
$\mathcal{P} \leftarrow$ Generate Personalities(N)
Step 2: Generate diverse seed suggestions
$\mathcal{S}_{suggest} \leftarrow$ Generate Seed Suggestions(N)
Step 3: Initialize each island
$\mathcal{I} \leftarrow \emptyset$
for $i = 0$ to $N-1$ **in parallel do**
  $isl_i.pers \leftarrow \mathcal{P}[i]$
  $isl_i.seed \leftarrow$ Generate Seed$(base_seed, \mathcal{S}_{suggest}[i])$
  $isl_i.pop \leftarrow$ Initialize Pop$(M, isl_i.pers, isl_i.seed)$
  $isl_i.pop \leftarrow$ Evaluate$(isl_i.pop)$
  $isl_i.elite \leftarrow \text{argmin}_{h \in isl_i.pop}\text{Obj}(h)$
  $isl_i.reflection \leftarrow \emptyset$
  $\mathcal{I} \leftarrow \mathcal{I} \cup \{isl_i\}$
end for
return $\mathcal{I}$

evolutionary search. For instance, one island may receive a "conservative" personality that focuses on incremental refinement of existing rule structures, while another receives an "exploration" personality that encourages bold structural innovations and the introduction of novel heuristic factors. Some personalities generated by LLM are shown in supplementary material Fig. S3. This design elevates the multi-island model from simple population isolation to multi-strategy collaborative search, fundamentally ensuring differentiation in evolutionary trajectories across islands.

Let T_{Π} denote the prompt template for generating personalities. It instructs the LLM to act as an expert in Multi-Island Genetic Algorithms and design distinct personalities for each island. Specifically, the prompt provides example personality types and asks the LLM to generate analogous but distinct personalities for all islands. The full prompt template is provided in the supplementary material S6. The personality set for all islands can be produced through a single call to the generator LLM:

$$\{\Pi_1, \ldots, \Pi_{N_{\text{isl}}}\} = \text{LLM}_{\text{gen}}(T_{\Pi}, N_{\text{isl}}) \tag{1}$$

These generated natural language descriptions Π_k serve as key conditional inputs for subsequent evolution operations (particularly population initialization and mutation), continuously guiding each island's evolutionary trajectory.

3.2.2 *Generating diverse seed heuristic rules*

To maximize inter-island differences from the initial stage, each island S_k constructs its initial population around a unique seed heuristic rule $h_{k,0}^{\text{seed}}$. The generation of this seed rule follows a two-stage LLM interaction process.

First, the reflector LLM generates textual suggestions for the seed functions (the initial heuristic rule code) of all islands, as shown in supplementary material Fig. S4. This aims to ensure that the mathematical structure of the initial population on each island has initial differences:

$$\{\text{Sugg}_1, \ldots, \text{Sugg}_{N_{\text{isl}}}\} = \text{LLM}_{\text{ref}}(T_{\text{sugg}}, N_{\text{isl}}) \tag{2}$$

Subsequently, for each island k, the generator LLM produces specific seed function code based on a base heuristic rule template h_0^{base} and the corresponding suggestion Sugg_k:

$$h_{k,0}^{\text{seed}} = \text{LLM}_{\text{gen}}(T_{\text{seed}}, h_0^{\text{base}}, \text{Sugg}_k) \tag{3}$$

Note that the base heuristic rule template h_0^{base} is shared across all islands, serving as a common and simple starting point. The inter-island differentiation originates from the island-specific suggestions Sugg_k, which guide the generator LLM to produce structurally diverse seed rules from this common base.

This two-stage design ensures that each island's seed function both adheres to its personalized direction and exhibits diversity in mathematical form, establishing a differentiated foundation for subsequent independent evolution. It should be particularly noted that in the first stage, the reflector LLM comprehensively considers the personality characteristics of all islands to generate seed suggestions with substantial differences. In the second stage, the generator LLM produces specific code implementations aligned with each island's strategic direction based on its specific personality and suggestions.

3.2.3 *Parallelized population initialization*

After obtaining each island's personality Π_k and seed rule $h_{k,0}^{\text{seed}}$, the system generates initial populations $P_k(0) = \{h_{k,0}^{(1)}, \ldots, h_{k,0}^{(N_p)}\}$ for each island in parallel. Specifically, for each island S_k, the system sends a prompt containing task specifications, the island's unique personality Π_k, and its seed rule $h_{k,0}^{\text{seed}}$ to the generator LLM, requesting generation of an initial population of size N_p. Since each island's generation prompt is guided by its unique personality, the LLM generates initial rule sets with specific stylistic preferences. This stands in contrast to GP-based methods, where initial populations are generated through random tree construction without any strategic guidance, resulting in arbitrary and often redundant starting points.

For any island $k \in \{1, \ldots, N_{\text{isl}}\}$, the generation process of its initial population can be expressed as:

$$P_k(0) = \{\text{LLM}_{\text{gen}}(T_{\text{init}}, \Pi_k, h_{k,0}^{\text{seed}})\}_{i=1}^{N_p} \tag{4}$$

The entire initialization process employs multiprocessing technology to achieve parallelization, significantly improving algorithm execution efficiency.

3.3 Parallel reflective evolution: intra-island dynamics

Algorithm 2 Single island evolution

Require: Island isl_i, population size M, mutation rate P_m

Select
$Pairs \leftarrow$ Random Select$(isl_i.pop \cup isl_i.elite, M)$
Short-term reflection and crossover
$SR \leftarrow []$
for $(p_1, p_2) \in Pairs$ **in parallel do**
  $sr \leftarrow$ Short-term Reflect(p_1, p_2)
  SR.append(sr)
end for
$P_{cross} \leftarrow$ Crossover$(Pairs, SR)$
$P_{cross} \leftarrow$ Evaluate Population(P_{cross})
$isl_i.pop \leftarrow P_{cross}$
$isl_i.elite \leftarrow$ Update$(isl_i.pop, isl_i.elite)$
Long-term reflection
$isl_i.LR \leftarrow$ Long-term Reflect$(SR, isl_i.LR)$
Mutation with personality
$n_{mut} \leftarrow \lfloor M \times P_m \rfloor$
$prompt \leftarrow$ Get Prompt$(isl_i.elite, isl_i.LR, isl_i.pers)$
$P_{mut} \leftarrow$ Mutate$(prompt, n_{mut})$
$P_{mut} \leftarrow$ Evaluate Population(P_{mut})
$isl_i.pop \leftarrow isl_i.pop \cup P_{mut}$
$isl_i.elite \leftarrow$ Update$(isl_i.pop, isl_i.elite)$
return isl_i

Algorithm 2 elaborates the reflective evolution process that executes independently and in parallel within each island. This process leverages the semantic reasoning capability of LLMs to perform evolution operations based on logical reflection, transcending traditional genetic algorithms.

3.3.1 Parent selection

Within the population $P_k(t)$ of each island k (where t denotes the current iteration count), the algorithm selects N_c pairs of parent individuals $\{(h_a^{(i)}, h_b^{(i)})\}_{i=1}^{N_c}$ for reproduction. The selection strategy follows the approach of ReEvo, randomly sampling from all successfully executed and evaluated individuals in the current population, while avoiding pairing two individuals with identical fitness values. This strategy aims to ensure performance differences between parents used for reflection, thereby providing meaningful comparison material for the LLM.

Let $P'_k(t) \subseteq P_k(t)$ denote the subset of valid individuals that successfully execute on the instance set $\mathcal{I}_t$. The parent selection process can be formulated as:

$$\begin{aligned} \{(h_a^{(i)}, h_b^{(i)})\} &\leftarrow \text{Select}(P'_k(t), N_c), \\ \text{s.t.} \quad \forall i,\ &Obj(h_a^{(i)}, \mathcal{I}_t) \neq Obj(h_b^{(i)}, \mathcal{I}_t) \end{aligned} \tag{5}$$

3.3.2 Short-term reflection and crossover

For each selected parent pair (h_a, h_b), the reflector LLM (LLM_{ref}) generates a short-term reflection R^{short}. This natural language text extracts improvement suggestions through comparative analysis of the performance differences, code structure, and intrinsic logic between the two parent rules. This reflection serves as a verbal gradient, providing semantic guidance beyond pure numerical fitness for evolutionary search.In this context, the "verbal gradient" refers to how the LLM uses the discrepancy in fitness between the two individuals to learn from their performance on the case and generate more effective prompts for future HDRs design.

Subsequently, this short-term reflection along with the parent pair is input into the generator LLM (LLM_{gen}). During the crossover step, the LLM generates a new set of heuristic strategies by combining user_generator, performance data of parent strategies, reflection, and detailed generation instructions. The LLM synthesizes the advantages of both parents according to the improvement suggestions in the reflection, generating a logically superior offspring individual $h_{\text{offspring}}$. This process is formalized as follows:

Assume $Obj(h_b, \mathcal{I}_t) < Obj(h_a, \mathcal{I}_t)$, indicating that h_b exhibits superior performance.

- Generate short-term reflection:

$$R_{k,t}^{short,(i)} = \text{LLM}_{\text{ref}}(\mathcal{T}_{\text{short}}, h_a, h_b, \text{“worse”}, \text{“better”}) \tag{6}$$

- Generate offspring:

$$h_{\text{offspring}} = \text{LLM}_{\text{gen}}(\mathcal{T}_{\text{cross}}, h_a, h_b, R^{\text{short}}) \tag{7}$$

This process is repeated for all N_c parent pairs, thereby generating an offspring set $P_{k,t}^{\text{cross}}$ containing N_c new individuals.

3.3.3 Accumulation of long-term reflection

After completing the crossover operations of one generation, the knowledge embodied in all generated short-term reflections is distilled and accumulated. The reflector LLM receives the long-term reflection $R_k^{\text{long}}(t-1)$ accumulated from the previous generation of this island, as well as all short-term reflections $\{R_{k,t}^{\text{short},(i)}\}_{i=1}^{N_c}$ generated in the current generation. The task of the LLM is to summarize, generalize, and refine this accumulated knowledge, generating a new and more comprehensive long-term reflection $R_k^{\text{long}}(t)$.

$$R_{k,t}^{\text{long}} = \text{LLM}_{\text{ref}}(\mathcal{T}_{\text{long}}, R_{k,t-1}^{\text{long}}, \{R_{k,t}^{\text{short},(i)}\}_{i=1}^{N_c}) \tag{8}$$

This process proceeds independently on each island, enabling each island to gradually form a unique and continuously

evolving knowledge base or expertise according to its own evolutionary trajectory and personalized orientation. Over time, the long-term reflections of different islands will diverge due to their distinct initial personalities and evolutionary experiences, which constitutes another key manifestation of how Multi-Island-ReEvo achieves multi-strategy parallel exploration.

3.3.4 Personalized guided elite mutation

The mutation operation represents one of the most significant differences between Multi-Island-ReEvo and the standard ReEvo framework. In Multi-Island-ReEvo, mutation is not merely a random perturbation introducing novelty, but rather a directional exploration conforming to the macro-strategy of the island.

The mutation operator acts on the elite individual of the current island, that is, the individual with the best objective value $h^*_{k,t} = \arg\min_{h \in \{P_k(t) \cup h^*_{k,t-1}\}} \text{Obj}(h)$. When the generator LLM generates a new mutated individual, its prompt not only includes the code of the elite individual and the latest long-term reflection of this island $R^{\text{long}}_k(t)$, but also specifically includes the personalized description of this island Π_k. The LLM is explicitly instructed that the generated mutated individual code needs to conform to the stated personalized description of the island. This mechanism fundamentally differs from GP mutation, which applies random perturbations without any directional guidance. Here, the combination of long-term reflection and island personality provides the LLM with both historical knowledge and strategic orientation, enabling purposeful structural modifications rather than blind random changes.

For instance, for an island with a conservative personality, the LLM might fine-tune and optimize based on the elite rule, while for an exploratory island, the LLM might introduce entirely new logical branches or heuristic factors, conducting bolder structural innovations. This mechanism ensures that mutation operations continuously strengthen the differentiation between islands.

The mutation process generates N_m new individuals, forming the set $P^{\text{mut}}_{k,t}$, with its formal representation as:

$$P^{\text{mut}}_{k,t} = \{\text{LLM}_{\text{gen}}(\mathcal{T}_{\text{mut}}, h^*_{k,t}, R^{\text{long}}_k(t), \Pi_k)\}_{i=1}^{N_m} \quad (9)$$

The parameter Π_k in the formula explicitly embodies this core innovation of personalized guidance.

3.3.5 Population update

After crossover and mutation operations produce new offspring individuals, population update is performed to form the next generation population $P_k(t+1)$.

$$P_k(t+1) = P^{\text{cross}}_{k,t} \cup P^{\text{mut}}_{k,t} \quad (10)$$

And the elite individual are also refreshed accordingly.

3.4 Individual evaluation

After each crossover or mutation operation produces a new heuristic rule, the system evaluates its performance. The detailed process is shown in supplementary material Fig. S5. Each newly generated rule (that is, a Python function) is executed by the system on a set of DHFSP problem instances to determine the processing sequence and machine allocations for jobs in these instances. The system records and calculates the average makespan across all instances, which is used as objective value $Obj(h, \mathcal{I}_t)$. A shorter average makespan represents higher individual fitness. This evaluation result serves as the core basis for driving population evolution, pushing the population toward superior solutions.

3.5 Migration

To facilitate the exchange of superior genes among different islands and prevent any independent sub-population from premature convergence to a local optimum, the system periodically executes migration operations.

3.5.1 Migration trigger and topology structure

Migration operations do not occur in every generation but are triggered periodically according to a preset migration interval τ_{mig}. For example, when the global iteration count t satisfies $(t \bmod \tau_{mig} = 0)$, the migration process is initiated.

This method adopts a bidirectional ring topology structure to organize inter-island communication. This is one of the most popular migration topologies, due to its simplicity and effectiveness [51]. In this structure, all islands are arranged in a ring where each island S_k is connected to its left neighbor $S_{(k-1) \bmod N_{\text{isl}}}$ and right neighbor $S_{(k+1) \bmod N_{\text{isl}}}$. During each migration event, an island randomly selects one of its two neighbors as the partner for individual exchange. The bidirectional ring topology ensures that any superior genes (that is, high-performance rule fragments) generated on any island eventually have an opportunity to propagate to all other islands in the network through step-by-step transmission, thereby avoiding the formation of information islands and ensuring sufficient gene propagation.

Algorithm 3 Ring topology best-worst migration

Require: Array of N islands Isl, migration rate R_m
Ensure: Islands Isl after migration

for $i \leftarrow 1$ **to** N **in parallel do**
  $src \leftarrow i, tgt \leftarrow (i + \text{Random}(\{-1, 1\})) \bmod N$
  $n_{mig} \leftarrow \lfloor |Isl_{[src]}.pop| \times R_m \rfloor$
  $P_{best} \leftarrow$ Top n_{mig} from $Isl_{[src]}.pop$
  $P_{worst} \leftarrow$ Bottom n_{mig} from $Isl_{[tgt]}.pop$
  Replace P_{worst} with P_{best} in $Isl[tgt].pop$
  $Isl_{[tgt]}.elite \leftarrow \text{Update}(Isl_{[tgt]}.pop, Isl_{[tgt]}.elite)$
end for
return Isl

3.5.2 *Best-worst migration strategy*

To maximize the positive impact of migration operations on the entire evolutionary process, this method adopts the best-worst migration strategy. This strategy accelerates global population optimization through a dual-action mechanism [52]:

- **Selection of migrating individuals**: The source island S_k as sender selects the top $\lceil \rho_{mig} \cdot N_p \rceil$ best individuals from its current population $P_k(t)$ based on fitness values, forming the elite pool M_k^{out} to be migrated out. This operation can rapidly inject validated high-performance solutions into other populations.

$$M_k^{\text{out}} = \text{TopK}(P_k(t), \lceil \rho_{\text{mig}} \cdot N_p \rceil, \text{key} = -\text{Obj}) \quad (11)$$

- **Selection of destination**: From the two neighbors of island S_k, randomly select one as the destination island S_j.

$$j \leftarrow \text{RandomChoice}(\{(k-1) \bmod N_{\text{isl}}, (k+1) \bmod N_{\text{isl}}\}) \quad (12)$$

- **Replacement of individuals**: In the receiving island S_j, the elite pool M_k^{out} from S_k replaces the worst $\lceil \rho_{mig} \cdot N_p \rceil$ individuals in population P_j. This operation not only creates space for the integration of superior genes but also actively removes inefficient solutions from the population, improving the average fitness of the population.

$$P_j(t) \leftarrow \left(P_j(t) \setminus \text{WorstK}(P_j(t), \lceil \rho_{\text{mig}} \cdot N_p \rceil, \text{key} = -\text{Obj})\right) \cup M_k^{\text{out}} \quad (13)$$

The effectiveness of this strategy is further enhanced in Multi-Island-ReEvo. Because each island has a distinct personality, its evolved best individuals are likely to differ significantly in semantics and logic. Therefore, when an elite individual from an exploratory island migrates to a conservative island, it brings not only a high fitness value but also a completely new problem-solving approach that the island itself would have difficulty discovering, thereby achieving efficient strategy crossover and fusion.

4 LLM-based framework for DHFSP heuristic rules generation

This paper proposes an innovative framework that leverages LLMs to generate efficient heuristic rules for DHFSP. The framework is inspired by how human experts approach complex problems through learning from experience and applying knowledge in practice. Its core lies in the organic integration of two phases, namely offline training and online application, aiming to discover and utilize high-quality scheduling rules through automated means.The overall framework is detailed illustrated in Fig 2.

During the offline training phase, the primary objective is to autonomously evolve and optimize heuristic rules through continuous iteration and learning in a simulated environment. To ensure comprehensive and robust training, we construct a training set encompassing diverse scenarios. This set comprises three types of problem instances. The first type includes regular production scenarios with only dynamic order arrivals. The second type encompasses scenarios with only stochastic machine faults. The third type involves complex dynamic scenarios where new order arrivals and machine faults occur simultaneously. In each training iteration, the system randomly selects one case from each of these three scenario types and additionally selects one more case, forming a training batch of four problem instances. Subsequently, the LLM generates a set of candidate HDRs for this batch of cases. By evaluating the performance of these rules on the current batch in the simulation environment with mean completion time as the primary evaluation metric, the system identifies the optimal rules. For the same batch of cases, the generated rules undergo multiple iterations. Each iteration reflects on and improves upon the best-performing rules produced previously, thereby driving the rule population toward higher performance levels. In our experiments, a total of 20 batches of cases are sampled to ensure that the final HDRs possess sufficient generalization capability and superior performance.

In the subsequent online application phase, the framework utilizes the optimal HDRs evolved during the training phase to provide rapid and accurate decision support for real-world scheduling problems. When confronting new practical production scheduling tasks, the system no longer requires multiple rounds of evolutionary search. Instead, it employs the thoroughly trained optimal HDRs as prior knowledge and integrates them with specific information from the current practical scenario, such as real-time arriving orders and current machine status. Through a single iteration, it rapidly

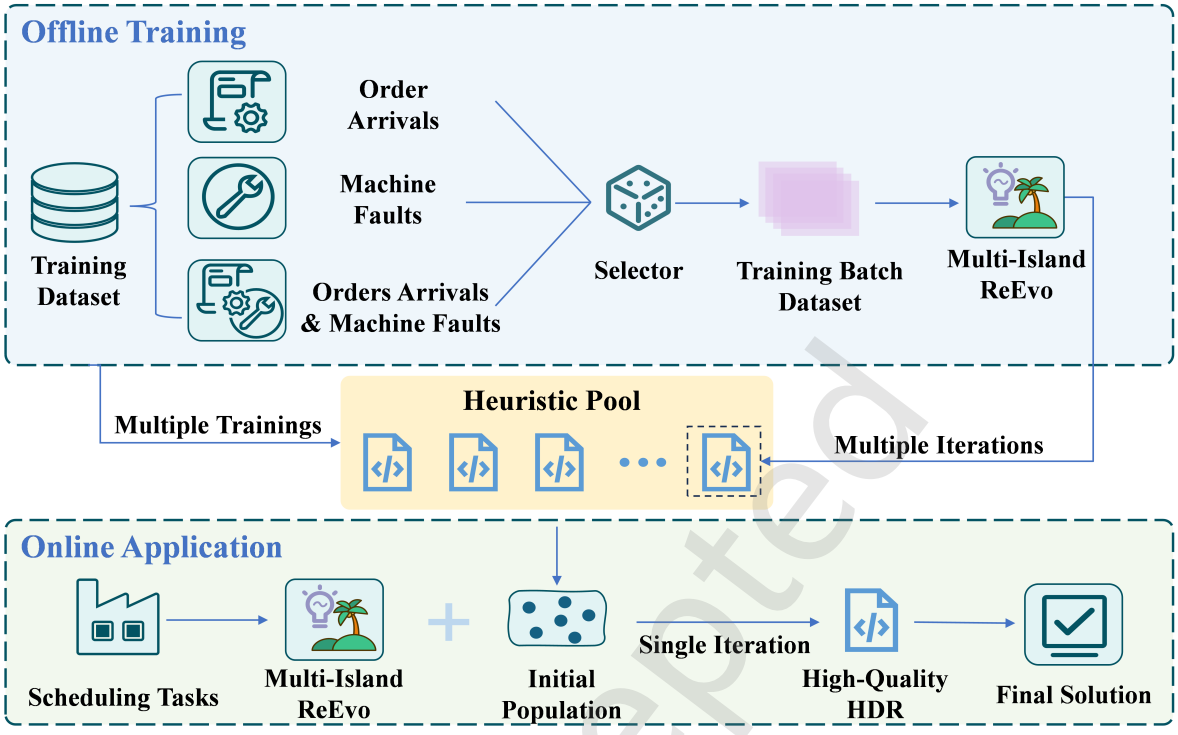

Fig. 2 The overall framework for HDR training and application.

generates a high-quality heuristic rule tailored to the current scenario. This design substantially reduces decision-making time, enabling the framework to meet the stringent real-time requirements of industrial production.

5 Numerical experimental and analysis

To systematically evaluate the effectiveness and superiority of the proposed Multi-Island-ReEvo method in solving the DHFSP, we designed a series of comparative experiments and ablation studies. These experiments aim to simulate dynamic hybrid flow shop environments under uncertainty and verify the decision quality and robustness of our method in complex scheduling scenarios through performance comparisons with existing automated algorithm design methods, several LLM-based frameworks, and classical HDRs.

5.1 Experimental environment and parameter configuration

To comprehensively evaluate the proposed method, a three-tiered comparison framework is established. The experimental evaluation employs DHFSP instances featuring dynamic random order arrivals and machine failures, covering three distinct scenario categories. The comparison framework includes two categories of baseline methods. First, two state-of-the-art evolutionary heuristic generation approaches from dynamic flexible job shop scheduling (DFJSP) are adapted: the GEP algorithm [59], and multi-tree genetic programming (MTGP) algorithm [60], which employs separate trees for job sequencing and machine selection. Both DFJSP and DHFSP share the two-stage decision structure of job selection followed by machine assignment, making these methods directly applicable with appropriate adaptations. Second, 12 classical HDRs serve as baselines, combining 6 job sequencing rules and 2 machine selection rules (Table 1), to assess whether automatically generated HDRs surpass hand-crafted ones.

The Multi-Island-ReEvo method adopts unified parameter configurations across all phases. Three LLM APIs serve as both generators and reflectors: qwen-turbo-latest, deepseek-v3, and gpt-4o-mini, with temperature set to 1.0. The framework consists of 3 islands, each maintaining 20 individuals, with 52 iterations and mutation rate of 0.5. Migration occurs every 5 iterations with a rate of 0.2. Due to space constraints, detailed parameter configurations for GEP and MTGP are provided in Tables S1 and S2 of the supplementary materials. All experiments are implemented in Python on a server with Intel Xeon Platinum 8352V CPU at 2.10GHz running Ubuntu 22.04.

The training set comprises 120 problem instances equally distributed across three scenario categories (detailed in Section 4), with 40 instances per category. The problem instances are generated according to the following specifications. For

Table 1 Two types of dispatching rules

Rule	Category	Content
SPT_J	Job sequencing rule	Jobs with the smallest processing time at the current operation have the highest priority
LPT_J	Job sequencing rule	Jobs with the largest processing time at the current operation have the highest priority
SRT_J	Job sequencing rule	Jobs with the smallest remaining processing time have the highest priority
LRT_J	Job sequencing rule	Jobs with the largest remaining processing time have the highest priority
SSO_J	Job sequencing rule	Jobs with the smallest processing time at the subsequent operation have the highest priority
LSO_J	Job sequencing rule	Jobs with the largest processing time at the subsequent operation have the highest priority
SPT_M	Machine selection rule	Prioritize selecting the machine with the shortest processing time
NLR_M	Machine selection rule	Prioritize selecting the machine with the smallest load rate

multi-batch order scenarios, each batch contains a random number of jobs between 15 and 20, with the total number of batches ranging from 5 to 8. The arrival time of each batch follows a uniform distribution over the interval [1, 2000]. For single-batch scenarios, the order size ranges from 100 to 120 jobs. In scenarios with machine failures, the number of failure events varies from 2 to 5. Each failure event has a start time uniformly distributed in [1, 2900] and an end time uniformly distributed in [failure start time + 50, 3000]. Across all scenarios, the number of machines ranges from 12 to 20, and the number of processing stages exceeds 10. Machines are randomly assigned to stages while ensuring that at least one stage contains two or more parallel machines. The processing time for each job on a given machine is randomly generated within [10, 40]. During each training iteration, the system randomly selects one instance from each of the three scenario categories, plus one additional random instance, forming a training batch of four problem instances.

5.2 Performance verification under dynamic environment cases

This section evaluates the performance of HDRs generated by Multi-Island-ReEvo in handling uncertainty. The test set contains 200 new instances generated using the same methodology as training but entirely unseen during training: 100 instances with dynamic order arrivals and 100 with machine failures.

To eliminate the dimensional differences in absolute makespan values across instances of varying scales and to fairly evaluate the relative performance of each algorithm, we adopt the Gap Ratio as the primary evaluation metric.Let $\mathcal{M}$ denote the set of all comparative algorithms (including the proposed method and all baselines), and let $\mathcal{I}$ represent the set of test instances. For any given test instance $i \in \mathcal{I}$, let $C_{i,m}$ denote the objective function value (i.e., Makespan) obtained by algorithm $m \in \mathcal{M}$.First, the Best Solution, denoted as C_i^*, is defined as the minimum makespan found by any algorithm in the comparison set for instance i:

$$C_i^* = \min_{m \in \mathcal{M}} \{C_{i,m}\} \tag{14}$$

Based on this, the Gap Ratio for algorithm m on instance i is defined as follows:

$$\text{Gap}_{i,m} = \frac{C_{i,m} - C_i^*}{C_i^*} \tag{15}$$

As illustrated in Fig. 3, Multi-Island-ReEvo demonstrates significant performance advantages across all three LLM variants when handling dynamic order arrivals. The median gap ratio approaches zero with substantially more compact interquartile ranges (IQR) than competing methods, indicating excellent average performance and high consistency across diverse instances. Notably, the maximum gap ratio for all three variants remains below 2%, demonstrating exceptional stability. In contrast, classical dispatching rules and evolutionary methods (GEP and MTGP) exhibit considerable performance fluctuations. The fixed nature of classical rules limits their adaptability, while GEP and MTGP struggle to maintain consistent quality despite their evolutionary capabilities. Multi-Island-ReEvo effectively addresses these limitations by leveraging LLM-guided evolution to generate adaptive HDRs that absorb uncertainty from demand fluctuations.

As shown in Fig. 4, Multi-Island-ReEvo maintains its competitive advantage under machine failures with even stronger robustness. The median and lower quartile for all three variants converge tightly to zero, consistently identifying best or near-best solutions. Among variants, deepseek-v3 and gpt-4o-

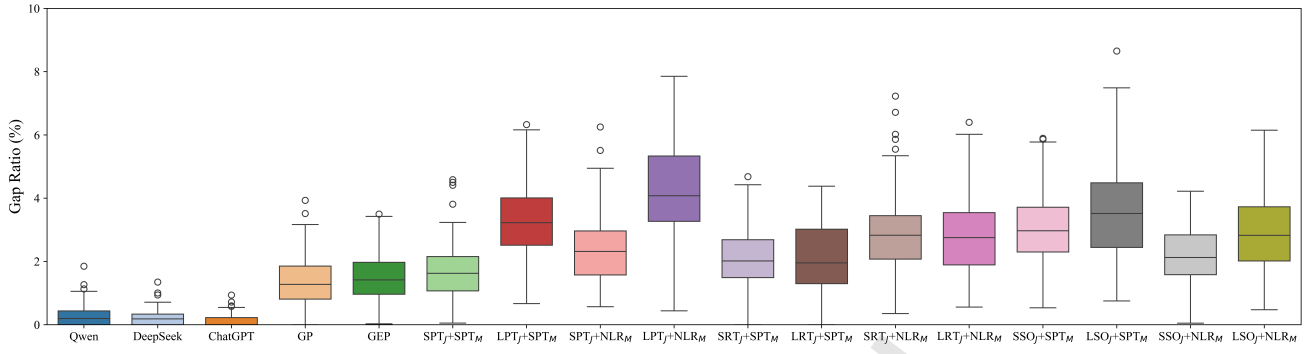

Fig. 3 Gap Ratio of 17 methods in cases with only random order arrivals.

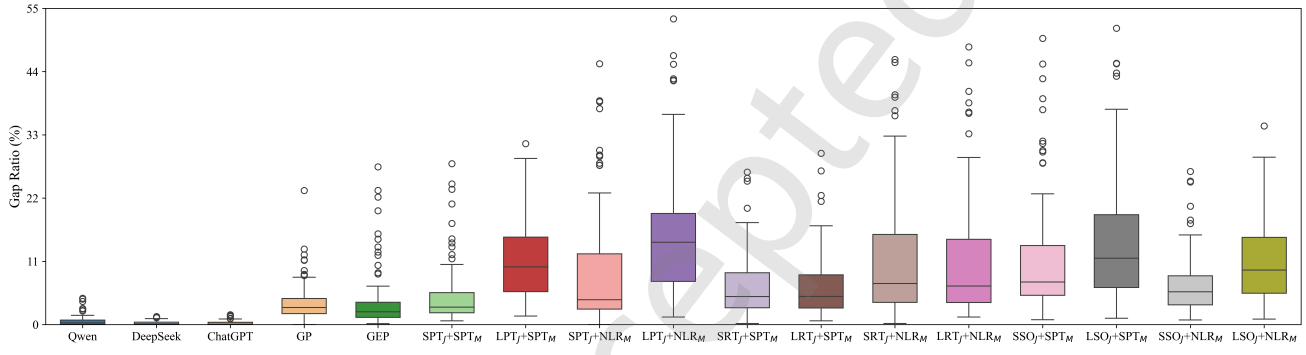

Fig. 4 Gap Ratio of 17 methods in cases with only machine failures.

mini exhibit particularly strong performance with extremely compact boxes, while qwen-turbo-latest, though slightly inferior, still substantially exceeds other benchmarks. GEP and MTGP demonstrate reasonable performance in certain instances but show larger fluctuations and reduced stability through extended boxes and increased outliers. Classical dispatching rules exhibit generally poor performance with higher median values and dispersed distributions. These observations highlight that rules evolved by our framework possess superior adaptability when handling continuous disturbances.

To further quantify performance differences, Fig. 5 presents statistics on best solution frequency and average makespans across 100 instances in both scenarios. For dynamic order arrivals (left panel), the three LLM variants exhibit similar average makespans (3561.6 to 3567.0). In machine failures (right panel), gpt-4o-mini and deepseek-v3 maintain nearly identical performance at approximately 4454, while qwen-turbo-latest shows a slightly higher average of 4470.2, suggesting greater challenges with resource disruptions. In contrast, GEP and MTGP rarely obtain best solutions and exhibit significantly deteriorated average performance, exceeding 4611 compared to approximately 4454 for LLM-based methods. This performance gap underscores the fundamental advantage of the language-heuristic paradigm in capturing generalizable scheduling principles. It should be noted that the "best solution" refers to the best result among all compared methods for each instance, rather than the theoretical optimal solution.

Notably, in a small number of instances, GEP, MTGP, or classical HDRs obtain the best solution rather than Multi-Island-ReEvo. However, our method remains very close to optimal, evidenced by extremely low maximum gap ratios and compact distributions, maintaining a consistently high performance lower bound—a characteristic particularly valuable where reliability and consistency matter as much as peak performance.

The superior performance stems from two complementary mechanisms. First, the language-heuristic paradigm achieves deep integration between LLM semantic understanding and scheduling logic. Through exposure to diverse instances with various disturbances, the framework guides LLMs to extract universal scheduling principles transcending specific problem characteristics, producing HDRs as knowledge assets encoding generalized wisdom for robust performance. Second, the personalized multi-island architecture and reflective evolution jointly construct an efficient search paradigm. The multi-island structure maintains search breadth through multiple populations with distinct preferences, preventing premature convergence, while reflective evolution provides semantic-

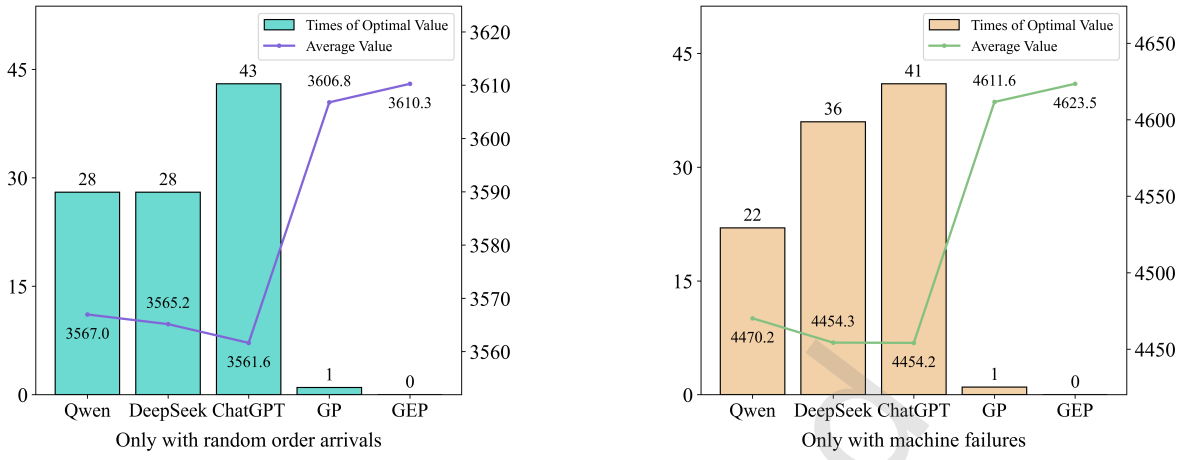

Fig. 5 Comparison between 3 LLMs methods, GP and GEP.

level guidance through LLM reasoning, ensuring search depth and efficiency. This synergy guarantees high quality and diversity in generated HDRs, explaining consistent near-best performance even without finding the absolute optimum.

5.3 Ablation study

To validate the design choices underlying the Multi-Island-ReEvo architecture, comprehensive ablation studies are conducted comparing our approach with two representative LLM-based evolutionary frameworks: ReEvo[21] and HsEvo[24]. ReEvo employs a single-population reflective evolution mechanism, effectively removing the multi-island architecture. HsEvo builds upon ReEvo but replaces the reflective evolution mechanism with harmony search and rapid reflection strategies, with parameters configured according to the original literature[24]. All three methods are trained on identical problem instances from the training set in Section 4.1. The test set comprises 100 instances featuring simultaneous dynamic order arrivals and machine failures, representing the most challenging scenario where multiple sources of uncertainty interact. These instances are generated using the same methodology but remain completely unseen during training.

The radar charts in Fig. 6 compare the three methods across two dimensions: average solution quality and optimization capability. In the left chart depicting average makespan (lower values indicate better performance), Multi-Island-ReEvo driven by all three LLMs (axes G, H, and I) exhibits indicators in the innermost region with the lowest average values, demonstrating superior performance across 100 test instances. The right chart illustrates the total number of obtaining best solutions (higher values reflect stronger optimization). Multi-Island-ReEvo consistently obtains the highest number of best solutions across all LLM variants, with data points occupying the outermost periphery, highlighting its substantially enhanced capability compared to simplified frameworks.

To obtain a more comprehensive understanding of performance distribution and solution stability, violin plots are examined in Figure 7. Methods A through C correspond to ReEvo variants, D through F represent HsEvo variants, and G through I denote Multi-Island-ReEvo variants, each paired respectively with qwen-turbo-latest, deepseek-v3, and gpt-4o-mini. The Multi-Island-ReEvo methods (G, H, I) display notably narrow and concentrated distributions, with the vast majority of observations tightly clustered around a zero gap ratio, indicating a high degree of consistency in generating superior-quality solutions. Variants H and I, in particular, exhibit exceptional performance, with distributions almost exclusively confined to the near-zero region, demonstrating their capacity to reliably produce optimal or near-optimal solutions irrespective of instance characteristics. In contrast, both ReEvo (A, B, C) and HsEvo (D, E, F) show considerably broader distributions accompanied by pronounced tails extending toward higher gap ratios. Certain instances reveal gap ratios as high as 20% to 30%, reflecting substantial deviations from the best attainable performance. This variability suggests that these simplified frameworks experience greater performance volatility and are more prone to yielding suboptimal solutions when faced with complex instances. Absent the multi-island architecture or alternative search strategies, these methods lack the robustness necessary to sustain consistent performance across a diverse range of scheduling scenarios.

The violin plot analysis corroborates our earlier findings while providing a more nuanced understanding of performance stability. It reveals that the superiority of Multi-Island-ReEvo extends beyond merely achieving better average perfor-

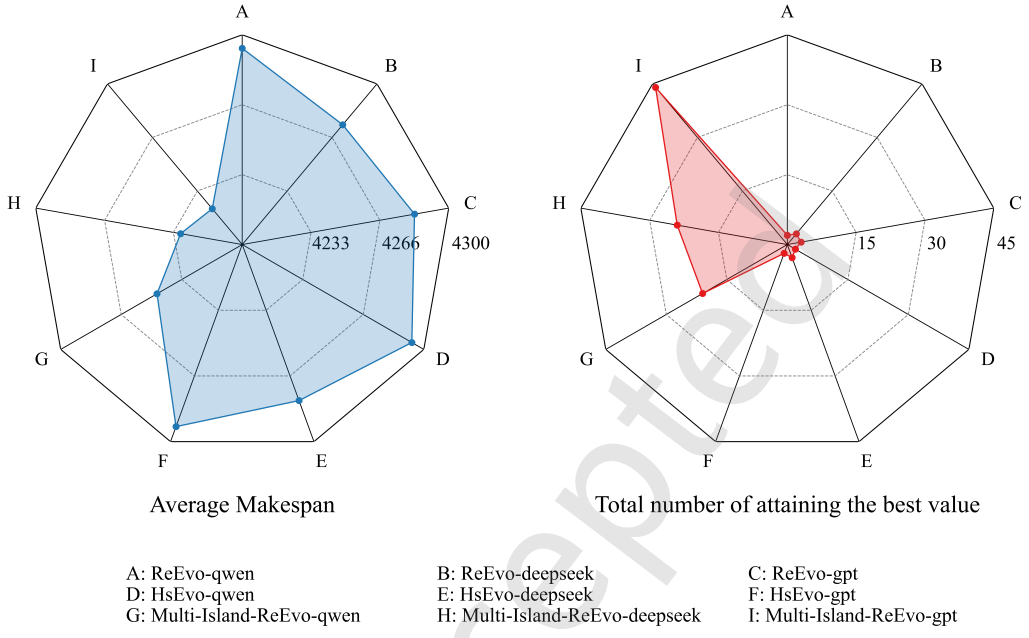

Fig. 6 Comparison of average makespan and best solution frequency across 3 methods on different LLMs.

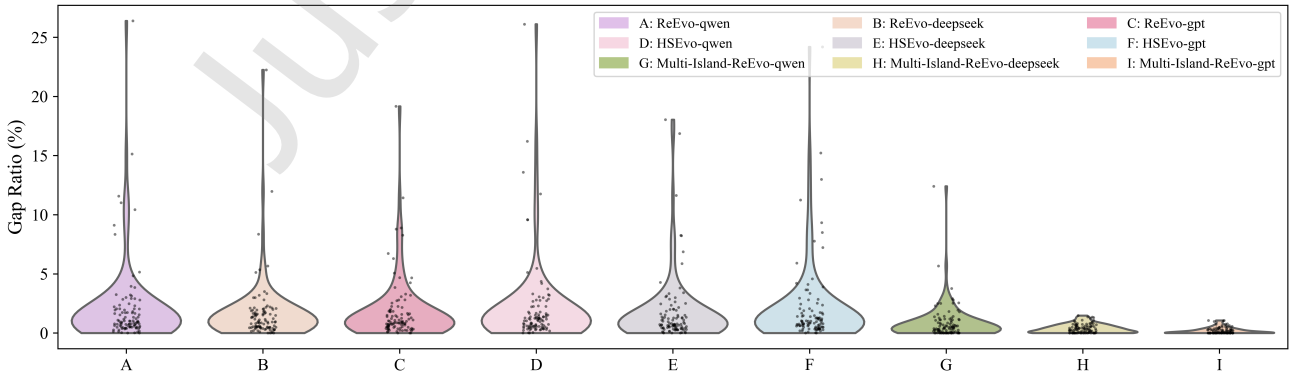

Fig. 7 Gap Ratio of different methods in complicated dynamic cases.

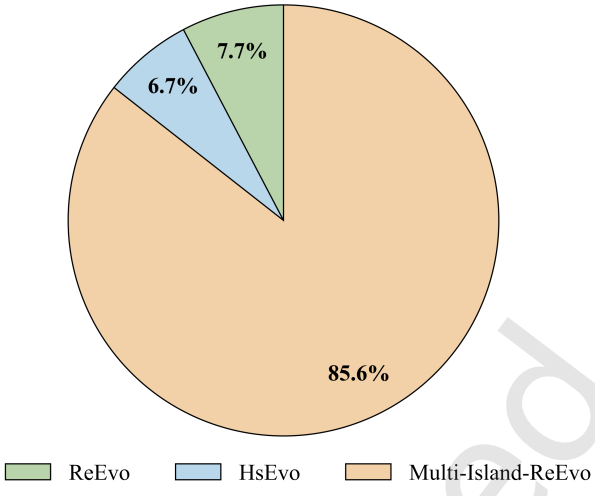

Fig. 8 The percent of 3 methods attaining the best value.

mance or obtaining more best solutions. Rather, the method demonstrates fundamentally stronger robustness by maintaining consistently high solution quality across diverse problem instances, effectively eliminating the performance variability that plagues alternative approaches. This stability is particularly crucial in practical manufacturing environments where reliable performance is as valuable as peak performance.

The pie chart in Fig. 8 provides quantitative assessment of different methods' contributions to discovering best solutions. Multi-Island-ReEvo contributes 85.6% of all best solutions, while ReEvo and HsEvo each contribute less than 8%. This notable difference provides compelling evidence that the personalized multi-island mechanism plays a pivotal role in enhancing performance. The architecture's ability to maintain multiple diverse populations with distinct exploration strategies effectively prevents premature convergence and enables more thorough solution space exploration, substantially improving the likelihood of discovering high-quality solutions.

After verifying the superiority of Multi-Island-ReEvo, the performance characteristics of different LLMs within each framework are investigated. Fig. 9 presents three pie charts illustrating the distribution of best solutions obtained by the three LLMs under each method. The results reveal consistent patterns across frameworks. Within both Multi-Island-ReEvo and ReEvo, gpt-4o-mini (referred to as ChatGPT in the figure) obtains the highest number of best solutions, demonstrating superior performance in guiding evolutionary search. DeepSeek follows closely, exhibiting strength slightly exceeding Qwen.

This performance ranking aligns well with recent comprehensive evaluations of these models on code generation and logical reasoning benchmarks. GPT-4o-mini typically excels in tasks requiring complex reasoning, nuanced understanding of problem constraints, and generation of sophisticated code structures. DeepSeek has also demonstrated strong competitiveness in code-related tasks, particularly in understanding algorithmic logic and generating efficient implementations. These capabilities are directly relevant to the task of generating effective HDRs, which requires both understanding the scheduling problem structure and synthesizing executable code that captures optimal decision-making logic. The consistency between LLM performance in our scheduling domain and their general capabilities in reasoning and code generation suggests that the language-heuristic paradigm effectively leverages the intrinsic strengths of these foundation models.

A representative HDR example generated by our method is shown in Fig 10. This heuristic rule exemplifies an "exploration" evolutionary strategy by integrating composite decision logic with controlled stochasticity. For job sequencing, it employs a look-ahead LPT approach normalized by total workload to clear bottlenecks, while machine selection utilizes a threshold-triggered mechanism that prioritizes speed unless high load dictates balancing. Critically, the rule's "exploration" personality is manifested through the injection of a random "performance_factor" and adaptive weighting at the output layer. This stochastic perturbation is not mere noise but a strategic mechanism to break deterministic deadlocks. By forcing diverse decision paths under similar states, the algorithm maintains robustness against dynamic fluctuations, ensuring it avoids stagnation in local optima—a hallmark of an explorative search policy.

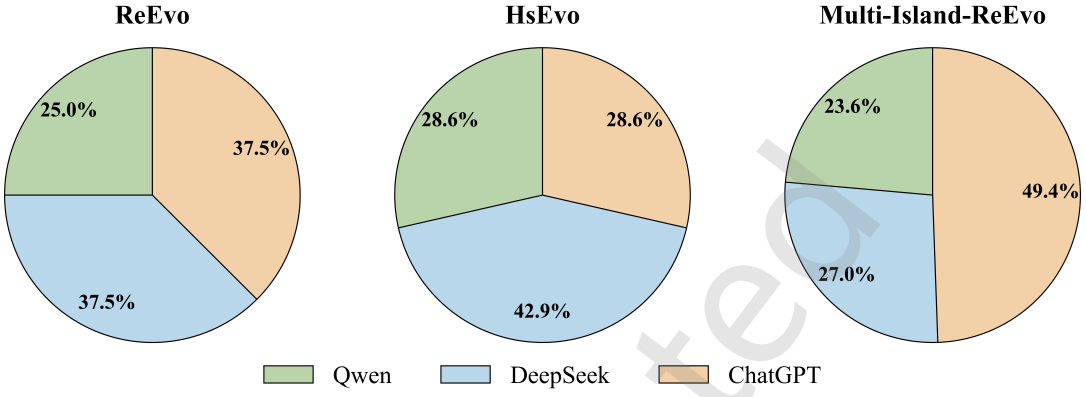

Fig. 9 Distribution of best solutions obtained by three LLMs within each evolutionary framework.


```
import numpy as np

def get_combined_expression_v2(pt=None, wkr=None, rm=None, so=None, twk=None, ptm=None,
ur=None, select_composite=True) -> np.ndarray:
    if select_composite:
        mean_wkr = np.mean(wkr[wkr > 0]) if wkr is not None else 1
        combined_expr = (pt * 1.5 + so + mean_wkr) / (1 + np.log1p(twk)) + (rm + 1) / (mean_wkr + 1)
    else:
        adjusted_ur = np.clip(ur, 0.7, None)
        combined_expr = ptm / (adjusted_ur + 1e-5)

    # Dynamic adjustments based on real-time performance
    performance_factor = np.random.uniform(0.8, 1.2)  # Adaptive scaling
    adaptive_weights = 1 / (1 + np.log1p(np.sum(combined_expr) + 1e-5))
    return np.clip(combined_expr * adaptive_weights * performance_factor, 0, None)
```

Fig. 10 One HDR generated by the ChatGPT under the personality of “exploration”.

Beyond performance, analyzing the generated HDRs in Fig10 reveals the transformative value of LLMs in algorithm design. Human-designed rules rely on domain expertise but utilize rigid logic lacking adaptability in complex dynamic scenarios. Conversely, GP/GEP methods generate adaptive but notoriously uninterpretable mathematical "black boxes". LLMs bridge this gap by shifting from purely mathematical combinations to human-readable programmatic logic. By naturally employing explicit conditional branching (if/else), intermediate variables, and contextual mechanisms (e.g., adaptive scaling), LLM-designed methods synthesize comprehensive policies. They uniquely combine the sophisticated adaptability of evolutionary search with the transparent interpretability of human engineering.

6 Conclusion and Future Work

This paper addresses the complex DHFSP by proposing Multi-Island-ReEvo, an LLM-driven automated heuristic design framework that integrates a personalized multi-island architecture with reflective evolution mechanisms. The framework leverages LLM capabilities to establish diverse evolutionary objectives for parallel populations and guide genetic operators through natural language feedback. Extensive experiments validate the method's exceptional performance across diverse dynamic scenarios. When confronted with random order arrivals and machine failures, the generated HDRs significantly outperform those from classical evolutionary algorithms (GEP and MTGP) and widely used hand-crafted rules, achieving superior average performance with remarkable stability. Ablation studies quantitatively demonstrate that the multi-island mechanism is indispensable: Multi-Island-ReEvo contributes 85.6% of all best solutions compared to single-population variants (ReEvo and HsEvo), validating its critical role in maintaining diversity and preventing premature convergence.

However, certain limitations exist. First, algorithm performance depends on the underlying LLM capabilities and prompt engineering quality. Second, API invocations introduce computational costs and potential latency, though the framework demonstrates practical viability through offline training of heuristics for efficient online deployment. Future research directions include: (1) investigating more efficient prompting strategies to harness LLM potential while reducing computational requirements; (2) extending the framework to multi-objective DHFSP scenarios, such as concurrently optimizing makespan and total tardiness; (3) integrating domain-specific scheduling knowledge into prompt design to enhance heuristic quality; and (4) exploring locally deployed open-source LLMs as substitutes for commercial APIs to reduce operational costs and address data privacy concerns in industrial applications.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant 52305534, the Strategic Science and Technology Talent Training Program of Hubei Province under Grant No 2024DJA033, Sichuan Province Engineering Technology Research Center of Broadband Electronics Intelligent Manufacturing, Grant BEIM2024B003, and the Fundamental Research Funds for the Central Universities under Grant No 2024BRA004.

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

[1] Y. Yao, Q. Liu, X. Li, and L. Gao, Constraint programming for agv and machine integrated scheduling problem in flexible manufacturing system, *IEEE Transactions on Automation Science and Engineering*, vol. 23, pp. 2378–2390, 2026.

[2] Q. Liu, X. Li, and L. Gao, A novel milp model based on the topology of a network graph for process planning in an intelligent manufacturing system, *Engineering*, vol. 7, no. 6, pp. 807–817, 2021.

[3] F. Liu, X. Li, C. Lu, and W. Gong, Adaptive knowledge-based multi-objective evolutionary algorithm for hybrid flow shop scheduling problems with multiple parallel batch processing stages, *Swarm and Evolutionary Computation*, vol. 95, p. 101929, 2025.

[4] C. Lu, L. Gao, J. Yi, and X. Li, Energy-efficient scheduling of distributed flow shop with heterogeneous factories: A real-world case from automobile industry in china, *IEEE Transactions on Industrial Informatics*, vol. 17, no. 10, pp. 6687–6696, 2020.

[5] F. Zhao, J. Zhou, J. Lv, and Y. Du, A proximal policy optimization-guided general co-evolution search framework for distributed flexible assembly flowshop scheduling problem with sequence-independent setup times, *Expert Systems with Applications*, vol. 314, p. 131757, 2026.

[6] K. Peng, X. Deng, C. Zhang, Q.-K. Pan, L. Ren, and X. Pang, An improved imperialist competitive algorithm for hybrid flowshop rescheduling in steelmaking-refining-continuous casting process, *Meas. Control*, vol. 53, no. 9-10, pp. 1920–1928, 2020.

[7] X.-R. Tao, Q.-K. Pan, X.-L. Jing, and W.-M. Li, A dynamic multi-objective evolutionary greedy algorithm for distributed hybrid flow shop rescheduling problem, *Swarm and Evolutionary Computation*, vol. 97, p. 102054, 2025.

[8] J. Huang, X. Li, Q. Liu, and L. Gao, Efficient scheduling for fixed-type multi-robot collaborative problem in flexible job shop, *Robotics and Computer-Integrated Manufacturing*, vol. 98, p. 103157, 2026.

[9] L. Deng, Y. Qiu, W. Gong, Y. Di, and C. Li, A dynamic decision-driven memetic algorithm for fuzzy distributed hybrid flow shop rescheduling considering quality control, *Expert Systems with Applications*, vol. 257, p. 125002, 2024.

[10] K. Wang, S. H. Choi, H. Qin, and Y. Huang, A cluster-based scheduling model using spt and sa for dynamic hybrid flow shop problems, *The International Journal of Advanced Manufacturing Technology*, vol. 67, no. 9, pp. 2243–2258, 2013.

[11] L. Meng, K. Gao, Y. Ren, B. Zhang, H. Sang, and Z. Chaoyong, Novel milp and cp models for distributed hybrid flowshop scheduling problem with sequence-dependent setup times, *Swarm and Evolutionary Computation*, vol. 71, p. 101058, 2022.

[12] C. Wang, Q.-K. Pan, and H.-Y. Sang, The cascaded flow-shop joint scheduling problem: A mathematical model and population-based iterated greedy algorithm to minimize total tardiness, *Robotics and Computer-Integrated Manufacturing*, vol. 88, p. 102747, 2024.

[13] W. Wang, B. Zhang, X. Jiang, B. Jia, H. Sang, and L. Meng, Decomposition-based multi-objective approach for a green hybrid flowshop rescheduling problem with consistent sublots, *International Journal of Production Research*, vol. 62, no. 21, pp. 7904–7932, 2024.

[14] X. Chen, J. Li, Z. Wang, Q. Chen, K. Gao, and Q. Pan, Optimizing dynamic flexible job shop scheduling using an evolutionary multi-task optimization framework and genetic programming, *IEEE Transactions on Evolutionary Computation*, 2025.

[15] F. Zhao, C. Zhao, L. Wang, and H. Sang, A deep reinforcement learning framework assisted by genetic programming for dynamic flexible job shop scheduling, *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2025.

[16] X. Wu, Z. Cao, and S. Wu, Real-time hybrid flow shop scheduling approach in smart manufacturing environment, *Complex System Modeling and Simulation*, vol. 1, no. 4, pp. 335–350, 2021.

[17] F. Liu, X. Tong, M. Yuan, X. Lin, F. Luo, Z. Wang, Z. Lu, and Q. Zhang, Evolution of heuristics: Towards efficient automatic algorithm design using large language model, *41st International Conference on Machine Learning (ICML 2024)*, pp. 32 201–32 223, 2024.

[18] B. Romera-Paredes, M. Barekatain, A. Novikov, M. Balog, M. P. Kumar, E. Dupont, F. J. Ruiz, J. S. Ellenberg, P. Wang, O. Fawzi *et al.*, Mathematical discoveries from program search with large language models, *Nature*, vol. 625, no. 7995, pp. 468–475, 2024.

[19] F. Zhao, B. Wu, L. Wang, and H. Sang, A large language model-assisted reinforcement learning framework with evolutionary algorithm for hybrid flow-shop scheduling, *IEEE Transactions on Evolutionary Computation*, 2026.

[20] R. Li, L. Wang, H. Sang, L. Yao, and L. Pan, Llm-assisted automatic memetic algorithm for lot-streaming hybrid job shop scheduling with variable sublots, *IEEE Transactions on Evolutionary Computation*, 2025.

[21] H. Ye, J. Wang, Z. Cao, F. Berto, C. Hua, H. Kim, J. Park, and G. Song, Reevo: Large language models as hyper-heuristics with reflective evolution, *Advances in neural information processing systems*, vol. 37, pp. 43 571–43 608, 2024.

[22] F. Yu, L. Gao, X. Li, C. Lu, and Q. Liu, Automated scheduling heuristic generation and evaluation via large language model, *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2026.

[23] J. Huang, Q. Liu, X. Li, L. Gao, and Y. Teng, Automatic programming via large language models with population self-evolution for dynamic fuzzy job shop scheduling problem, *IEEE Transactions on Fuzzy Systems*, 2026.

[24] P. V. T. Dat, L. Doan, and H. T. T. Binh, Hsevo: Elevating automatic heuristic design with diversity-driven harmony search and genetic algorithm using llms, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, 2025, pp. 26 931–26 938.

[25] J.-J. Wang and L. Wang, A cooperative memetic algorithm with learning-based agent for energy-aware distributed hybrid flow-shop scheduling, *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 3, pp. 461–475, 2021.

[26] R. Li, W. Gong, L. Wang, C. Lu, Z. Pan, and X. Zhuang, Double dqn-based coevolution for green distributed heterogeneous hybrid flowshop scheduling with multiple priorities of jobs, *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 4, pp. 6550–6562, 2023.

[27] J. Chen, M. Wang, X. T. Kong, G. Q. Huang, Q. Dai, and G. Shi, Manufacturing synchronization in a hybrid flowshop with dynamic order arrivals, *Journal of Intelligent Manufacturing and Special Equipment*, vol. 30, no. 7, pp. 2659–2668, 2019.

[28] Y. Chen, J. Zhang, M. Rauf, J. Mumtaz, and S. Huang, Dynamic scheduling of hybrid flow shop problem with uncertain process time and flexible maintenance using neuroevolution of augmenting topologies, *IET Collaborative Intelligent Manufacturing*, vol. 6, no. 3, p. e12119, 2024.

[29] K. Tliba, T. M. Diallo, O. Penas, R. Ben Khalifa, N. Ben Yahia, and J.-Y. Choley, Digital twin-driven dynamic scheduling of a hybrid flow shop, *JOURNAL OF INTELLIGENT MANUFACTURING*, vol. 34, no. 5, pp. 2281–2306, 2023.

[30] Y. Zuo, F. Zhao, and J. Zhang, A bi-population cooperative scatter search algorithm for distributed hybrid flow shop scheduling with machine breakdown, *Computers & Industrial Engineering*, vol. 197, p. 110624, 2024.

[31] Q.-Y. Li, Q.-K. Pan, L. Wang, L. Gao, and W.-M. Li, Dynamic cascaded flow-shop scheduling using an evolutionary greedy algorithm, *IEEE Transactions on Evolutionary Computation*, 2025.

[32] X.-r. Tao, Q.-k. Pan, and L. Gao, An iterated greedy algorithm with reinforcement learning for distributed hybrid flowshop problems with job merging, *IEEE Transactions on Evolutionary Computation*, 2024.

[33] Y. Li, Q. Liu, C. Zhang, X. Li, and L. Gao, Graph-based dual-agent deep reinforcement learning for dynamic human-machine hybrid reconfiguration manufacturing scheduling, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 55, no. 11, pp. 8729–8741, 2025.

[34] Y. Liu, J. Fan, L. Zhao, W. Shen, and C. Zhang, Integration of deep reinforcement learning and multi-agent system for dynamic scheduling of re-entrant hybrid flow shop considering worker fatigue and skill levels, *Robotics and Computer-Integrated Manufacturing*, vol. 84, p. 102605, 2023.

[35] L. Luo, X. Yan, Q. Wu, and V. S. Sheng, Evolution strategies-guided deep reinforcement learning for dynamic hybrid flow-shop scheduling problem, *Tsinghua Science and Technology*, 2024.

[36] W. Xu, X. Luo, Y. Zhao, and Y. Zhang, Graph neural architecture for dynamic hybrid flowshop: Addressing stochastic events and uncertain processing sequences, *Neurocomputing*, p. 130636, 2025.

[37] L. Shi, G. Guo, and X. Song, Multi-agent based dynamic scheduling optimisation of the sustainable hybrid flow shop in a ubiquitous environment, *International Journal of Production Research*, vol. 59, no. 2, pp. 576–597, 2021.

[38] J. Duan, F. Liu, Q. Zhang, J. Qin, and Y. Zhou, Genetic programming hyper-heuristic-based solution for dynamic energy-efficient scheduling of hybrid flow shop scheduling with machine breakdowns and random job arrivals, *Expert Systems with Applications*, vol. 254, p. 124375, 2024.

[39] L. Luo and X. Yan, Selection hyperheuristic with knowledge-based q-learning for dynamic distributed hybrid flow shop scheduling problem considering operation inspection, *Swarm and Evolutionary Computation*, vol. 95, p. 101936, 2025.

[40] B. Li, K. Mellou, B. Zhang, J. Pathuri, and I. Menache, Large language models for supply chain optimization, *CoRR*, 2023.

[41] S. Guo, N. Yin, J. Kwok, and Q. Yao, Nested-refinement metamorphosis: Reflective evolution for efficient optimization of networking problems, in *Findings of the Association for Computational Linguistics: ACL 2025*, 2025, pp. 17 398–17 429.

[42] R. Zhong, A. G. Hussien, J. Yu, and M. Munetomo, Llmoa: A novel large language model assisted hyper-heuristic optimization algorithm, *Advanced Engineering Informatics*, vol. 64, p. 103042, 2025.

[43] A. Novikov, N. Vũ, M. Eisenberger, E. Dupont, P.-S. Huang, A. Z. Wagner, S. Shirobokov, B. Kozlovskii, F. J. Ruiz, A. Mehrabian *et al.*, Alphaevolve: A coding agent for scientific and algorithmic discovery, *arXiv preprint arXiv:2506.13131*, 2025.

[44] D. Forniés-Tabuenca, A. Uribe, U. Otamendi, A. Artetxe, J. C. Rivera, and O. L. de Lacalle, Remoh: A reflective evolution of multi-objective heuristics approach via large language models, *arXiv:2506.07759*, 2025.

[45] Z. Zhao, D. Tang, C. Liu, L. Wang, Z. Zhang, H. Zhu, K. Chen, Q. Nie, and Y. Ji, A large language model-based multi-agent manufacturing system for intelligent shopfloors, *Advanced Engineering Informatics*, vol. 69, p. 103888, 2026.

[46] R. Liao, J. Qiu, X. Chen, and X. Li, Llm4eo: Large language model for evolutionary optimization in flexible job shop scheduling, *arXiv preprint arXiv:2511.16485*, 2025.

[47] F. Zhao, J. Zhou, L. Wang, and H. Sang, A hierarchical optimization algorithm with dual-cache synced tuning mechanism for distributed flexible job shop scheduling problem, *IEEE Transactions on Cybernetics*, 2025.

[48] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii, *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[49] C. Li, Y. Han, B. Zhang, Y. Wang, J. Li, and K. Gao, A novel multi-objective hybrid evolutionary algorithm based on variable weight strategy for distributed hybrid flowshop scheduling with batch processing machines and variable sublots, *Applied Soft Computing*, vol. 170, p. 112650, 2025.

[50] J. Cohoon, S. Hegde, W. Martin, and D. Richards, Punctuated equilibria: a parallel genetic algorithm, in *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, 1987, pp. 148–154.

[51] M. Ruciński, D. Izzo, and F. Biscani, On the impact of the migration topology on the island model, *Parallel computing*, vol. 36, no. 10-11, pp. 555–571, 2010.

[52] E. Cantú-Paz, Migration policies, selection pressure, and parallel evolutionary algorithms, *Journal of Heuristics*, vol. 7, no. 4, pp. 311–334, 2001.

[53] E. Alba and M. Tomassini, Parallelism and evolutionary algorithms, *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, pp. 443–462, 2002.

[54] B. Zhang, Q.-k. Pan, L. Gao, X.-l. Zhang, and K.-k. Peng, A multi-objective migrating birds optimization algorithm for the hybrid flowshop rescheduling problem. *Soft Computing*, vol. 23, no. 17, 2019.

[55] N. Zhu, F. Zhao, Y. Yu, and L. Wang, A cooperative learning-aware dynamic hierarchical hyper-heuristic for distributed heterogeneous mixed no-wait flow-shop scheduling, *Swarm and Evolutionary Computation*, vol. 90, p. 101668, 2024.

[56] H. Qin, Y. Han, Q. Chen, L. Wang, Y. Wang, J. Li, and Y. Liu, Energy-efficient iterative greedy algorithm for the distributed hybrid flow shop scheduling with blocking constraints, *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 7, no. 5, pp. 1442–1457, 2023.

[57] Q.-K. Pan, P. N. Suganthan, J. J. Liang, and M. F. Tasgetiren, A local-best harmony search algorithm with dynamic sub-harmony memories for lot-streaming flow shop scheduling problem, *Expert Systems with Applications*, vol. 38, no. 4, pp. 3252–3259, 2011.

[58] C. Wang, Y. Yao, W. Li, and X. Li, A multi-population co-evolutionary algorithm for solving energy-efficient hybrid flow shop scheduling problem, *Expert Systems with Applications*, vol. 297, p. 129536, 2026.

[59] L. Nie, L. Gao, P. Li, and X. Li, A gep-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates, *Journal of Intelligent Manufacturing*, vol. 24, no. 4, pp. 763–774, 2013.

[60] F. Zhang, Y. Mei, and M. Zhang, Genetic programming with multi-tree representation for dynamic flexible job shop scheduling, in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2018, pp. 472–484.

Author biography

Yuning Lei is an undergraduate student of Grade 2023 in the School of Naval Architecture and Ocean Engineering, Huazhong University of Science and Technology (HUST), China. His research interests include shop floor scheduling and large language models.

Jin Huang received the M.S. degree in naval architecture and ocean engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2023, where he is currently pursuing the Ph.D. degree in mechanical engineering with the Huazhong University of Science and Technology. His current research interests include operations research, reinforcement learning, and large language models.

Xinyu Li received the Ph.D. degree in industrial engineering from the Huazhong University of Science and Technology (HUST), China, 2009. He is currently a Professor with the Department of Industrial Manufacturing Systems Engineering, State Key Laboratory of Intelligent Manufacturing Equipment Technology, School of Mechanical Science Engineering, HUST. He has published more than 100 refereed articles. His research interests include intelligent algorithm, big data, and machine learning.

Qihao Liu received the Ph.D. degree in industrial engineering from Huazhong University of Science and Technology (HUST), China, in 2022. He was a Post-Doctoral Research Associate of Industrial Engineering with the School of Mechanical Science and Engineering, HUST, from 2022 to 2024. Since January 2025, he has been a Lecturer in Industrial Engineering at HUST. His research interests include intelligent process planning and shop scheduling.

Liang Gao received the B.Sc. degree in mechatronic engineering from Xidian University, Xi'an, China, in 1996, and the Ph.D. degree in mechatronic engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2002. He is currently a Professor with the Department of Industrial Manufacturing System Engineering, State Key Laboratory of Intelligent Manufacturing Equipment Technology, School of Mechanical Science Engineering, HUST. He has published more than 400 refereed articles. His research interests include operations research and optimization, big data, and machine learning. He also serves as the Co-Editor-in-Chief for IET Collaborative Intelligent Manufacturing and an Associate Editor for Swarm and Evolutionary Computation and the Journal of Industrial and Production Engineering.