

Closed-Loop LLM-Based Hyperparameter Control for Evolutionary Algorithm in Hybrid Flow Shop Scheduling of Electrolytic Aluminum Production

Fuqing Zhao^(✉), Cheng Zhao, Tianpeng Xu, Ningning Zhu, and Houbin Song

Abstract The hybrid flow shop scheduling problem (HFSP) arises in electrolytic aluminum production, where multi-stage operations with parallel resources and strict molten-metal time windows require timely task completion to reduce thermal losses and prevent downstream congestion. This paper studies the HFSP with total tardiness (TTD) minimization, presents a mixed-integer linear programming (MILP) model, and develops a Large Language Models-assisted evolutionary algorithm, termed LLM-AEA. LLM-AEA combines a permutation-based evolutionary algorithm with triggered local search and a Large Language Models (LLMs)-driven hyperparameter controller. A multidimensional representation of the population state is created using fitness statistics, diversity indicators, elite ratios, and positional variance, which characterizes exploration-exploitation dynamics. Based on multi-round feedback, the LLMs updates crossover and mutation rates and allocates the local-search budget within trust regions using smoothed adjustments to stabilize parameter changes. When stagnation is detected, a triggered local search leveraging structural and tardiness-oriented neighborhoods intensifies the search, while random immigrants promote diversity recovery. Experiments across multiple instance scales show improvements over comparison algorithms, and results indicate robust performance across different choices of the underlying LLMs.

Keywords Hybrid flow shop scheduling, Large language models, Evolutionary algorithm, Electrolytic aluminum production

1 Introduction

Electrolytic aluminum production based on the Hall-Héroult process constitutes a continuous, rhythm-sensitive molten-

metal flow system. In a potline, alumina together with electrolyte additives (e.g., cryolite and fluorides) is prepared and metered into electrolytic cells equipped with consumable carbon anodes, where alumina is reduced in a high-temperature molten bath to generate liquid aluminum. Molten metal is tapped from cells at prescribed intervals to sustain bath chemistry, cell stability, and metal balance [1]. The tapped metal is collected in ladles and transferred by overhead handling systems to downstream refining or casting units. Across these linked operations, completion timing is constrained by tight thermal windows: prolonged waiting increases heat loss and raises the likelihood of congestion and mismatch on shared handling and processing resources. Such characteristics make timeliness a central planning concern for stable and efficient operation.

The planning-level scheduling setting associated with these coordinated operations is captured through a hybrid flow shop abstraction, focusing on assignment and sequencing decisions across multiple successive stages with parallel machines. In this abstraction, jobs represent molten-aluminum batches and stages correspond to major processing steps and downstream units along the molten-metal flow. Coordination requirements are expressed as job due-date targets that reflect expected completion times aligned with the plant operating rhythm, and deviations from these targets are quantified via total tardiness (TTD) [2]. Stage processing times are specified as effective planning-cycle times consistent with standard operating procedures, enabling a tractable representation of time-sensitive coordination while preserving the essential objective of punctual completion. Fig. 1 outlines the main process stages and the motivating molten-metal flow.

The hybrid flow shop scheduling problem (HFSP) describes a multi-stage production system arranged in series, where at least one stage comprises multiple parallel machines and each job visits all stages following an identical technological route. At each stage, a job is processed non-preemptively on exactly one machine, subject to machine-capacity and technological precedence constraints, and processing times may vary across jobs, stages, and machines [3]. Under the considered setting, each job is associated with a due date.

• Fuqing Zhao, Cheng Zhao, Tianpeng Xu, and Ningning Zhu are affiliated with the School of Computer Science and Artificial Intelligence, Lanzhou University of Technology, Lanzhou 730050, China (e-mail: zhaofq@lut.edu.cn; zczc1528@163.com; xutp@lut.edu.cn; 307516638@qq.com).

• Houbin Song is affiliated with the School of Automation and Electrical Engineering, Lanzhou University of Technology, Lanzhou 730050, China (e-mail: songhb@lut.edu.cn).

Manuscript received: 2026-03-05; revised: 2026-05-13; accepted: 2026-06-15

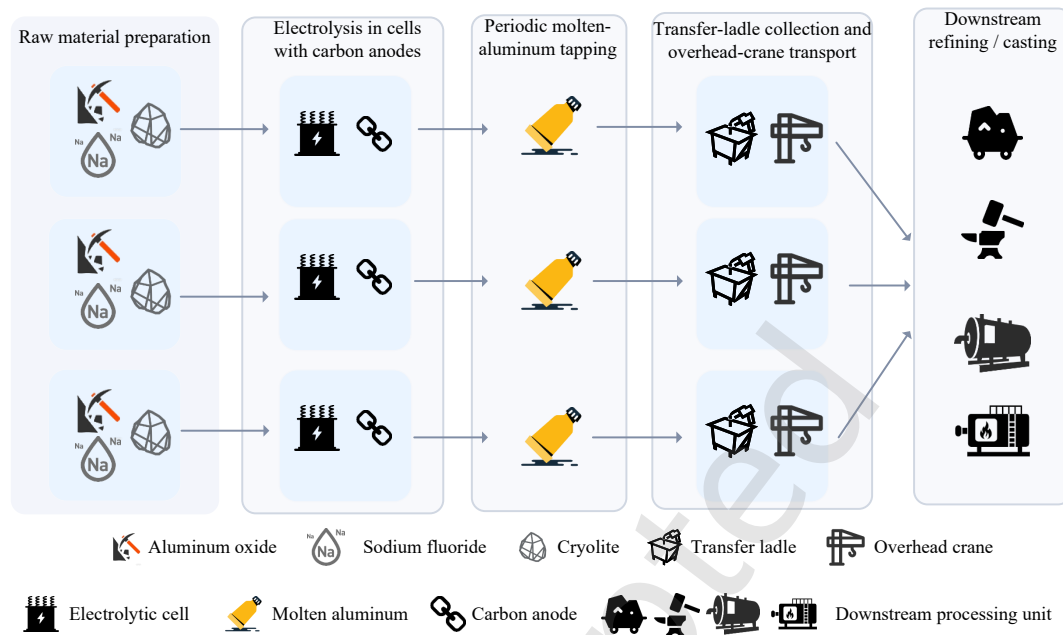


Fig. 1 Schematic of the electrolytic aluminum production process and molten-metal handling flow considered in this study

The objective minimizes TTD, which directly penalizes late completion relative to coordination targets and serves as a time-oriented proxy for operational loss and downstream mismatch risk.

Several practical characteristics of electrolytic aluminum production make this scheduling problem different from a canonical HFSP. The molten-aluminum batches follow a fixed technological route, but each stage contains multiple unrelated parallel machines, so the processing sequence and machine assignment are jointly determined rather than separable. A sequencing change at an upstream stage may alter the waiting time of a batch, the workload distribution among parallel machines, and the earliest feasible starting times at subsequent stages. Such delay propagation is particularly critical in molten-metal handling, where completion delays may disturb the expected production rhythm and increase the risk of heat loss, resource congestion, and mismatch with downstream refining or casting units. Moreover, the TTD objective does not simply measure the length of a schedule; it penalizes only the late completion of batches relative to their due-date targets. Therefore, effective improvement moves must identify operations that contribute to tardiness and adjust their positions without causing additional downstream delays. These features create a highly coupled and time-sensitive scheduling environment, in which feasible schedules are easy to construct but high-quality schedules that maintain punctual molten-metal flow are difficult to obtain.

From a computational perspective, the HFSP generalizes the classical flow shop and remains NP-hard [4] even under simplified settings. As the number of jobs, stages, and machines increases, the combinatorial search space grows rapidly, making exact methods computationally prohibitive. In industrial environments, this challenge is further compounded by heterogeneous equipment and tight operational coordination. Consequently, heuristic and metaheuristic methods—including genetic algorithms (GA), artificial bee colony (ABC) algorithms, and iterated greedy schemes—are widely adopted to solve practical HFSP variants.

Large language models (LLMs) have recently been explored in evolutionary computation and scheduling optimization as optimizer components, hyper-heuristic modules, surrogate evaluators, and feedback-driven coordinators [5–8]. These studies suggest that LLMs can be useful when optimization decisions involve heterogeneous feedback, qualitative search-state interpretation, and repeated adaptation without task-specific model training. In the present study, the LLMs is not introduced to directly generate schedules, replace the objective evaluation, or design new search operators. Instead, it is used as a training-free online hyperparameter controller. This choice is motivated by the search characteristics of the considered HFSP: the population may enter different regimes, such as premature convergence, dispersed but slow progress, or balanced exploration and exploitation, and these regimes cannot be sufficiently described by a single fitness value.

The LLMs can jointly interpret fitness statistics, stagnation information, permutation-level diversity, elite concentration, positional variance, and the current hyperparameter values, and then provide candidate adjustments for the crossover rate, mutation rate, and local-search budget. Compared with fixed rules or offline-trained controllers, this mechanism avoids task-specific training and allows the same control protocol to be applied across different instance scales and base LLMs. To ensure reliability, the raw LLMs suggestions are not used directly but are constrained by trust regions, smoothed updates, and global bounds.

A large language model-assisted evolutionary framework, termed LLM-AEA, is introduced for the HFSP with TTD minimization motivated by electrolytic aluminum production scheduling. Candidate schedules are encoded as job permutations and evaluated through a stage-by-stage decoding procedure. Global exploration is carried out by a permutation-based evolutionary algorithm with order crossover and swap mutation. To enhance intensification, a stagnation-triggered local-search module is embedded, combining structural neighborhoods with tardiness-oriented moves, while random immigrants promote diversity recovery. A closed-loop LLMs controller adaptively adjusts crossover probability, mutation probability, and local-search budget using a multidimensional population-state representation that integrates fitness statistics, diversity indicators, elite ratios, and positional variance. Trust-region restrictions and smoothed updates stabilize parameter changes and improve robustness across different instance scales and search phases.

The main contributions of this paper are summarized as follows:

- (1) **A training-free, closed-loop LLMs controller for online hyperparameter adjustment:** A model-agnostic interaction protocol is designed in which an LLMs periodically receives compact population summaries and returns updated crossover probability, mutation probability, and local-search budget. The suggestions are stabilized through bounded trust regions and smoothing to ensure interpretable and robust online control without offline training.
- (2) **A permutation-aware population state representation for search-regime diagnosis:** A multidimensional state vector is constructed by combining fitness distribution statistics with complementary structural signals, including distinct-permutation diversity, elite concentration, and normalized positional variance. This representation supports the discrimination of converged/stagnant, dispersed/slow-progress, and balanced search regimes, enabling phase-dependent parameter adaptation.
- (3) **An integrated stagnation-handling mechanism with guided intensification and diversity recovery:** A triggered local-search module is embedded to intensify the search by mixing structural and tardiness-oriented neighborhoods, and it is coupled with controlled random-immigrant injection for diversity restoration.

The remainder of this paper is organized as follows: Section 2 reviews related literature. Section 3 presents the HFSP formulation, including notation and a mixed-integer linear programming (MILP) model. Section 4 details the proposed LLM-AEA framework. Section 5 reports and discusses the experimental results. Section 6 presents further analyses of the experimental results. Finally, Section 7 concludes the paper and outlines directions for future research.

2 Related Work

Liu [9] develops an end-to-end deep reinforcement learning framework based on proximal policy optimization and graph attention networks to solve dynamic HFSP with job insertions, minimizing total weighted tardiness and enabling scalable real-time production decisions. Ren [10] tackles a dynamic reentrant HFSP with multi-stage batch processing machines by designing a hierarchical strategy-optimized double deep Q-network (DDQN) that jointly learns batching and scheduling policies and outperforms heuristic rules and existing Deep Reinforcement Learning (DRL) methods. Extending DRL to more complex dynamic environments, Liu [11] proposes a cooperative multi-agent framework based on Proximal Policy Optimization (PPO) for dynamic HFSP with unrelated parallel machines and preventive maintenance, simultaneously optimizing makespan, processing cost, and maintenance cost. Beyond single-factory settings, Luo [12] introduces an EDA/PPO hybrid optimization algorithm for stochastic distributed HFSP (DHFSP) with processing time perturbations, aiming to minimize makespan, while Di [13] develops an evolution-strategy-trained multi-graph neural network method for real-time DHFSP with new job arrivals and machine breakdowns, targeting both makespan and energy consumption. Focusing on rescheduling under disruptions, Zuo [14] addresses the DHFSP with machine breakdowns through a rescheduling method and a two-stage learning scatter search algorithm enhanced by reinforcement learning and hybrid search to improve scheduling performance.

For HFSP with sophisticated constraints, metaheuristic and hyper-heuristic methods have also been extensively explored. Wang [15] studies a constrained HFSP where only some jobs share a common deadline and proposes a variable-

representation discrete ABC algorithm with tailored heuristics that efficiently produce high-quality solutions. From a hyper-heuristic perspective, Chen [16] proposes a Q-learning-based multiobjective framework for HFSP with lot streaming, aiming to simultaneously minimize makespan, energy consumption, and tardiness. Considering fuzziness and batch processing, Wang [17] tackles an energy-efficient fuzzy HFSP with batch processing machines by proposing a multiobjective dynamic ABC algorithm that jointly minimizes fuzzy makespan and energy consumption. In distributed environments with additional deterioration and blocking constraints, Liu [18] investigates a DHFSP with deterioration effects and proposes an evolutionary multitasking memetic algorithm that uses transfer learning and local intensification to solve it efficiently, while Liu [19] develops a three-phase multi-strategy iterated greedy algorithm for DHFSP with blocking constraints to significantly reduce makespan.

Mathematical programming models have recently been used not only as exact solvers for small-scale HFSP variants, but also as formal benchmarks for verifying heuristic and metaheuristic algorithms. Huang [20] formulate a two-stage stochastic programming MILP model for HFSP under multiple uncertain factors and develop an H-PDDE algorithm, showing that the stochastic formulation better captures uncertain production environments. Alexander [21] present two MILP formulations for a multi-stage hybrid assembly flow shop with release times and preventive maintenance, and validate the formulations through exact solution experiments. For distributed and assembly-oriented settings, Yu [22] investigate a distributed assembly hybrid flow shop with dual-resource constraints and TTD minimization, where CPLEX is used to validate the MILP model on small-sized instances before a knowledge-based iterated greedy algorithm is developed for larger scales. Yu [23] further consider an energy-efficient distributed assembly hybrid flow shop with worker resources, jointly minimizing TTD and total energy consumption, and use CPLEX to verify the MILP model on small-sized instances. From the energy-management perspective, Taguement [24] develop a time-indexed 0-1 ILP model for a multi-line HFSP under time-of-use electricity pricing, while Shao [25] design an MILP model for a distributed heterogeneous HFSP with multilevelled mixed-model assembly and combine it with a learning competitive swarm optimizer.

Energy efficiency and distributed production considerations form another important research stream. Lu [26] introduces an energy-aware DHFSP model and a hybrid multiobjective iterated greedy algorithm to jointly minimize makespan and total energy consumption, whereas Liu [27] formulates

an energy-efficient DHFSP model for forging production and proposes an elite cooperative non-dominated sorting genetic algorithm II to minimize makespan and energy consumption. Earlier work in energy-aware DHFSP includes the bi-population cooperative memetic algorithm of Wang [28] and the cooperative memetic and improved multiobjective whale optimization algorithms of Wang [29], which integrate collaborative initialization, bi-population cooperation, learning-based agents, and local intensification while also considering device dynamic reconfiguration processes and adjustable processing modes, thereby highlighting the central role of energy and reconfiguration in such systems. Energy-efficient HFSP variants have also been addressed by Qin [30], who develop an improved iterated greedy algorithm tailored for energy-efficient blocking HFSP, and by Shao [31], who propose a multiobjective evolutionary algorithm based on multiple-neighborhood local search to simultaneously minimize makespan, total weighted earliness and tardiness, and total workload. Uncertainty is further incorporated by Cai [32], who tackle energy-efficient DHFSP with fuzzy processing times using a cooperated shuffled frog-leaping algorithm that jointly optimizes fuzzy makespan, energy consumption, and agreement indices.

Recent LLMs-assisted optimisation studies have assigned the LLMs different functional roles. Some methods use LLMs as high-level hyper-heuristic components to construct or select search-operator sequences [6], while others employ them as surrogate evaluators [7], or multi-agent coordinators that automate broader optimisation workflows [8]. In scheduling, the recently reported LLMs-assisted automatic memetic framework further uses LLMs guidance to design local-search operators [5]. These studies demonstrate the versatility of LLMs in optimisation, but their primary focus is typically on operator generation, solution evaluation, or workflow automation. By contrast, the present study considers a narrower but more explicit role for the LLMs, namely, online hyperparameter control within a fixed permutation-based evolutionary framework. The LLMs updates crossover, mutation, and local-search parameters according to a structured population-state representation.

The above studies indicate that HFSP-related research has been developed through mathematical programming, classical metaheuristics, problem-specific search frameworks, learning-based methods, and recent LLMs-assisted optimization. Mathematical programming models provide rigorous formulations and useful references for small-scale validation, but the simultaneous consideration of machine assignment, job sequencing, stage precedence, and tardiness variables

makes their direct use difficult for larger HFSP instances. Classical metaheuristics such as GA, ABC, PSO, and DE provide flexible search frameworks, whereas their effectiveness in permutation-based HFSPs depends heavily on parameter settings and on whether the adopted operators can reflect problem-related timing information, especially when unrelated parallel machines make assignment and sequencing decisions interact with each other. Learning-based and hyperheuristic methods introduce adaptive decisions into the search process, yet they commonly rely on predefined state-action spaces, task-specific training, or manually prepared operator pools. Recent LLMs-assisted optimization methods further reduce part of the manual design burden by using LLMs for operator construction, surrogate evaluation, or workflow-level coordination; however, for HFSP with unrelated parallel machines, multi-stage delay propagation, and TTD-oriented completion requirements, LLMs-based online hyperparameter control within a fixed evolutionary search framework remains insufficiently explored.

2.1 Research Gap

Based on the above literature review and the characteristics of the considered problem, the existing research gaps can be summarized as follows:

- (1) **Limited and costly online parameter control:** Most state-of-the-art metaheuristics for HFSP still depend on manually tuned operator probabilities and local-search effort. Learning-based controllers can adapt online, but typically require task-specific training and carefully engineered state inputs, which reduces portability when instance scales and operating conditions change.
- (2) **Under-utilization of population-level structural signals:** Many adaptive schemes monitor only scalar fitness trends or simple diversity measures, and seldom expose richer, permutation-structure-aware population information that directly reflects exploration-exploitation regimes in permutation search.
- (3) **Weak coupling between stagnation diagnosis and intensification design:** Local search is often triggered by ad hoc rules with fixed budgets and neighborhoods. Such triggers do not leverage global population feedback to decide when to intensify and how strongly, and they are rarely integrated with explicit diversity recovery mechanisms.

3 Hybrid Flow Shop Scheduling Problem

3.1 Problem Description

This study considers a HFSP with TTD minimization motivated by electrolytic aluminum production. In the scheduling abstraction, each job represents a molten-aluminum batch, and the production system consists of m stages arranged in series. All jobs follow the same processing route through these stages. At stage k ($k = 1, \dots, m$), there are l_k parallel machines. The machines are unrelated, meaning that the processing time of job j at stage k depends on the selected machine i and is denoted by $p_{j,k,i}$.

The scheduling task is to determine, for each job at each stage, both the machine assignment and the processing sequence on the selected machine. Each job requires exactly one non-preemptive operation at every stage and must be processed on exactly one machine at that stage. All jobs are available at time zero, and all machines are assumed to be continuously available during the planning horizon. A job can be processed by at most one machine at any time, and each machine can handle at most one job at any time. In addition, a job may start processing at stage $k + 1$ only after completing its operation at stage k . Transportation times between consecutive stages are neglected.

Each job j is associated with a due date d_j . Let C_j denote the completion time of job j at the final stage. The tardiness of job j is then defined as

$$T_j = \max\{0, C_j - d_j\}.$$

The objective of the problem is to minimize the total tardiness of all jobs, thereby improving completion timeliness and reducing the risk of downstream mismatch and molten-metal heat loss. The notation and decision variables used in the MILP model are summarized in Table 1.

3.2 MILP Model

The HFSP with TTD minimization considered in this study is modeled using a disjunctive MILP structure. The model describes machine assignment, job sequencing, stage precedence, and machine-capacity restrictions through assignment variables, pairwise sequencing variables, and Big-M non-overlap constraints [33, 34]. Compared with a general HFSP model, the considered model incorporates machine-dependent processing times and tardiness variables associated with job due dates, thereby characterizing the timeliness requirements of electrolytic-aluminum production scheduling.

$$TTD = \sum_{j=1}^n T_j \quad (1)$$

Table 1 Notation and decision variables

Notation	Description
j	Index of a job, $j = 1, \dots, n$
j'	Index of another job, $j' = 1, \dots, n, j' \neq j$
k	Index of a stage, $k = 1, \dots, m$
i	Index of a parallel machine at stage k , $i = 1, \dots, l_k$
J_j	Job j
n	Total number of jobs
m	Total number of stages
l_k	Number of parallel machines at stage k
$p_{j,k,i}$	Processing time of job j on machine i at stage k
d_j	Due date of job j
U	A sufficiently large positive constant
Variable	Description
$S_{j,k}$	Starting time of job j at stage k
$y_{k,j,i}$	1 if job j is processed on machine i at stage k , and 0 otherwise
$z_{k,j,j'}$	1 if job j is processed before job j' at stage k , and 0 otherwise
C_j	Completion time of job j at the last stage
T_j	Tardiness of job j
TTD	Total tardiness of all jobs

$$\min TTD \quad (2)$$

s.t.

$$C_j = S_{j,m} + \sum_{i=1}^{l_m} p_{j,m,i} y_{m,j,i}, \forall j \quad (3)$$

$$T_j \geq C_j - d_j, \forall j \quad (4)$$

$$S_{j,1} \geq 0, \forall j \quad (5)$$

$$S_{j,k+1} \geq S_{j,k} + \sum_{i=1}^{l_k} p_{j,k,i} y_{k,j,i}, \forall j, k = 1, \dots, m-1 \quad (6)$$

$$\sum_{i=1}^{l_k} y_{k,j,i} = 1, \forall j, k \quad (7)$$

$$z_{k,j,j'} + z_{k,j',j} \leq 1, \forall k, j \neq j' \quad (8)$$

$$z_{k,j,j'} + z_{k,j',j} \geq y_{k,j,i} + y_{k,j',i} - 1, \forall k, i, j > j' \quad (9)$$

$$S_{j',k} - S_{j,k} - p_{j,k,i} + U(3 - y_{k,j,i} - y_{k,j',i} - z_{k,j,j'}) \geq 0, \quad \forall k, i, j \neq j' \quad (10)$$

$$y_{k,j,i} \in \{0, 1\}, \forall k, j, i \quad (11)$$

$$z_{k,j,j'} \in \{0, 1\}, \forall k, j, j' \quad (12)$$

$$S_{j,k} \geq 0, \forall j, k \quad (13)$$

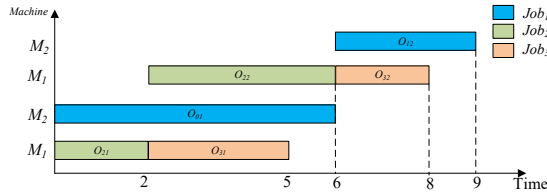
$$C_j \geq 0, T_j \geq 0, \forall j \quad (14)$$

The objective function in (2) minimizes the total tardiness of all jobs. Constraint (3) defines the completion time of job j at the last stage as the finishing time of its operation at stage m . Constraint (4) specifies the tardiness as the positive deviation of the completion time from the due date. Constraint (5) ensures that the starting time of each job at the first stage is not earlier than zero, while Constraint (6) enforces that an operation of a job can start only after the completion of its operation at the previous stage. Constraint (7) guarantees that each job is processed on exactly one machine at each stage. Constraints (8) and (9) jointly define a unique processing order for any pair of jobs that are assigned to the same machine. Constraint (10) prevents time overlaps between jobs processed on the same machine according to the defined order. Constraint (13) enforces the nonnegativity of all starting times. Constraints (11) and (12) specify the binary nature of the assignment and sequencing decision variables, respectively, and Constraint (14) imposes the nonnegativity of the completion times and tardiness of all jobs.

To validate the correctness of the proposed MILP model, a small-scale instance with three jobs and two stages was constructed and solved by CPLEX. The detailed data of this instance are reported in Table 2. Fig. 2 illustrates the Gantt chart of the solution obtained by CPLEX for the validation instance. The schedule shows that Job 3 and Job 1 incur tardiness values of 2 and 1, respectively, and therefore the total tardiness is 3.

Table 2 Small-scale instance used for MILP model validation

	Stage 1		Stage 2		d_j
	M1	M2	M1	M2	
Job1	4	6	5	3	8
Job2	2	5	4	6	7
Job3	3	4	2	5	6

**Fig. 2** A solution obtained by CPLEX

4 A Hyperparameter-Adaptive Evolutionary Algorithm Framework

4.1 The proposed framework

The proposed framework integrates a permutation-based evolutionary algorithm, a triggered local search, and an LLMs-driven hyperparameter controller for the HFSP with TTD minimization. The overall workflow is summarized in the flowchart shown in Fig. 3. Starting from a randomly generated population, each individual is decoded into a feasible schedule and evaluated. An iterative cycle then performs termination checking, elitist updating, and stagnation detection; during stagnated phases, local search refines the neighborhood around the current best solution, and random immigrants are injected to restore population diversity. This diversity recovery mechanism alleviates the structural similarity that may arise during permutation-based evolutionary search. Once the hyperparameter tuning condition is met, population statistics and structural features are computed and provided to the LLMs. Based on the resulting multidimensional search state, the LLMs adaptively update crossover, mutation, and local search parameters. Guided by these parameters, tournament selection, order crossover, and swap mutation generate a new population, after which evaluation, control, and improvement steps repeat until the maximum generation index is reached and the best schedule is obtained.

4.2 Encoding and decoding

In the proposed framework, each chromosome adopts a permutation-based representation for the HFSP. A chromosome is a one-dimensional sequence of job indices $(\pi_1, \pi_2, \dots, \pi_J)$, where each job appears exactly once, and the relative order of jobs in the sequence defines a

global dispatching priority shared by all stages. For illustration, Fig. 4 depicts a chromosome of length eight, $\pi = (J_3, J_7, J_1, J_5, J_2, J_8, J_4, J_6)$, whose permutation governs the order in which these eight jobs enter every production stage.

Decoding transforms a chromosome such as π into a feasible schedule by simulating job processing stage by stage on the parallel machines of the HFSP. For a three-stage configuration with two machines per stage, jobs are scanned in the order of π at each stage; for every job, the decoder computes for each machine the earliest feasible starting time as the maximum of the job completion time at the previous stage and the machine ready time, and assigns the job to the machine that yields the smallest completion time. This procedure produces the machine-level Gantt chart in Fig. 4, where Stage 1 uses machines M_1 and M_2 , Stage 2 uses M_3 and M_4 , and Stage 3 uses M_5 and M_6 , with colored bars indicating the processing intervals of $J_3, J_7, J_1, J_5, J_2, J_8, J_4$, and J_6 on each machine.

4.3 Offspring production strategy

The algorithm employs an order crossover (OX) operator tailored for permutation-based chromosomes. Two parents are selected, and a contiguous block of jobs from the first parent is copied directly into the offspring as a preserved subsequence. The remaining positions in the offspring are then filled according to the relative order of the jobs in the second parent, skipping all jobs already contained in the preserved block, so the offspring inherits both a building block from the first parent and the global ordering information from the second parent.

As illustrated in Fig. 5, the first parent chromosome is $(1, 2, 3, 4, 5, 6, 7, 8)$ and the second parent chromosome is $(3, 7, 5, 8, 2, 1, 4, 6)$. In this example, the subsequence $(3, 4, 5, 6)$ in the first parent is selected as the preserved block. During order crossover, this block is copied directly into the offspring at the same positions. The remaining subsequence $\{7, 8, 2, 1\}$ is then considered in the order in which they appear in the second parent and sequentially placed into the empty positions of the offspring. This procedure yields the offspring shown in Fig. 5 and illustrates that the crossover operator maintains the permutation constraint while combining structural information from both parents.

Mutation relies on a simple swap operator acting on permutation chromosomes. Two positions are chosen at random, and the jobs at these positions exchange their locations, which introduces a local perturbation without altering the job set. Fig. 6 presents an example in which the jobs with indices 3 and

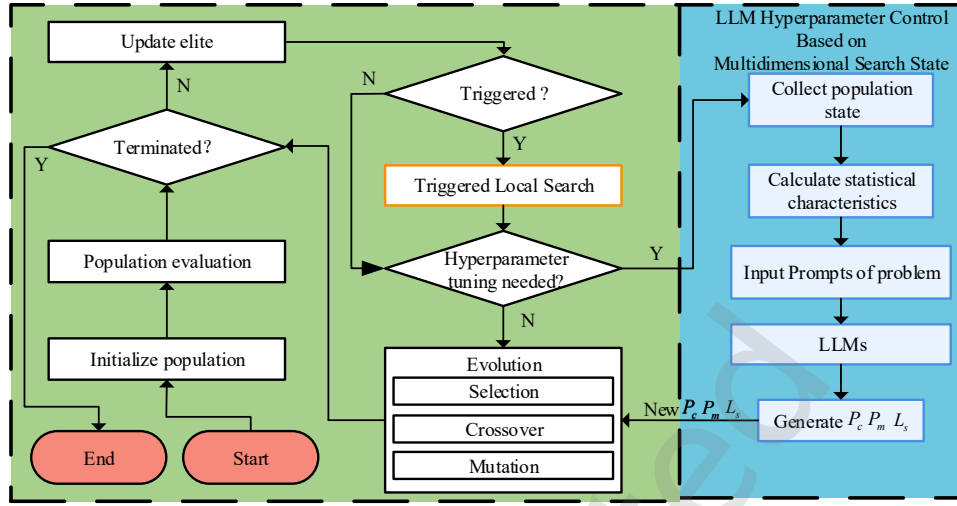


Fig. 3 Framework of LLM-AEA

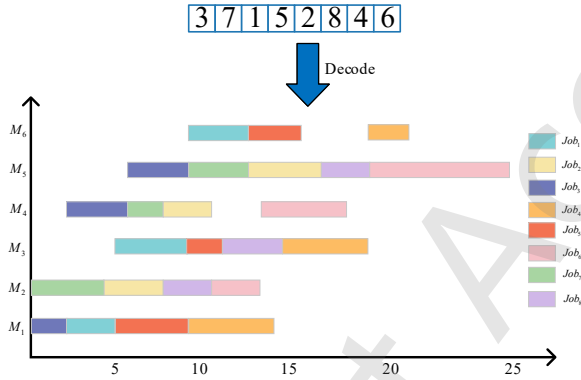


Fig. 4 Example of decoding

7 in the parent sequence (1, 2, 3, 4, 5, 6, 7, 8) are swapped, producing the mutated offspring (1, 2, 7, 4, 5, 6, 3, 8). This swap mutation enhances local exploration around promising permutations and complements the global recombination effect of the order crossover. Since the considered HFSP involves multiple unrelated parallel machines across successive stages, preserving beneficial partial job orders while maintaining sufficient structural perturbation helps balance coordination quality and exploration in the highly coupled permutation search space.

4.4 Triggered local search mechanism

Triggered local search acts as an intensification mechanism that is activated when the evolutionary search becomes stagnant. A plateau counter records the number of consecutive generations without improvement of the global best TTD;

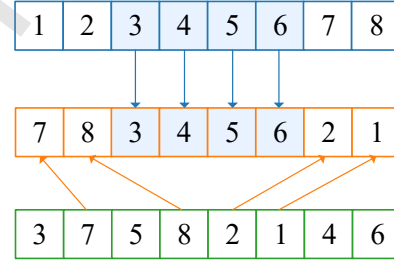


Fig. 5 Example of crossover

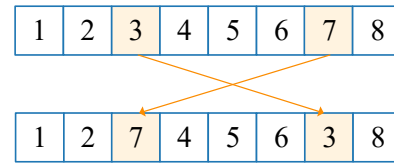


Fig. 6 Example of mutation

once this counter reaches a predefined patience threshold, Algorithm 1 is invoked. The current best individual serves as the initial solution of the local search, and at most $ls_{\max}$ iterations are performed. After the local search terminates, the improved solution π^{LS} replaces the worst individual in the population, and a fixed proportion of additional individuals is replaced by random immigrants, so that the next generation starts from a population that inherits both the refined best solution and renewed diversity.

In each iteration of Algorithm 1, a neighbor of the incumbent permutation is generated and accepted whenever it

yields a lower TTD. With probability 0.5, a purely structural move is applied: either **SWAPMOVE**, which exchanges two randomly chosen positions in the sequence, or **RELOCATEMOVE**, which removes a job from its current position and reinserts it at another position. With the remaining probability, problem-specific information is exploited. The tardiness of each job is computed, the set of tardy jobs is identified, and one tardy job is selected with probability proportional to its tardiness contribution. The selected job is removed from the current permutation to form a base sequence; a limited set of insertion positions is then sampled from the base sequence, and reinserting the job at these positions generates several candidate permutations. After evaluating the TTD of these candidates, the permutation with the smallest TTD is selected as the proposed neighbor. If no job is tardy, the procedure falls back to a simple swap move. By repeatedly combining these structural and tardiness-guided neighborhoods, the triggered local search refines the job sequence around high-quality regions while preserving the feasibility of the HFSP schedule. These tardiness-guided modifications prioritize jobs with large tardiness values and reduce the propagation of downstream completion delays caused by unfavorable job positions in the permutation.

4.5 Multidimensional search state features

In each generation, the framework constructs a multidimensional search state vector that summarizes both fitness progression and structural properties of the population. This state description extends beyond traditional indicators such as the best or average fitness and provides a richer view of whether the search moves in a strongly converged regime, a highly dispersed regime, or a balanced regime. For the considered HFSP, populations with similar TTD values may still exhibit substantially different permutation structures and diversity distributions, especially under multi-stage delay propagation and unrelated parallel-machine interactions. The state vector later serves as input to the LLMs-based hyperparameter controller in the subsequent subsection, so the controller receives quantitative information about stagnation, diversity, and structural concentration of elite solutions rather than relying solely on scalar fitness values.

The first component of the state characterizes fitness statistics and stagnation. For a population P_g of size N in generation

Algorithm 1 Triggered local search mechanism

Input: Current best solution π^{best} with fitness f^{best} , plateau counter $plateau$, patience threshold $patience$, maximum iterations ls_{max} , population P

Output: Improved solution π^{LS} with fitness f^{LS} , updated population P

```

1  $\pi^{LS} \leftarrow \pi^{best}; f^{LS} \leftarrow f^{best}$ 
2 if  $plateau \geq patience$  then
3    $\pi \leftarrow \pi^{best}; f \leftarrow f^{best}$ 
4   for  $iter = 1$  to  $ls_{max}$  do
5     draw  $u \sim U(0, 1)$ 
6     if  $u < 0.5$  then
7        $v \sim U(0, 1)$ 
8       if  $v < 0.5$  then
9          $\pi' \leftarrow \text{SWAPMOVE}(\pi)$ 
10      else
11         $\pi' \leftarrow \text{RELOCATEMOVE}(\pi)$ 
12      end if
13    else
14      compute tardiness vector  $T$  for  $\pi$ 
15       $S \leftarrow \{j \mid T_j > 0\}$ 
16      if  $S = \emptyset$  then
17         $\pi' \leftarrow \text{SWAPMOVE}(\pi)$ 
18      else
19        select job  $k \in S$  with probability
20           $T_k / \sum_{j \in S} T_j$ 
21         $\pi_{base} \leftarrow$  remove  $k$  from  $\pi$ 
22        choose candidate positions  $\mathcal{L}$  in  $\pi_{base}$ 
23         $\pi' \leftarrow \pi; f' \leftarrow f$ 
24        for each  $pos \in \mathcal{L}$  do
25           $\pi^{cand} \leftarrow$  insert  $k$  at  $pos$  in  $\pi_{base}$ 
26           $f^{cand} \leftarrow \text{EVALUATE}(\pi^{cand})$ 
27          if  $f^{cand} < f'$  then
28             $\pi' \leftarrow \pi^{cand}; f' \leftarrow f^{cand}$ 
29          end if
30        end for
31      end if
32    end if
33    if  $f'$  is not yet computed then
34       $f' \leftarrow \text{EVALUATE}(\pi')$ 
35    end if
36    if  $f' < f$  then
37       $\pi \leftarrow \pi'; f \leftarrow f'$ 
38    end if
39  end for
40   $\pi^{LS} \leftarrow \pi; f^{LS} \leftarrow f$ 
41  replace worst individual in  $P$  by  $(\pi^{LS}, f^{LS})$ 
42  inject random immigrants into  $P$ 
43 end if
44 return  $(\pi^{LS}, f^{LS}, P)$ 

```

g , the best, average, and worst TTD values satisfy

$$\begin{aligned} f_g^{\text{best}} &= \min_{i \in P_g} f_i \\ f_g^{\text{avg}} &= \frac{1}{N} \sum_{i \in P_g} f_i \\ f_g^{\text{worst}} &= \max_{i \in P_g} f_i \end{aligned} \quad (15)$$

where f_i denotes the TTD of individual i . The plateau length records the number of consecutive generations without improvement of the global best objective value, and evolves according to

$$\text{plateau}_g = \begin{cases} 0, & f_g^{\text{best}} < f_{g-1}^{\text{best}}, \\ \text{plateau}_{g-1} + 1, & f_g^{\text{best}} \geq f_{g-1}^{\text{best}}, \end{cases} \quad (16)$$

A large value of plateau_g in Eq. (16) indicates that the search remains trapped in a region where the best fitness no longer decreases.

Population diversity is measured at the permutation level and reflects how many distinct sequences exist in the current population. Let U_g denote the set of distinct permutations in P_g , and N denote the population size. The diversity indicator for generation g is defined as

$$D_g = \frac{|U_g|}{N} \quad (17)$$

which lies in $[0, 1]$ by construction. Values of D_g in Eq. (17) close to one indicate that most individuals differ from each other and the search explores a wide region of the solution space, whereas values close to zero indicate that the population collapses around a few permutations.

The elite ratio describes how many individuals concentrate near the current best fitness level. For generation g , a dynamic elite threshold is defined as

$$f_{\text{thr},g} = f_g^{\text{best}} + 0.1(f_g^{\text{worst}} - f_g^{\text{best}}) \quad (18)$$

and the elite set contains all individuals with fitness not larger than this threshold,

$$E_g = \{i \in P_g \mid f_i \leq f_{\text{thr},g}\} \quad (19)$$

The elite ratio then reads

$$R_g^{\text{elite}} = \frac{|E_g|}{N} \quad (20)$$

A high value of R_g^{elite} in Eq. (20) indicates that a large fraction of the population lies in a narrow band close to the best solution and the search has already formed a dense basin of near-optimal permutations, while a low value indicates a more dispersed fitness with relatively few near-elite individuals.

Structural diversity is further quantified at the level of job positions by a positional variance index. For each job $j \in \{J_1, \dots, J_n\}$, denote by $\text{pos}_{g,j}^i$ the position of job j in individual i of generation g . The mean position and variance

of job j over the population satisfy

$$\mu_{g,j} = \frac{1}{N} \sum_{i=1}^N \text{pos}_{g,j}^i \quad (21)$$

$$\sigma_{g,j}^2 = \frac{1}{N} \sum_{i=1}^N (\text{pos}_{g,j}^i - \mu_{g,j})^2 \quad (22)$$

The average positional variance over all jobs is given by

$$\bar{\sigma}_g^2 = \frac{1}{n} \sum_{j=1}^n \sigma_{g,j}^2 \quad (23)$$

and an approximate upper bound for the variance of positions distributed over $\{1, \dots, n\}$ is

$$\sigma_{\text{max}}^2 = \frac{n^2 - 1}{12} \quad (24)$$

The normalized positional variance index, therefore, takes the form

$$V_g = \frac{\bar{\sigma}_g^2}{\sigma_{\text{max}}^2}, \quad V_g \in [0, 1] \quad (25)$$

Values of V_g in Eq. (25) near zero indicate that jobs occupy very similar positions across the population, which reflects strong structural convergence; values near one indicate that job positions differ substantially, which reflects high positional diversity.

All these quantities together form the multidimensional search state vector at generation g ,

$$\mathbf{s}_g = (N, f_g^{\text{best}}, f_g^{\text{avg}}, f_g^{\text{worst}}, \text{plateau}_g, D_g, R_g^{\text{elite}}, V_g) \quad (26)$$

The state vector in Eq. (26) integrates information about fitness distribution, stagnation history, permutation-level diversity, and job-level structural variance. In the subsequent subsection, $\mathbf{s}_g$ serves as the input descriptor of the current search regime for the LLMs-based hyperparameter controller, which uses these signals to determine whether the algorithm enters an intensification-oriented, exploration-oriented, or balanced adjustment mode.

4.6 LLMs-based hyperparameter control mechanism

The LLMs-based controller adapts three key hyperparameters of the LLM-AEA framework: the crossover rate, the mutation rate, and the maximum number of local search iterations. At the end of each generation g , once a burn-in phase has passed, the procedure in Algorithm 2 checks whether the generation index and the current plateau length satisfy predefined triggering conditions (Step 1). If the conditions are not satisfied, the parameters for the next generation remain unchanged. Otherwise, the controller takes as input the multidimensional state vector $\mathbf{s}_g$ defined in the previous subsection, together

with the current parameter triplet $(p_g^{\text{cross}}, p_g^{\text{mut}}, l_{s_{\max,g}})$, and queries the LLMs for a suggested update (Step 2). The LLMs receives the best, average, and worst TTD, the plateau length, the population size, the diversity indicator, the elite ratio, the normalized positional variance, and the current hyperparameters, and interprets this information in terms of search regimes such as strongly converged and stagnant, highly dispersed and slow-converging, or intermediate balanced states. Fig. 7 shows the prompt information used in this study.

Prompt Of Hyperparameter Control

Task: You are an automatic hyperparameter controller for a genetic algorithm that minimizes total tardiness in a hybrid flow shop scheduling problem. Your ONLY task is to slightly update three hyperparameters: P_c : controls exploitation via recombining good solutions; P_m : controls exploration via random changes; L_s : controls local search intensity

Input: $[Generation, TTD_{\text{best}}, TTD_{\text{avg}}, TTD_{\text{worst}}, Plateau, POP_{\text{size}}, Diversity, Elite_{\text{ratio}}, Positional_{\text{variance}}, Current_{\text{params}}]$

Output: $[P_c, P_m, L_s]$

Other information: If $(Diversity < 0.3 \text{ OR } Positional_{\text{variance}} < 0.3)$ and $Plateau \geq 10$ and $Elite_{\text{ratio}} \geq 0.5$: population is structurally converged around a narrow basin of good solutions but progress has stalled. => Needs more exploration.
 Else if $(Diversity > 0.7 \text{ or } positional_{\text{variance}} > 0.7)$ And $Plateau \geq 5$ and $Elite_{\text{ratio}} < 0.4$: population is highly diverse and progress is slow, with few near-optimal solutions. => Needs more exploitation.
 Otherwise: => Blanced mode: minor, stable adjustments only.

More exploration: slightly Increase P_m by about +0.02 to +0.05; keep P_c in a moderate-to-high range; you may slightly increase L_s , but keep it within a reasonable range.
 Needs more exploitation: keep P_c HIGH and you may increase it slightly; slightly decrease P_m by about -0.02 to -0.05; you may slightly increase L_s to intensify local search.
 Blanced : keep P_c and P_m close to their current values; with only very small changes

Fig. 7 Example of prompt

The raw suggestion of the LLMs is represented by the parameter triplet

$$(\tilde{p}_g^{\text{cross}}, \tilde{p}_g^{\text{mut}}, \tilde{l}_{s_{\max,g}}) \quad (27)$$

which follows qualitative rules: in strongly converged and stagnant regimes, the recommendation increases exploration by slightly raising the mutation rate and relaxing local search limits. In highly dispersed regimes with slow progress the recommendation favors exploitation by keeping the crossover rate high, slightly lowering the mutation rate, and intensifying local search. In balanced regimes only small adjustments around the current values are suggested. The raw suggestions are not used directly; instead, Algorithm 2 applies a post-processing that enforces trust regions, smooths the update, and guarantees global bounds (Step 3).

First, a generation-dependent trust region restricts the raw LLMs proposals to reasonable intervals that gradually narrow as the search progresses. For the crossover rate, a fixed trust interval is used,

$$p_g^{\text{cross,tr}} = \min \{0.95, \max \{0.80, \tilde{p}_g^{\text{cross}}\}\} \quad (28)$$

while the mutation rate employs a piecewise trust region that depends on the generation index,

$$(mr_{\min}(g), mr_{\max}(g)) = \begin{cases} (0.20, 0.35), & g < 100, \\ (0.15, 0.35), & 100 \leq g < 200, \\ (0.10, 0.40), & g \geq 200, \end{cases} \quad (29)$$

$$p_g^{\text{mut,tr}} = \min \{mr_{\max}(g), \max \{mr_{\min}(g), \tilde{p}_g^{\text{mut}}\}\} \quad (30)$$

The local search intensity is similarly restricted to a moderate range,

$$l_{s_{\max,g}}^{\text{tr}} = \min \{250, \max \{100, \tilde{l}_{s_{\max,g}}\}\} \quad (31)$$

The trust-region clipping in Eqs. (28)-(31), implemented in Step 3 of Algorithm 2, prevents overly aggressive moves suggested by the LLMs and reflects the intuition that, as the generation index increases, the algorithm gradually shifts from a very exploratory regime to a more stable, exploitation-oriented regime.

Second, the controller combines the trusted proposals with the previous parameters through a convex combination, which yields a smoothed update that avoids abrupt jumps. With a smoothing coefficient $\alpha = 0.1$, the next-generation parameters are computed as

$$p_{g+1}^{\text{cross}} = (1 - \alpha) p_g^{\text{cross}} + \alpha p_g^{\text{cross,tr}} \quad (32)$$

$$p_{g+1}^{\text{mut}} = (1 - \alpha) p_g^{\text{mut}} + \alpha p_g^{\text{mut,tr}} \quad (33)$$

$$l_{s_{\max,g+1}} = \left\lfloor (1 - \alpha) l_{s_{\max,g}} + \alpha l_{s_{\max,g}}^{\text{tr}} \right\rfloor \quad (34)$$

where $\lfloor \cdot \rfloor$ denotes rounding to the nearest integer. Finally, a global clipping step enforces hard bounds on the smoothed parameters,

$$\begin{aligned} p_{g+1}^{\text{cross}} &\in [0.50, 0.99] \\ p_{g+1}^{\text{mut}} &\in [0.01, 0.50] \end{aligned} \quad (35)$$

$$l_{s_{\max,g+1}} \in [50, 300]$$

The combination of the smoothed update in Eqs. (32)-(34) and the global bounds in Eq. (35), summarized in Step 3 of Algorithm 2, guarantees that the actual crossover rate, mutation rate, and local search budget used in the next generation remain within safe and interpretable ranges. The updated triplet $(p_{g+1}^{\text{cross}}, p_{g+1}^{\text{mut}}, l_{s_{\max,g+1}})$ then enters the offspring production strategy and the triggered local search mechanism

in generation $g + 1$, closing the feedback loop between the multidimensional search state and adaptive parameter control.

Algorithm 2 LLMs-based hyperparameter control

Input : Generation g ; state vector $\mathbf{s}_g$; current parameters; adaptation start $adapt_start_gen$; LS patience $ls_patience$

Output Updated parameters $(p_{g+1}^{cross}, p_{g+1}^{mut}, ls_{max,g+1})$:

```

1 /* default: keep parameters unchanged */
2  $p_{g+1}^{cross} \leftarrow p_g^{cross}$ ,
3  $p_{g+1}^{mut} \leftarrow p_g^{mut}$ ,
4  $ls_{max,g+1} \leftarrow ls_{max,g}$ 

5 /* Step 1: check adaptation trigger */
6 if  $g < adapt\_start\_gen$  then
7   return  $(p_{g+1}^{cross}, p_{g+1}^{mut}, ls_{max,g+1})$ 
8 end if
9 if not  $(g \bmod 20 = 0$  or  $(plateau_g \geq ls\_patience$  and
10   $g \bmod 5 = 0))$  then
11   return  $(p_{g+1}^{cross}, p_{g+1}^{mut}, ls_{max,g+1})$ 
12 end if

13 /* Step 2: query LLMs for raw suggestions */
14  $(\tilde{p}_g^{cross}, \tilde{p}_g^{mut}, \tilde{ls}_{max,g}) \leftarrow \text{QUERYLLM}(\mathbf{s}_g,$ 
15   $(p_g^{cross}, p_g^{mut}, ls_{max,g}))$ 
16 if LLMs call fails or output invalid then
17   return  $(p_{g+1}^{cross}, p_{g+1}^{mut}, ls_{max,g+1})$ 
18 end if

19 /* Step 3: trust region, smoothing, and global
20  bounds */
21  $(p_{g+1}^{cross}, p_{g+1}^{mut}, ls_{max,g+1}) \leftarrow$ 
22  POSTPROCESS( $\tilde{p}_g^{cross}, \tilde{p}_g^{mut}, \tilde{ls}_{max,g},$ 
23   $p_g^{cross}, p_g^{mut}, ls_{max,g}, g$ )
24 return  $(p_{g+1}^{cross}, p_{g+1}^{mut}, ls_{max,g+1})$ 

```

4.7 Computational complexity analysis

Let n denote the number of jobs, m the number of stages, l_k the number of parallel machines at stage k , N the population size, G the maximum number of generations, and L_s the maximum number of local-search iterations. In the proposed LLM-AEA, the main computational cost arises from individual decoding and fitness evaluation, population evolution, and the stagnation-triggered local search. For a chromosome, the decoding procedure assigns jobs to machines stage by stage according to the job permutation, and scans the available parallel machines at each stage to determine the earliest completion time. Therefore, the computational complexity

of decoding and fitness evaluation for a single individual is $O(\sum_{k=1}^m nl_k)$. In each generation, the computational complexities of selection, crossover, and mutation are $O(N)$, $O(Nn)$, and $O(N)$, respectively. Since the decoding and evaluation of all individuals in the new population dominate the total cost, the per-generation complexity can be written as $O(N \sum_{k=1}^m nl_k)$.

When the search becomes stagnant, the local search is triggered. Since each neighborhood search requires re-decoding and re-evaluating a candidate solution, the complexity of one local-search call is $O(L_s \sum_{k=1}^m nl_k)$. In addition, the computation of the multidimensional search-state features has complexity $O(Nn)$, whereas the LLMs-based hyperparameter control performs inference only on a fixed-dimensional state vector and can therefore be regarded as $O(1)$ in the asymptotic sense, which does not alter the dominant complexity of the algorithm. Accordingly, the overall time complexity of the proposed method can be expressed as

$$O\left(GN \sum_{k=1}^m nl_k + G_{ls} L_s \sum_{k=1}^m nl_k\right),$$

where G_{ls} denotes the number of local-search activations. It can thus be seen that the main computational bottleneck of the proposed method lies in the repeated decoding and evaluation of population individuals, together with the local intensification process during stagnant phases.

5 Experimental Results and Discussion

The detailed implementation of LLM-AEA is presented in Section 4. This section will demonstrate parameter settings, comparative experiments, and ablation experiments to evaluate the performance of LLM-AEA. All algorithms are implemented in Python 3.9.21 using Visual Studio Code on a Windows 10 operating system. Experiments are conducted on a system equipped with an Intel(R) Xeon(R) CPU E3-1246 v3, 32 GB RAM, and an NVIDIA CMP 30HX GPU with 6 GB memory. All experiments are carried out under identical hardware and software environments to ensure fair comparisons.

5.1 Design of Experiments

In this paper, HFSP test instances are generated based on [35], including 10 small-scale instances (20 jobs $\times$ 3 stages) and 10 large-scale instances (60 jobs $\times$ 5 stages). Five machines are available at each stage. Let $p_{j,k,i}$ denote the processing time of job j on machine i at stage k . All processing times are generated randomly by a computer and are independently sampled from a discrete uniform distribution:

$$p_{j,k,i} \sim U\{1, 50\}, \quad j = 1, \dots, n, \quad k = 1, \dots, m, \quad i = 1, \dots, l_k,$$

For each job j , the due date d_j is set to 1.5 times the sum of its average processing times over all stages:

$$d_j = 1.5 \sum_{k=1}^m \bar{p}_{j,k}, \quad \bar{p}_{j,k} = \frac{1}{l_k} \sum_{i=1}^{l_k} p_{j,k,i}. \quad (36)$$

In the experiments, ABC, DE, PSO, and LLM-AEA, the same permutation encoding and decoding procedure are used. TSCNSA retains its original representation-transfer mechanism, with its objective evaluation adapted to the TTD criterion. For each instance, each algorithm is independently run 20 times, and all algorithms are terminated after 200 iterations.

The experimental results are evaluated using the Relative Percentage Deviation (RPD) and the Average Relative Percentage Deviation (ARPD) [36]. The formulas for calculating RPD and ARPd in Eqs. 37 and 38.

$$\text{RPD} = \frac{(TTD - TTD^*)}{TTD^*} \times 100\% \quad (37)$$

where TTD is obtained by a certain algorithm, and TTD^* is the optimal solution obtained from all comparison algorithms in one instance.

$$\text{ARPd} = \frac{1}{R} \times \sum_{i=1}^R \text{RPD} \quad (38)$$

where R represents the number of times all algorithms are executed on a single instance.

5.2 Parameter setting

The designed LLM-AEA contains several parameters that may affect its search performance, including the population size, tournament size, local-search triggering patience, immigrant ratio, adaptation starting generation, and smoothing coefficient. Since the proposed method adopts an LLMs-based closed-loop controller to adaptively adjust the crossover rate, mutation rate, and local-search iterations during the search process, the parameter experiment mainly focuses on the initial and control-related parameters that define the population structure, selection pressure, stagnation response, diversity recovery, and update stability.

The Taguchi method is employed to calibrate these parameters. Six parameters are considered, and each parameter is assigned three candidate levels. Specifically, the population size P_s is set to $\{100, 200, 300\}$, the tournament size T_s is set to $\{2, 3, 5\}$, the local-search triggering patience L_{tp} is set to $\{10, 25, 40\}$, the immigrant ratio I_r is set to $\{0.05, 0.15, 0.25\}$, the adaptation starting generation A_{sg} is set to $\{40, 80, 120\}$, and the smoothing coefficient α is set to $\{0.1, 0.3, 0.5\}$. An orthogonal table L_{18} is designed for

the parameter experiment. For each parameter combination, the selected benchmark instances are independently solved by the DeepSeek-V4-based [37] LLM-AEA under the same stopping criterion, and the mean TTD value is used as the response value. Fig. 8 presents the main effect plots of the six parameters. Based on the main effect plots, the optimal parameter configuration of LLM-AEA is set as $P_s = 300$, $T_s = 2$, $L_{tp} = 40$, $I_r = 0.25$, $A_{sg} = 40$, and $\alpha = 0.1$. The detailed parameter settings of LLM-AEA are summarized in Table 3.

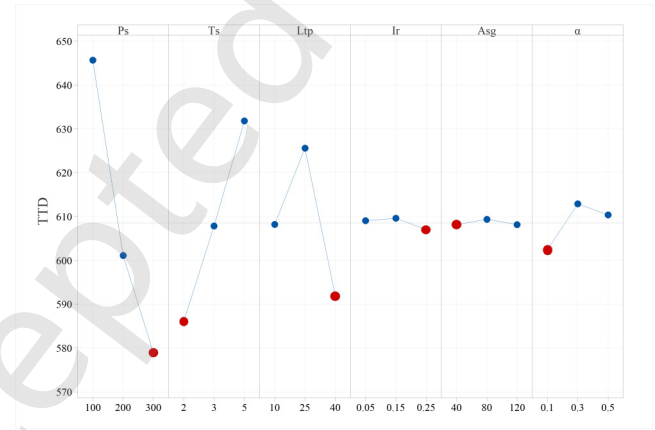


Fig. 8 Main effect plots of LLM-AEA parameters

Table 3 Parameter settings of LLM-AEA

Parameter	Value
Population size	300
Number of generations	200
Initial crossover rate	0.9
Initial mutation rate	0.25
Initial maximum local-search iterations	150
Tournament size	2
Local-search triggering patience	40
Immigrant ratio	0.25
Adaptation starting generation	40
Smoothing coefficient	0.1

5.3 Ablation Experiment

To verify the effectiveness of the key components in LLM-AEA, an ablation experiment is further conducted. All variants adopt the same encoding scheme, decoding procedure, parameter settings, termination criterion, and number of independent runs. Only the specified component is removed in each variant, so that its contribution to the overall performance can be examined. Specifically, V1 denotes the variant without the LLMs-driven online hyperparameter control mechanism; V2 denotes the variant without the tardy-job-guided neighborhood; V3 denotes the variant without diversity-related feature

inputs; V4 denotes the variant without the triggered local search mechanism; and V5 denotes the variant without the random immigrant mechanism. DV4, G4, and G5 denote the complete LLM-AEA using DeepSeek-V4 [37], GPT-4 [39], and GPT-5 [40] as the underlying LLMs, respectively. ARPD is adopted to evaluate the performance of different variants, where a smaller ARPD indicates that the obtained solution is closer to the best TTD found in the current experiments.

Fig. 9 presents the ARPD box plot of the ablation variants. It can be observed that the ARPD distributions of the three complete algorithms, namely DV4, G4, and G5, are generally lower than those of most ablation variants. This indicates that the complete LLM-AEA achieves better solution quality and stability across different instances. In contrast, the ARPD distributions of V2 and V3 shift upward noticeably. In particular, V3 shows a relatively high median and overall distribution, while V2 also exhibits a clear performance degradation. These results indicate that the tardy-job-guided neighborhood and the diversity-related state features are the two most influential components in LLM-AEA. For the HFSP with TTD minimization, the tardy-job-guided neighborhood can act more directly on the critical jobs that contribute to the increase in the objective value. Meanwhile, structural information such as population diversity, elite ratio, and positional variance helps the LLMs identify the current search state more accurately, thereby supporting more appropriate hyperparameter adjustment.

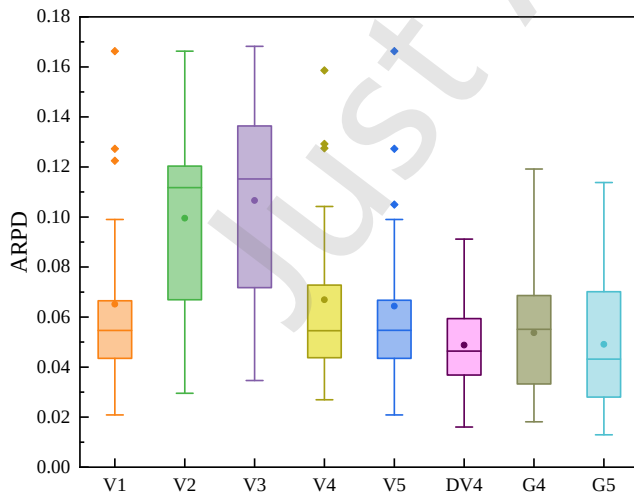


Fig. 9 The box plot of LLM-AEA and LLM-AEA variants

The results of V1, V4, and V5 further illustrate the auxiliary effects of the other components. After removing the LLMs-based online hyperparameter control, the ARPD distribution of V1 becomes higher than that of the complete algorithms, indicating that fixed parameters are less capable

of adapting to the exploration and exploitation requirements at different search stages. When the local search mechanism is removed, the performance of V4 also deteriorates, showing that the triggered local search plays an important role in intensifying the search during stagnant phases. Similarly, V5 performs worse than the complete algorithms after removing the random immigrant mechanism, which suggests that random immigrants can introduce new permutation structures when the population tends to converge, thereby alleviating the loss of diversity.

To further examine the overall performance differences among different variants, the Friedman test [41] is conducted on the ablation results, as reported in Table 4. The obtained p-value is smaller than 0.05, indicating that statistically significant differences exist among the algorithm variants. In terms of average rank, the complete LLM-AEA variants achieve the leading positions, whereas the variants without diversity-related feature inputs and without the tardy-job-guided neighborhood show the weakest rankings. This result further confirms that the complete framework maintains robust performance under different underlying LLMs, and that the diversity-related state representation and the tardiness-oriented neighborhood play critical roles in improving the search effectiveness.

Table 4 Friedman test results of LLM-AEA and LLM-AEA variants

Algorithm	Rank	p-value
V1	4.15	1.18E-14
V2	6.80	
V3	7.55	
V4	4.95	
V5	4.05	
DV4	2.90	
G4	3.10	
G5	2.50	

Overall, the ablation results verify that the effectiveness of LLM-AEA is achieved by the complementary design of closed-loop LLMs control, diversity-aware state representation, tardiness-guided local improvement, and diversity recovery mechanisms.

5.4 Comparison and Discussion

This subsection compares the proposed algorithm with seven competitors, including three classical metaheuristics, one recent state-of-the-art HFSP algorithm, and three LLM-AEA implementations based on different LLMs backbones. The classical baselines are ABC, Particle Swarm Optimization

(PSO), and Differential Evolution (DE) [38]. Since the HFSP considered in this study is solved by a permutation-based representation, all three baselines are implemented in the same discrete search space, where each solution is encoded as a job permutation and evaluated by the decoding procedure described in Section 4. In ABC, candidate solutions are treated as food sources and updated through permutation-preserving neighborhood search. In PSO, each particle is represented by a job permutation, and its update is guided by the personal-best and global-best solutions through discrete permutation transformations. In DE, mutation and crossover are adapted to the permutation domain by using permutation-preserving recombination operators to generate feasible offspring.

In addition, a recent state-of-the-art algorithm, namely TSCNSA, is included for comparison [33]. TSCNSA explores the search process through two successive stages, namely exploration and exploitation, and couples different solution representation spaces by transferring information across stages. It further employs cross-neighborhood interactions to generate guiding solutions in alternative representation spaces, thereby strengthening the coordination between global exploration and local refinement. Including TSCNSA makes it possible to assess the proposed method against not only classical metaheuristics but also a more advanced HFSP-oriented search framework.

Three complete instances of the LLM-AEA framework are also considered: DV4, G4, and G5. These three algorithms share the same LLM-AEA search framework, but differ in the underlying LLMs used for online hyperparameter control. This setting is used to examine whether the proposed framework can maintain robust performance under different LLMs backbones.

For fairness, all algorithms evaluate candidate schedules using the same decoding procedure and objective function. The classical baselines are implemented in the same permutation space as LLM-AEA, whereas TSCNSA preserves its original representation-transfer mechanism with necessary adaptation to the considered TTD objective. Each algorithm is independently executed 20 times on every benchmark instance, and ARPD is adopted to evaluate performance. Detailed numerical results and comparisons are reported in Table 5, Fig. 10, and Fig. 11.

As shown in Table 5, the three LLM-AEA variants obtain clearly smaller mean ARPD values than the classical metaheuristics and TSCNSA. On small-scale instances, G4, G5, and DV4 achieve competitive results, while DE obtains the best value on one instance. Nevertheless, the LLM-AEA variants show better overall performance across the majority

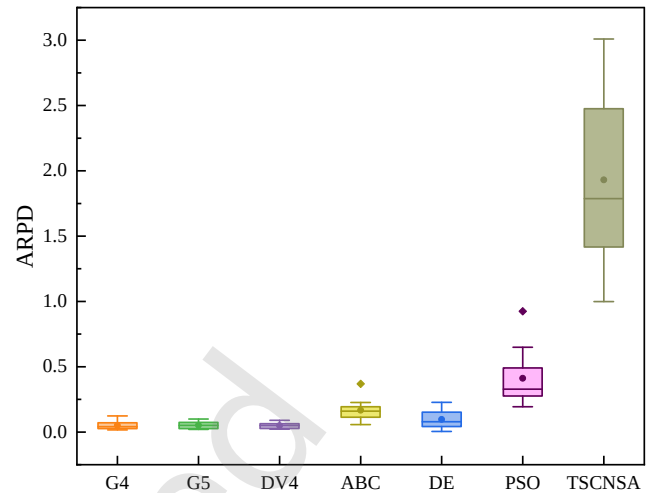


Fig. 10 ARPD box plot of seven algorithms on small-scale instances

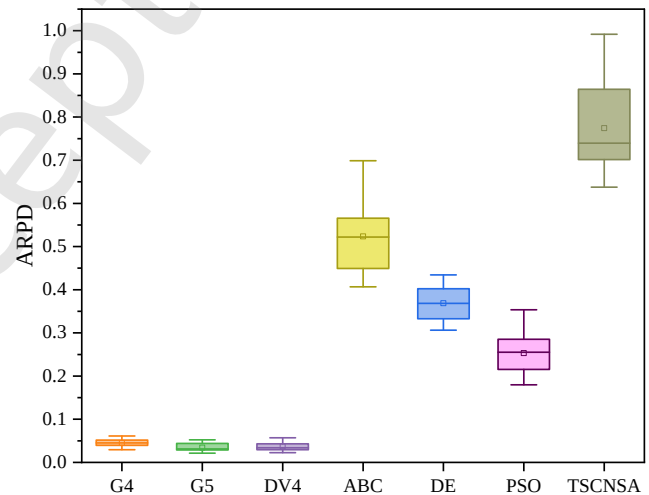


Fig. 11 ARPD box plot of seven algorithms on large-scale instances

of small-scale instances. On large-scale instances, the superiority of LLM-AEA becomes more evident, and the best ARPD values are consistently obtained by G4, G5, or DV4. These results indicate that the proposed closed-loop LLMs-based hyperparameter control is particularly effective when the problem scale increases and the search space becomes more complex.

The ARPD distributions in Fig. 10 and Fig. 11 further confirm this observation. On small-scale instances, the three LLM-AEA variants have lower and more compact ARPD distributions than ABC, PSO, and TSCNSA, indicating better robustness across different instances. Although DE performs competitively on some small-scale cases, its distribution is still generally higher than those of the LLM-AEA variants. On large-scale instances, the distributions of G4, G5, and DV4 remain concentrated near the lower ARPD region, whereas ABC, DE, PSO, and TSCNSA show much larger deviations.

Table 5 ARPD values in comparison algorithms

Instance	G4	G5	DV4	ABC	DE	PSO	TSCNSA
Small1	0.0164	0.0286	0.0229	0.1535	0.0927	0.1936	1.7860
Small2	0.0354	0.0261	0.0405	0.1773	0.0407	0.3531	1.7888
Small3	0.0316	0.0708	0.0489	0.1933	0.2277	0.4371	2.4755
Small4	0.0711	0.0741	0.0526	0.1172	0.0661	0.2781	0.9990
Small5	0.0259	0.0207	0.0276	0.0980	0.0043	0.2758	1.5920
Small6	0.0707	0.0581	0.0690	0.1131	0.0749	0.3035	1.4170
Small7	0.1229	0.0992	0.0892	0.1671	0.1518	0.9249	3.0090
Small8	0.0625	0.0763	0.0641	0.2268	0.1866	0.4911	2.5514
Small9	0.0250	0.0261	0.0227	0.0575	0.0425	0.2058	1.3841
Small10	0.0472	0.0447	0.0513	0.3687	0.0827	0.6489	2.3018
Large1	0.0518	0.0455	0.0570	0.5209	0.3793	0.2624	0.7331
Large2	0.0612	0.0441	0.0349	0.4492	0.3063	0.1800	0.7070
Large3	0.0299	0.0318	0.0321	0.5231	0.3460	0.2483	0.7460
Large4	0.0435	0.0255	0.0350	0.6129	0.3955	0.3091	0.8642
Large5	0.0295	0.0309	0.0432	0.6988	0.4344	0.3537	0.9916
Large6	0.0486	0.0290	0.0295	0.5540	0.4195	0.2750	0.8191
Large7	0.0456	0.0314	0.0558	0.4878	0.3575	0.2222	0.7016
Large8	0.0450	0.0523	0.0289	0.4068	0.3327	0.2153	0.6376
Large9	0.0534	0.0214	0.0226	0.5656	0.4024	0.2854	0.8717
Large10	0.0394	0.0292	0.0329	0.4170	0.3136	0.1797	0.6710
Mean	0.0478	0.0433	0.0430	0.3454	0.2329	0.3321	1.3524

This suggests that the adaptive parameter-control mechanism and stagnation-handling strategy of LLM-AEA can better maintain search effectiveness and stability on large-scale HFSP instances.

Table 6 The average rank of comparison algorithms in two scales

Algorithm	Small		Large	
	Rank	p-value	Rank	p-value
TSCNSA	7.00		7.00	
PSO	6.00		4.00	
DE	3.60		5.00	
ABC	4.90	$2.24E - 09$	6.00	$2.28E - 10$
DV4	2.10		2.20	
G5	2.30		1.50	
G4	2.10		2.30	

As shown in Table 6, the p-values on both small-scale and large-scale instances are smaller than 0.05, indicating that statistically significant differences exist among the compared algorithms. In terms of average rank, the three LLM-AEA variants consistently obtain the leading positions on both scales. In particular, DV4 and G4 obtain the best ranks on small-scale instances, while G5 achieves the best rank on large-scale instances. This indicates that the proposed framework is not sensitive to a specific underlying LLMs and can maintain competitive performance across different LLMs backbones.

To further analyze the statistical differences, the

Bonferroni–Dunn test is used as a post-hoc test. The critical difference (CD) is computed according to Eq. (39),

$$CD = q_{\alpha} \sqrt{\frac{k(k+1)}{6N}} \quad (39)$$

where k denotes the number of compared algorithms and N denotes the number of experimental datasets used for ranking [42]. The corresponding CD diagrams under 95% and 90% confidence levels are shown in Fig. 12 and Fig. 13. On small-scale instances, the LLM-AEA variants are located in the best-rank region, while most comparison algorithms are placed in higher-rank regions. On large-scale instances, the ranking advantage of the LLM-AEA variants becomes more pronounced, and G5 obtains the best average rank. These results further support the conclusion that LLM-AEA provides statistically reliable and robust performance advantages over the baseline algorithms, especially on larger-scale HFSP instances.

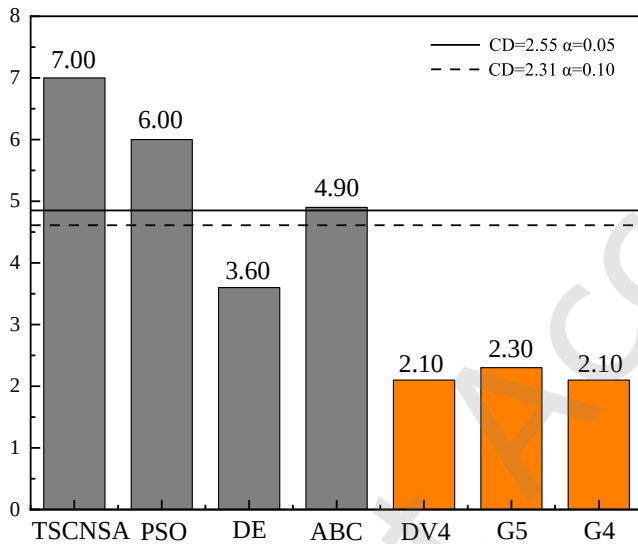
6 Further analyses

6.1 Computational Time Analysis

Computational time is an important supplementary indicator for evaluating scheduling algorithms, because an algorithm with high solution quality may still be difficult to apply in time-sensitive scenarios if its decision-making time is excessive. Therefore, this subsection further compares the CPU time [43] and LLMs invocation time of all algorithms

Table 7 Comparison of CPU time and LLMs invocation time among all algorithms

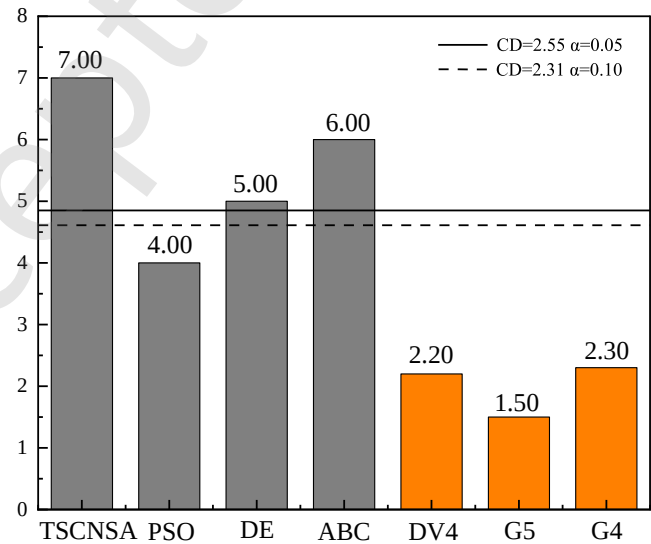
Instance	ABC	DE	PSO	TSCNSA	G4_All	G4_api	G5_All	G5_api	DV4_All	DV4_api
Small1	16.6885	11.3686	13.5144	0.1086	112.8714	94.4301(83.66%)	68.4618	49.4759(72.27%)	42.3306	23.2479(54.92%)
Small2	16.8237	11.3812	13.4624	0.0948	47.8541	30.2876(63.29%)	65.8587	48.0117(72.90%)	38.3527	20.6751(53.91%)
Small3	16.6865	11.4798	13.3953	0.0986	46.9994	29.7275(63.25%)	63.9340	45.9261(71.83%)	37.2028	19.8435(53.34%)
Small4	16.6891	11.3288	13.4491	0.0972	43.3306	27.1700(62.70%)	58.2582	41.4321(71.12%)	33.8506	17.6412(52.11%)
Small5	16.7061	11.3508	13.5196	0.1011	46.1329	29.3215(63.56%)	54.5814	38.2429(70.07%)	36.8771	19.8081(53.71%)
Small6	16.6584	11.2559	13.3859	0.1010	49.0329	31.3687(63.97%)	55.9116	38.9983(69.75%)	39.4546	21.7990(55.25%)
Small7	16.6895	11.3508	13.4643	0.0984	50.5571	32.1409(63.57%)	62.0316	43.7711(70.56%)	40.1609	22.2550(55.41%)
Small8	16.6843	11.3610	13.4816	0.0975	90.4637	72.2204(79.83%)	64.8019	46.3732(71.56%)	40.0536	22.0534(55.06%)
Small9	16.7189	11.3739	13.5479	0.1022	83.1193	64.5089(77.61%)	71.0111	52.4040(73.80%)	41.7582	23.0654(55.24%)
Small10	16.7055	11.3660	13.4981	0.0998	96.4241	77.7685(80.65%)	62.8932	44.4659(70.70%)	41.3565	22.7403(54.99%)
Large1	67.5262	41.4812	49.8688	0.4270	64.5402	18.0343(27.94%)	62.2721	16.3941(26.33%)	54.9288	8.4134(15.32%)
Large2	67.6022	41.5060	49.9441	0.4242	65.8570	19.4088(29.47%)	62.1623	16.1524(25.98%)	55.0841	8.6328(15.67%)
Large3	67.5487	41.4757	49.8617	0.4435	66.6268	20.3419(30.53%)	64.9963	19.1680(29.49%)	54.7260	8.3082(15.18%)
Large4	67.5676	41.5473	50.0045	0.4037	66.6384	20.3158(30.49%)	62.0467	15.9432(25.70%)	54.6004	8.2458(15.10%)
Large5	67.6735	41.5122	49.8046	0.3954	65.9314	19.4235(29.46%)	62.9445	16.7624(26.63%)	54.7599	8.3426(15.23%)
Large6	67.7146	41.5348	50.0490	0.4825	68.3458	21.8758(32.01%)	63.3734	17.1821(27.11%)	54.5728	7.9986(14.66%)
Large7	67.5315	41.4575	50.0124	0.4135	68.3934	21.9558(32.10%)	61.6862	15.4036(24.97%)	54.3603	7.9345(14.60%)
Large8	67.6253	41.4985	49.9815	0.4120	72.8159	26.1916(35.97%)	62.0385	15.6732(25.26%)	54.3771	7.9109(14.55%)
Large9	67.6047	41.5351	50.0819	0.4179	76.9710	30.3921(39.49%)	62.1487	15.8275(25.47%)	54.3076	7.9279(14.60%)
Large10	67.5835	41.5512	49.9971	0.4128	76.9899	30.3399(39.41%)	62.7340	16.4805(26.27%)	54.3082	7.8832(14.52%)

**Fig. 12** Bonferroni–Dunn test results on small-scale instances

to examine the computational overhead introduced by the LLMs-based online control mechanism.

The computational time results are reported in Table 7, and all time values are measured in seconds. In this table, G4_All, G5_All, and DV4_All represent the total elapsed running time of G4, G5, and DV4, respectively, while G4_api, G5_api, and DV4_api represent the accumulated waiting time for API responses during the corresponding algorithm runs. The percentages shown in parentheses after G4_api, G5_api, and DV4_api indicate the proportion of the API waiting time relative to the corresponding total elapsed running time.

For the non-LLMs comparison algorithms, TSCNSA has the shortest running time, whereas DE, PSO, and ABC require more computation because they perform population-based iterative search. However, the computational time should be considered together with the solution quality reported above.

**Fig. 13** Bonferroni–Dunn test results on large-scale instances

Although G4, G5, and DV4 require additional elapsed time due to API invocation, they achieve better overall optimization performance on the tested HFSP instances.

For G4, G5, and DV4, the total elapsed time consists of two parts: the local evolutionary computation time and the API waiting time. The results show that the CPU-side running times of these three algorithms are very close after excluding the API component. This indicates that the choice of the underlying LLMs mainly affects the response latency rather than the computational burden of the evolutionary search itself. Among the three algorithms, DV4 has the shortest total running time, mainly because its API waiting time is lower than those of G4 and G5.

The results also show that the relative contribution of API waiting time decreases as the instance scale increases. For small-scale instances, API invocation accounts for a relatively

high proportion of the total elapsed time. This is mainly because the algorithm can converge more quickly on smaller instances, which increases the frequency of API-triggered parameter adjustment and thus makes the API waiting time more prominent in the total running time. In contrast, for large-scale instances, the search process is less likely to converge prematurely, so the number of API invocations does not increase significantly compared with small-scale instances. As a result, schedule decoding, fitness evaluation, and local search become the dominant time-consuming components.

6.2 Results of Liao's benchmark instances

To further verify the applicability of the proposed algorithm to classical large-scale HFSP instances, Liao's benchmark instances [44] are used for supplementary experiments in this subsection. The Liao benchmark set is a commonly used test set in HFSP research. It consists of 10 instances, each with 30 jobs and 5 processing stages. The instances are named in the form of "j*c*e*", where "j" denotes the number of jobs, "c" denotes the number of stages, and "e" denotes the machine configuration and instance index under this scale. For example, instance j30c5e1 indicates that the instance contains 30 jobs and 5 stages, while the final "1" denotes the first instance under this machine-configuration type. Similarly, j30c5e10 represents the tenth instance under the same scale. For each instance, the due date data are calculated according to Eq. (36).

It should be noted that the optimization objective of the proposed algorithm remains TTD. That is, makespan is not directly optimized during the search process. To maintain comparability with existing HFSP studies, the makespan corresponding to each schedule is recorded simultaneously when calculating TTD and is analyzed in this subsection as a supplementary performance metric.

Table 8 reports the makespan results obtained by all algorithms on the 10 Liao benchmark instances. It can be observed that the three LLM-AEA algorithms based on different LLMs perform well on this benchmark set and obtain the smallest makespan on most instances. Among them, DV4 shows particularly strong performance on several instances, while G5 and G4 also achieve competitive results on some instances. In contrast, although ABC, DE, PSO, and TSCNSA obtain good makespan values on individual instances, their overall advantages are less stable than those of LLM-AEA. This indicates that the proposed LLM-AEA can still generate schedules with smaller makespan values even when TTD is used as the primary optimization objective.

6.3 Results of Carlier's benchmark instances

To further examine the applicability of the proposed algorithm to classical HFSP test sets, this subsection conducts supplementary experiments on Carlier benchmark instances [45]. The Carlier benchmark is a widely used standard test set in HFSP, and its original setting is generally evaluated with makespan minimization as the main optimization objective. The benchmark contains instances with different numbers of jobs, stages, and machine configurations, and can therefore be used to assess scheduling performance under different problem scales and resource structures. The instance naming rule is similar to Liao's benchmark, and the due date is calculated according to Eq. (36).

It should be noted that the proposed LLM-AEA is mainly designed for TTD minimization, whereas Carlier benchmark instances are commonly evaluated using makespan. Therefore, the makespan results reported in this subsection are used as a supplementary indicator to analyze the general scheduling capability of the algorithm on classical HFSP benchmarks, rather than as a direct measure of optimality with respect to the objective function studied in this paper. The experimental results of all algorithms on Carlier benchmark instances are reported in Table 9.

As shown in Table 9, different algorithms exhibit different performance patterns on the Carlier benchmark instances. TSCNSA achieves competitive makespan results on this test set and obtains superior solutions on many instances. This is mainly because its search mechanism is well aligned with the makespan minimization objective emphasized by the Carlier benchmark. Meanwhile, G4, G5, and DV4 also achieve competitive makespan results on some instances. This indicates that although the proposed evolutionary search framework is not specifically designed for makespan minimization, it still demonstrates a certain degree of adaptability to classical HFSP benchmarks.

The experimental results in this subsection provide supplementary evidence for the generalization capability of LLM-AEA. Although this benchmark focuses more on makespan minimization and differs from the TTD objective mainly investigated in this paper, LLM-AEA can still obtain competitive makespan results on several instances and maintain relatively consistent performance across different LLMs backends. These results indicate that the proposed method is not limited to the electrolytic aluminum production scheduling instances constructed in this study, but can also be applied to standard HFSP benchmark scenarios.

Table 9 Makespan in Carlier's benchmark instances

Instance	G4	G5	DV4	ABC	DE	PSO	TSCNSA
j10c5a2	98.00	98.00	98.00	98.00	98.00	98.00	94.50
j10c5a3	124.00	124.00	124.00	124.00	124.00	124.15	122.50
j10c5a4	132.75	132.75	132.75	134.80	134.15	134.25	130.90
j10c5a5	135.00	135.00	135.00	135.00	135.00	135.00	131.10
j10c5a6	117.50	117.50	117.50	113.50	118.00	120.00	113.20
j10c5b1	151.50	151.50	151.50	147.00	148.50	152.25	141.40
j10c5b2	111.00	111.00	111.00	111.00	111.00	111.00	111.00
j10c5b3	119.00	119.00	119.00	119.00	119.00	119.00	119.05
j10c5b4	137.00	137.00	137.00	137.00	137.00	137.00	137.40
j10c5b5	159.55	159.55	159.55	159.80	160.15	160.25	155.10
j10c5b6	116.00	116.00	116.00	116.40	116.40	116.10	115.30
j10c5b7	134.00	134.00	134.00	134.00	134.00	134.00	134.00
j10c5c1	81.70	81.70	81.70	82.00	81.55	81.20	79.75
j10c5c2	80.95	80.95	80.95	79.20	80.20	79.75	78.65
j10c5c4	84.05	84.05	84.05	80.65	83.00	84.55	72.95
j10c5c5	86.00	86.00	86.00	86.00	86.00	91.70	89.70
j10c5c6	80.55	80.55	80.55	80.75	81.95	80.85	75.15
j10c5d1	80.70	80.70	80.70	80.05	80.05	79.20	68.70
j10c5d2	84.75	84.75	84.75	84.60	84.60	85.65	76.75
j10c5d3	76.75	76.75	76.75	77.40	77.40	77.10	71.65
j10c5d4	80.15	80.15	80.15	80.30	80.35	79.95	75.80
j10c5d5	75.95	75.95	75.95	75.75	75.40	76.75	70.15
j10c5d6	71.90	71.90	71.90	72.50	72.35	71.95	66.45
j10c5d7	76.95	76.95	76.95	80.10	80.10	78.95	68.00
j10c10a1	155.95	155.95	155.95	155.40	155.10	155.40	150.10
j10c10a2	187.00	187.00	187.00	187.00	187.00	187.00	187.05
j10c10a3	165.00	165.00	165.00	165.00	165.00	165.00	166.30
j10c10a4	154.00	154.00	154.00	154.00	154.00	154.00	156.70
j10c10a5	163.00	163.00	163.00	163.00	163.00	163.00	163.00
j10c10a6	162.90	162.90	162.90	161.40	161.40	162.00	159.35
j10c10b1	180.90	180.90	180.90	183.00	182.30	183.00	180.90
j10c10b2	176.00	176.00	176.00	176.00	176.00	176.00	176.00
j10c10b3	187.00	187.00	187.00	187.00	187.00	187.00	186.70
j10c10b4	173.00	173.00	173.00	173.00	173.00	173.00	172.25
j10c10b5	190.00	190.00	190.00	190.00	190.00	190.00	189.30
j10c10b6	184.00	184.00	184.00	184.00	184.00	184.00	184.60
j10c10c1	130.50	130.50	130.50	131.05	131.05	129.40	116.85
j10c10c2	136.65	136.65	136.65	140.55	140.55	140.50	121.95

Continued on next page

Table 9 Makespan in Carlier's benchmark instances continued

Instance	G4	G5	DV4	ABC	DE	PSO	TSCNSA
j10c10c3	135.15	135.15	135.15	134.20	134.20	133.50	118.30
j10c10c4	137.35	137.35	137.35	136.45	136.45	134.75	122.60
j10c10c5	147.55	147.55	147.55	146.95	146.95	147.45	129.05
j10c10c6	122.20	122.20	122.20	120.35	120.35	120.30	108.20
j10c10d1	124.30	124.30	124.30	123.00	123.00	123.50	113.10
j10c10d2	138.35	138.35	138.35	135.80	135.80	135.05	120.65
j10c10d3	133.45	133.45	133.45	133.20	133.20	133.80	117.00
j10c10d4	134.10	134.10	134.10	134.70	134.70	133.50	117.05
j10c10d5	125.35	125.35	125.35	126.45	126.45	125.65	110.35
j10c10d6	120.05	120.05	120.05	122.65	122.65	122.90	108.05
j15c5a1	191.00	191.00	191.00	189.75	189.50	190.00	186.85
j15c5a2	167.00	167.00	167.00	167.00	167.00	167.00	167.95
j15c5a3	135.00	135.00	135.00	135.00	135.00	135.30	135.45
j15c5a4	162.00	162.00	162.00	162.00	162.00	161.85	161.55
j15c5a5	170.85	170.85	170.85	170.85	170.85	173.25	172.10
j15c5a6	192.00	192.00	192.00	192.00	192.00	191.85	188.85
j15c5b1	181.00	181.00	181.00	181.00	181.00	181.00	179.25
j15c5b2	168.00	168.00	168.00	168.00	168.00	168.00	167.00
j15c5b3	164.95	164.95	164.95	166.60	165.50	169.35	161.95
j15c5b4	157.00	157.00	157.00	158.50	158.00	159.50	157.45
j15c5b5	190.00	190.00	190.00	190.00	190.00	190.00	187.15
j15c5b6	191.60	191.60	191.60	193.15	193.00	194.75	187.65
j15c5c1	99.15	99.45	99.45	98.50	98.05	99.85	96.00
j15c5c2	98.55	98.50	98.50	98.30	98.80	98.55	98.55
j15c5c3	104.60	104.15	104.25	102.00	102.30	103.20	103.05
j15c5c4	101.80	101.25	101.25	102.00	102.15	102.25	99.60
j15c5c5	85.40	85.55	85.55	85.10	85.75	84.20	82.80
j15c5c6	103.05	103.25	103.10	101.75	102.40	104.50	102.60
j15c5d1	167.00	167.00	167.00	167.00	167.00	167.00	167.85
j15c5d2	108.70	108.70	108.70	106.25	105.30	107.35	102.20
j15c5d3	95.20	95.15	95.25	94.20	93.40	94.60	91.35
j15c5d4	98.65	98.65	98.65	98.15	97.85	98.70	93.75
j15c5d5	103.10	103.10	103.10	102.75	102.50	101.90	98.25
j15c5d6	96.00	95.65	95.95	94.15	95.85	93.90	92.25
j15c5e1	167.00	167.00	167.00	167.00	167.00	167.00	167.85
j15c5e2	154.00	154.00	154.00	154.00	154.00	154.00	154.25
j15c5e3	176.00	176.00	176.00	176.00	176.00	176.10	176.65
j15c5e4	164.00	164.00	164.00	164.00	164.00	164.30	165.40

Continued on next page

Table 8 Makespan in Liao's benchmark instances

Instance	G4	G5	DV4	ABC	DE	PSO	TSCNSA
j30c5e1	359.15	366.05	358.95	381.75	384.95	372.65	403.15
j30c5e2	417.00	414.80	418.25	426.85	423.85	418.05	433.40
j30c5e3	378.75	376.25	370.15	378.95	381.30	377.45	389.60
j30c5e4	386.60	387.30	384.75	401.50	402.50	389.90	407.05
j30c5e5	434.35	438.40	440.50	463.40	467.15	448.15	483.40
j30c5e6	310.50	309.35	310.60	317.25	327.45	311.50	323.65
j30c5e7	322.60	324.10	321.75	319.25	320.55	319.15	325.30
j30c5e8	386.55	382.55	387.70	432.75	430.35	414.70	427.70
j30c5e9	264.45	257.65	254.45	271.85	283.55	270.75	280.20
j30c5e10	316.40	315.90	311.95	326.10	332.85	323.70	339.35

Table 9 Makespan in Carlier's benchmark instances continued

Instance	G4	G5	DV4	ABC	DE	PSO	TSCNSA
j15c5e5	146.00	146.00	146.00	146.00	146.00	146.15	146.75
j15c5e6	159.00	159.00	159.00	159.00	159.00	159.75	160.10
j15c10a1	249.35	249.35	249.35	248.75	249.05	249.05	250.05
j15c10a2	235.00	235.00	235.00	235.00	235.00	235.00	230.55
j15c10a3	240.25	240.25	240.25	234.05	235.20	228.65	228.60
j15c10a4	245.00	245.00	245.00	245.10	245.30	246.50	239.20
j15c10a5	210.00	210.00	210.00	210.00	210.00	209.80	206.55
j15c10a6	234.40	234.40	234.40	237.15	236.05	238.25	232.05
j15c10b1	222.00	222.00	222.00	222.00	222.00	222.05	222.90
j15c10b2	187.00	187.00	187.00	187.00	187.00	187.20	188.25
j15c10b3	222.00	222.00	222.00	222.00	222.00	222.30	222.30
j15c10b4	221.00	221.00	221.00	221.00	221.00	221.00	221.40
j15c10b5	200.00	200.00	200.00	200.00	200.00	200.05	201.95
j15c10b6	219.00	219.00	219.00	219.00	219.00	219.00	219.50
j15c10c1	161.15	161.15	161.15	160.90	160.15	161.10	153.65
j15c10c2	155.70	155.70	155.70	156.50	155.70	155.90	147.15
j15c10c3	162.55	162.55	162.55	164.60	164.85	164.45	164.50
j15c10c4	156.40	156.40	156.40	155.40	155.40	155.65	150.20
j15c10c5	164.90	164.90	164.90	165.10	165.20	165.50	162.20
j15c10c6	150.15	150.15	150.15	149.70	148.85	148.35	144.35

7 Conclusions

This study formulated the HFSP motivated by electrolytic aluminum production under a TTD objective and presented an MILP model to formalize assignment, sequencing, and non-overlap constraints with unrelated parallel machines. On this basis, a large Language Model-assisted evolutionary algorithm, LLM-AEA, was developed using permutation encoding and stage-by-stage decoding to generate feasible schedules efficiently.

LLM-AEA combined evolutionary search, a stagnation-triggered local search, and a closed-loop LLMs controller that adjusted crossover rate, mutation rate, and local-search budget using population-level signals on fitness, stagnation, and permutation diversity. Trust-region clipping and smoothed updates stabilized parameter changes, while random immigrants supported diversity recovery. Experiments on small and large instances showed consistent improvements over comparison methods, with robust performance across different base LLMs.

Future research can extend the model toward richer plant constraints and dynamic settings with disruptions, and can explore multiobjective variants that incorporate energy and robustness while maintaining scalable online control.

Acknowledgment

This study was financially supported by the National Natural Science Foundation of China under Grant No.62473182, No.62563022. It was also supported by the Industry Support Project of Gansu Province College under Grant 2024CYZC-15, the Intellectual Property Program of Gansu Province under Grant 24ZSCQG045, the Science and Technology Program Funding of Gansu Province under Grant 26CXGA008, and the Lanzhou Science and Technology Planning Project under Grant 2025-3-019.

Declaration of competing interest

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

Reference

- [1] Kurian J Vachaparambil, Stein Tore Johansen, Ryan M Soncini, Asbjørn Solheim, Kristian Etienne Einarsrud, Kim Ronny Elstad, Jayson Tessier, Modelling Alumina Feeding and Transport in an Industrial Aluminium Reduction Cell Using a Pragmatic Computational Model, *Metallurgical and Materials Transactions B*, 2025, 56, 2, 1272-1288.
- [2] Fuqing Zhao, Cheng Zhao, Ling Wang, Hongyan Sang, A Deep Reinforcement Learning Framework Assisted by Genetic Programming for Dynamic Flexible Job Shop Scheduling, *IEEE Transactions on Evolutionary Computation*, 2025, 1-1.
- [3] Fuqing Zhao, Fumin Yin, Jianlin Zhang, Tian Peng Xu, A learning-based co-evolution optimization framework for energy-aware distributed heterogeneous flexible flow shop lot-streaming scheduling problem, *Expert Systems with Applications*, 2026, 296, 128986.
- [4] Cong Luo, Xinyu Li, Wenying Gong, Liang Gao, Affinity propagation hierarchical memetic algorithm for multimodal multi-objective flexible job shop scheduling with variable speed, *IEEE Transactions on Evolutionary Computation*, 2025.
- [5] Rui Li, Ling Wang, Hongyan Sang, Lizhong Yao, Lijun Pan, LLM-Assisted Automatic Memetic Algorithm for Lot-Streaming Hybrid Job Shop Scheduling With Variable Sublots, *IEEE Transactions on Evolutionary Computation*, 2025.
- [6] Rui Zhong, Abdelazim G Hussien, Jun Yu, Masaharu Mune-tomo, LLMOA: A novel large language model assisted hyper-heuristic optimization algorithm, *Advanced Engineering Informatics*, 2025, 64, 103042.
- [7] Hao Hao, Xiaoqun Zhang, Aimin Zhou, Large language models as surrogate models in evolutionary algorithms: A preliminary study, *Swarm and Evolutionary Computation*, 2024, 91, 101741.
- [8] Yidan Wang, Jiayin Wang, Zhiwei Chu, Multi-agent large language models as evolutionary optimizers for scheduling optimization, *Computers & Industrial Engineering*, 2025, 206, 111197.
- [9] Youshan Liu, Jiaxin Fan, Weiming Shen, A deep reinforcement learning approach with graph attention network and multi-signal differential reward for dynamic hybrid flow shop scheduling problem, *Journal of Manufacturing Systems*, 80, 2025, 643-661.
- [10] Xiaoyu Ren, Jian Zhang, Jiwei Li, et al., A hierarchical strategy-optimized double deep Q-network for dynamic reentrant hybrid flow-shop scheduling problem with multi-stage batch processing machines, *Computers & Industrial Engineering*, 2025, 111630.
- [11] Siqi Liu, Haiping Zhu, LieZheng Sheng, Integrated dynamic scheduling method for hybrid flow shop with machine preventive maintenance based on cooperative multi-agent deep reinforcement learning, *Robotics and Computer-Integrated Manufacturing*, 2026, 97, 103085.
- [12] Lin Luo, Xuesong Yan, Scheduling of stochastic distributed hybrid flow-shop by hybrid estimation of distribution algorithm and proximal policy optimization, *Expert Systems with Applications*, 2025, 271, 126523.
- [13] Yuanzhu Di, Libao Deng, Ling Wang, et al., Real-Time Scheduling for Energy-Efficient Distributed Hybrid Flow-Shop With Evolution Strategies-Based Multiple Graph Neural Networks, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2025.



- [14] Yang Zuo, Fuqing Zhao, Yang Yu, Two-stage learning scatter search algorithm for the distributed hybrid flow shop scheduling problem with machine breakdown, *Expert Systems with Applications*, 2025, 259, 125344.
- [15] Ze-Cheng Wang, Quan-Ke Pan, Liang Gao, et al., A variable-representation discrete artificial bee colony algorithm for a constrained hybrid flow shop, *Expert Systems with Applications*, 2024, 254, 124349.
- [16] Yarong Chen, Jinhao Du, Jabir Mumtaz, et al., An efficient Q-learning integrated multi-objective hyper-heuristic approach for hybrid flow shop scheduling problems with lot streaming, *Expert Systems with Applications*, 2025, 262, 125616.
- [17] Jing Wang, Deming Lei, Hongtao Tang, A multi-objective dynamical artificial bee colony for energy-efficient fuzzy hybrid flow shop scheduling with batch processing machines, *Expert Systems with Applications*, 2025, 259, 125244.
- [18] Huan Liu, Fuqing Zhao, Ling Wang, et al., Evolutionary multitasking memetic algorithm for distributed hybrid flow-shop scheduling problem with deterioration effect, *IEEE Transactions on Automation Science and Engineering*, 2024, 22, 1390-1404.
- [19] Dandan Liu, Zhiyuan Zou, Xu Liang, A multi-strategy iterated greedy algorithm with three-phase for hybrid flow shop scheduling in distributed factory, *Applied Soft Computing*, 2025, 113475.
- [20] Yiping Huang, Libao Deng, Jianlei Wang, Weiwei Qiu, Jinfeng Liu, Modeling and solution for hybrid flow-shop scheduling problem by two-stage stochastic programming, *Expert Systems with Applications*, 2023, 233, 120846.
- [21] David Anunay Alexander, Avishek Pandey, Sri Krishna Kumar, Mathematical models for multi-stage hybrid assembly flow-shop scheduling with preventive maintenance and release times, *Computers & Industrial Engineering*, 2023, 186, 109716.
- [22] Fei Yu, Chao Lu, Jiajun Zhou, Lvjiang Yin, Mathematical model and knowledge-based iterated greedy algorithm for distributed assembly hybrid flow shop scheduling problem with dual-resource constraints, *Expert Systems with Applications*, 2024, 239, 122434.
- [23] Fei Yu, Chao Lu, Lvjiang Yin, Jiajun Zhou, Modeling and optimization algorithm for energy-efficient distributed assembly hybrid flowshop scheduling problem considering worker resources, *Journal of Industrial Information Integration*, 2024, 40, 100620.
- [24] Sara Taguemount, Damien Lamy, Xavier Delorme, et al., Multi-line hybrid flow-shop scheduling problem with energy considerations, *International Journal of Production Research*, 2025, 63(9), 3231-3253.
- [25] Weishi Shao, Zhongshi Shao, Dechang Pi, et al., Energy-Efficiency Oriented Distributed Heterogeneous Hybrid Flow Shop Scheduling With Multilevelled Mixed-Model Assembly, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2025.
- [26] Chao Lu, Jiajun Zhou, Liang Gao, et al., Modeling and multi-objective optimization for energy-aware scheduling of distributed hybrid flow-shop, *Applied Soft Computing*, 2024, 156, 111508.
- [27] Chenfei Liu, Qiang Cheng, Chenyang Cheng, et al., Enhancing energy efficiency in distributed hybrid forging flow-shops: a novel scheduling approach to capitalise on residual forging heat, *International Journal of Production Research*, 2024.
- [28] Jing-jing Wang, Ling Wang, A bi-population cooperative memetic algorithm for distributed hybrid flow-shop scheduling, *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2020, 5(6), 947-961.
- [29] Jing-Jing Wang, Ling Wang, A cooperative memetic algorithm with learning-based agent for energy-aware distributed hybrid flow-shop scheduling, *IEEE Transactions on Evolutionary Computation*, 2021, 26(3), 461-475.
- [30] Hao-Xiang Qin, Yu-Yan Han, Biao Zhang, Lei-Lei Meng, Yi-Ping Liu, Quan-Ke Pan, Dun-Wei Gong, An improved iterated greedy algorithm for the energy-efficient blocking hybrid flow shop scheduling problem, *Swarm and Evolutionary Computation*, 2022, 69, 100992.
- [31] Weishi Shao, Zhongshi Shao, Dechang Pi, Multi-objective evolutionary algorithm based on multiple neighborhoods local search for multi-objective distributed hybrid flow shop scheduling problem, *Expert Systems with Applications*, 2021, 183, 115453.
- [32] Jingcao Cai, Deming Lei, A cooperated shuffled frog-leaping algorithm for distributed energy-efficient hybrid flow shop scheduling with fuzzy processing time, *Complex & Intelligent Systems*, 2021, 7(5), 2235-2253.
- [33] Yuan Kuang, Xiuli Wu, Ziqi Chen, Wence Li, A two-stage cross-neighborhood search algorithm bridging different solution representation spaces for solving the hybrid flow shop scheduling problem, *Swarm and Evolutionary Computation*, 2024, 84, 101455.
- [34] Xiuli Wu, Yang Zhao, A balance-oriented iterative greedy algorithm for the distributed heterogeneous hybrid flow-shop scheduling problem with blocking constraints, *Swarm and Evolutionary Computation*, 2025, 96, 102015.
- [35] Yi-Jian Wang, Gai-Ge Wang, Ling Wang, A tree-based multiobjective evolutionary algorithm for energy-efficient hybrid flow-shop scheduling, *IEEE Transactions on Evolutionary Computation*, 2024.
- [36] Yang Zuo, Fuqing Zhao, Jianlin Zhang, A bi-population cooperative scatter search algorithm for distributed hybrid flow shop scheduling with machine breakdown, *Computers & Industrial Engineering*, 2024, 197, 110624.
- [37] DeepSeek-AI, DeepSeek-V4: Towards Highly Efficient Million-Token Context Intelligence, Technical report, 2026, https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro/blob/main/DeepSeek_V4.pdf.
- [38] Godfrey Onwubolu, Donald Davendra, Scheduling flow shops using differential evolution algorithm, *European Journal of Operational Research*, 2006, 171(2), 674-692.

- [39] OpenAI, GPT-4 Technical Report, arXiv preprint arXiv:2303.08774, 2023.
- [40] OpenAI, GPT-5 System Card, Technical report, 2025, <https://openai.com/index/gpt-5-system-card/>.
- [41] Xiuli Wu, Ziqi Chen, A hybrid genetic and variable neighborhood search algorithm for the hybrid flow-shop scheduling problem in pyrotechnic manufacturing industry with batch processing machines and lot-streaming, *Swarm and Evolutionary Computation*, 2026, 100, 102234.
- [42] Fuqing Zhao, Yuqing Du, Changxue Zhuang, Ling Wang, Yang Yu, An iterative greedy algorithm for solving a multiobjective distributed assembly flexible job shop scheduling problem with fuzzy processing time, *IEEE Transactions on Cybernetics*, 2025.
- [43] Yuan Kuang, Jing Sun, Xingjia Gan, Dunwei Gong, Zhiping Liu, Mingming Zha, Dynamic multi-objective cooperative coevolutionary scheduling for mobile underwater wireless sensor networks, *Computers & Industrial Engineering*, 2021, 156, 107229.
- [44] Ching-Jong Liao, Evi Tjandradjaja, Tsui-Ping Chung, An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem, *Applied Soft Computing*, 2012, 12(6), 1755–1764.
- [45] Jacques Carlier, Emmanuel Neron, An exact method for solving the multi-processor flow-shop, *RAIRO-Operations Research-Recherche Opérationnelle*, 2000, 34(1), 1–25.

Author biography



Fuqing Zhao received the B.Sc. and Ph.D. degrees from the Lanzhou University of Technology, Lanzhou, China, in 1994 and 2006, respectively. Since 1998, he has been with the School of Computer Science Department, Lanzhou University of Technology, Lanzhou, China, where he became a Full Professor in

2012. He has been as the post Doctor with the State Key Laboratory of Manufacturing System Engineering, Xi'an Jiaotong University, Xi'an, China in 2009. He has been as a visiting scholar in Exeter Manufacturing Enterprise Center in Exeter University and Georgia Tech Manufacturing Institute in Georgia Institute of Technology from 2018-2019 and 2014-2015 respectively. He has authored two academic book and over 90 refereed papers. His current research interests include intelligent optimization and scheduling.



Cheng Zhao received the B.S. degree from Lanzhou University of Technology, Lanzhou, China, in 2024. He is currently pursuing the M.S. degree in computer application technology, Lanzhou University of Technology, Lanzhou, China.

His current research interests include evolutionary computation and dynamic scheduling theory.



Tianpeng Xu received his Ph.D. degree in Manufacturing Information Systems from Lanzhou University of Technology, Lanzhou, China, in 2025. He is currently an Associate Researcher at the School of Computer Science and Artificial Intelligence, Lanzhou University of Technology. His research inter-

ests include complex system modeling and intelligent scheduling optimization.



Ningning Zhu was awarded her Doctoral degree by Lanzhou University of Technology, Lanzhou, China in 2024. She currently works as an Associate Professor at the School of Computer and Artificial Intelligence of the university, and is also an on-the-job postdoctoral researcher under the university-

enterprise joint training program. She has published a number of high-level academic papers. Her research interests include intelligent optimization and production scheduling, as well as machine learning-enabled efficient scheduling.



Houbin Song graduated from Harbin University of Science and Technology, Harbin, China, and gained a Ph.D in electrical engineering from the same university in 2014.

He has been with the College of Electrical and Information Engineering, Lanzhou University of Technology, Lanzhou, China, since 1998. His research interests include problems related to flow-bulb hydrogenerators and the deep integration of power machinery and information technology.