npj | advanced manufacturing

Article

https://doi.org/10.1038/s44334-025-00061-w

Leveraging large language models for efficient scheduling in Human–Robot collaborative flexible manufacturing systems

Check for updates

Jin Huang[1,2,4], Yue Teng[1,2,4], Qihao Liu[1,2], Liang Gao[1,2], Xinyu Li[1,2] ✉, Chunjiang Zhang[1,2] & Guoqing Xu[2,3]

With the increasing demand for customized manufacturing, human-robot collaborative (HRC) systems combine human adaptability with robotic precision, offering a promising solution for flexible production. Unfortunately, real-time scheduling remains a significant challenge due to high demand variability, frequent disruptions, and complex task allocation. To address these issues, we propose an evolutionary scheduling framework utilizing a local large language model (LLM). This framework enhances domain-specific understanding by supervising the fine-tuning of the LLM on scheduling data. Additionally, we introduce a population self-evolution mechanism that incorporates individual co-evolution, self-evolution, and collective evolution to improve the generation of heuristic dispatching rules (HDRs). By leveraging the local LLM, our approach generates dynamic HDRs with lower computational overhead, facilitating effective task allocation and sequencing in HRC scenarios while ensuring data privacy. Validated across 54 real-world HRC scenarios, our method achieves a 21.52% average makespan reduction, compared to baseline methods, demonstrating its potential for flexible manufacturing systems.

Industry 5.0 aims to establish a manufacturing framework that prioritizes human-centric and efficient processes, fostering seamless collaboration between humans and robots[1]. In this paradigm, human-robot collaboration (HRC) becomes a cornerstone of modern innovative manufacturing systems[2–5]. An ideal HRC scenario involves delegating repetitive, low-skill, and ergonomically challenging tasks to robots, thereby alleviating the physical strain on humans. At the same time, it emphasizes the importance of human intelligence and robotic dexterity in both operational and cognitive functions[6]. In response to increasingly diversified demands, manufacturing is transitioning from mass production to customized assembly[7,8]. As a result, robots with attributes such as speed, strength, repeatability, and precision are being integrated into manufacturing systems[9]. In this context, humans, programmable robots, and computer numerical control manufacturing systems operate in distinct physical spaces, each performing specific tasks and leveraging their unique advantages[10].

However, the resilience of a manufacturing system extends beyond robotic operations and must also be adaptable to fluctuations in orders and disruptions[11]. Traditional mass production lacks the flexibility required to offer small-batch personalized products in response to changes or interruptions[12]. Flexible manufacturing systems (FMS), based on HRC, show great promise in addressing this limitation. In such systems, humans handle material supply and maintenance, while robots manage the transportation of tools and materials to processing machines, facilitating small-batch customized manufacturing[13]. This approach enables multi-variety mixed-line production and reduces the need for large inventories of materials and finished products, thereby improving resource utilization, as shown in Fig. 1.

Existing FMS for HRC often face challenges, such as complex production processes, frequent faults, and fluctuating order demands, leading to complicated management and limited real-time responsiveness[14]. To address these issues, current FMS typically rely on simple rule-based resource scheduling[15,16], which works well in specific scenarios but results in significantly lower equipment utilization rates in others. Moreover, to ensure timely delivery, substantial quantities of production materials are preemptively prepared, leading to resource wastage and hindering the widespread adoption of flexible manufacturing models[17].

[1]State Key Laboratory of Intelligent Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan, Hubei, China. [2]National Intelligent Design and CNC Technology Innovation Center, Huazhong University of Science and Technology, Wuhan, Hubei, China. [3]Guangzhou MINO Equipment Co., Ltd, Guangzhou, Guangdong, China. [4]These authors contributed equally: Jin Huang, Yue Teng. ✉e-mail: lixinyu@mail.hust.edu.cn

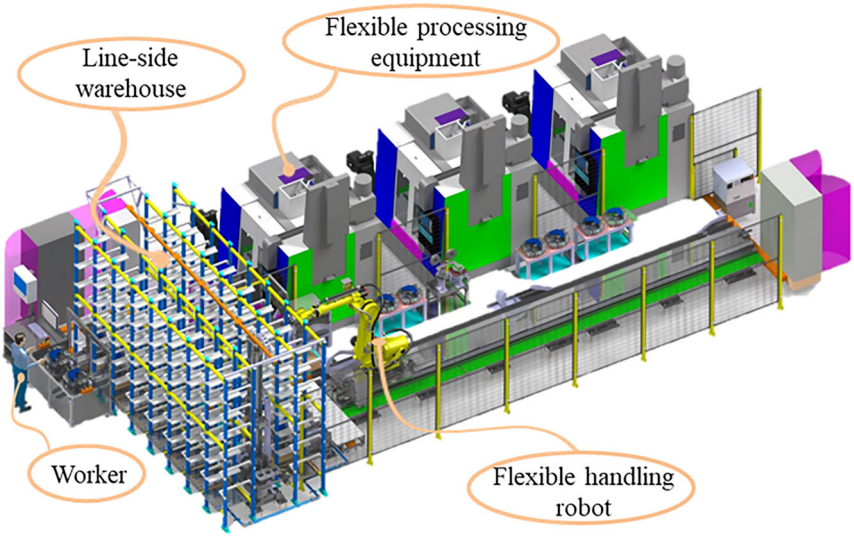


Fig. 1 | HRC flexible manufacturing system. The manufacturing system comprises a worker operator responsible for loading and unloading materials, with items stored in a line-side warehouse. A flexible handling robot transports workpieces, materials, and tools to flexible processing equipment for machining. After processing, the completed workpieces and resources are returned to the line-side warehouse. The system includes three flexible computer numerical control machines, each dedicated to machining standard deburring tools for automotive welding lines. These machines process a total of 12 types of components made from materials such as cast iron, aluminum alloy, and copper. The manufacturing processes encompass rough and finish machining, cleaning, inspection, and marking.

Traditional decision-making methods, which rely on workshop managers, are often inefficient and lead to significant resource waste. Workshop scheduling plays a critical role as the "brain" of the entire FMS, responsible for the optimal allocation of resources and task sequencing to meet production goals. A well-designed scheduling plan can enhance overall workshop efficiency without increasing resource input[18,19]. Solutions to workshop scheduling problems are generally divided into exact[18,20–22] and approximate[23–26] methods. Exact methods, such as branch-and-bound and dynamic programming, can achieve theoretically optimal solutions[27]. However, they are computationally intensive and time-consuming, making them unsuitable for the real-time demands of FMS. Consequently, approximate methods, including heuristic dispatching rules (HDRs)[15], metaheuristic algorithms[28–30], deep reinforcement learning (DRL)[24,31,32], genetic programming (GP)[25,33], and genetic expression programming (GEP)[34], have become increasingly widely used. However, metaheuristic algorithms may struggle to adapt quickly to dynamic changes due to the dynamic nature and large scale of FMS. HDRs and DRL methods are favored for their lower computational complexity and faster response times, but they may not consistently produce satisfactory results[35]. Although GP and GEP offer strong generalization capabilities, they are essentially random search processes that may not yield high-quality dynamic scheduling solutions within short time frames[36].

In recent years, with the increasing application of large language models (LLMs) across various domains[37–43], manufacturing systems have increasingly relied on AI to enhance quality, productivity, and overall performance[44]. The integration of LLMs with evolutionary algorithms has opened new opportunities for prompt engineering[45] and automated algorithm design[36,41,46]. Notably, the integration of LLMs with evolutionary algorithms has introduced new avenues for prompt engineering and automated algorithm design. Building on recent pioneering work, such as the early 2024 publication in *Nature* by the Google team, where LLMs combined with evolutionary algorithms achieved a new benchmark in combinatorial optimization[41], this research explores the transformative potential of LLMs within the domain of FMS. This approach benefits from the ability of LLMs to generate highly adaptable HDRs through meticulously designed prompts and iterative feedback mechanisms. By leveraging their language understanding and generation capabilities, LLMs can rapidly acquire domain-specific knowledge from training datasets and generate high-quality solutions in testing scenarios in a remarkably short timeframe. For example, models like ChatGPT and ChatGLM have been successfully used to evolve HDRs, addressing complex dynamic job shop scheduling challenges[36]. However, applying this to flexible manufacturing scheduling systems in HRC requires tackling specific challenges, such as machine selection within FMS, which complicates the direct application of such evolutionary frameworks in these contexts. Moreover, sharing production data with online LLMs often conflicts with data security requirements in manufacturing enterprises, making local deployment an essential consideration. While smaller models may lack sufficient inference capability, larger models demand high-performance hardware, resulting in prohibitively high costs. To overcome these challenges, this paper proposes an evolutionary framework for flexible manufacturing scheduling based on LLMs. The framework enhances HDR design through supervised fine-tuning (SFT) of the local qwen2.5-coder-7b model, built upon the population self-evolution (SeEvo) approach. This method enables the training of multiple cases, gathering the final HDRs, and ensures a high level of adaptability and efficiency in deployment. In the deployment phase, contextual prompts enable the rapid generation of high-quality scheduling plans for HRC manufacturing systems, ensuring a quick response to dynamic production environments. By combining the latest advances in LLMs with evolutionary techniques, this framework opens new opportunities for addressing the complexities of real-time, dynamic manufacturing scheduling, significantly advancing the field and pushing the boundaries of AI-driven optimization.

Results

Dynamic flexible manufacturing system scheduling performance testing

In recent years, the application of machine learning in FMS scheduling has experienced a surge, resulting in the development of rich datasets and benchmark tests. However, DRL, which is widely used by researchers to address dynamic workshop scheduling problems, does not consistently outperform traditional HDRs[47,48]. Similarly, while evolutionary frameworks such as GP[25] and GEP[34] have been employed for automatic algorithm design, these methods suffer from ineffective guided exploration. Their reliance on extensive random search limits their development and exploration capabilities[36].

To address this issue, we develop SeEvo, a language-guided heuristic framework designed to efficiently generate scheduling solutions for dynamic FMS environments by leveraging the capabilities of LLMs. Using a local qwen2.5-coder-7b model, we generate evolutionary prompts that guide the evolution of initial seed heuristic algorithms while continuously collecting effective datasets. The evolutionary process is inspired by individual co-evolution, individual self-evolution, and collective evolution[36]. Additionally, unlike online LLMs such as ChatGPT, which are used in the literature, we further enhance our local qwen2.5-coder-7b through SFT to improve HDR evolution efficiency and address data privacy concerns.

Notably, the effectiveness of SeEvo's outputs depends on the reasoning capabilities of the LLM and the quality of HDRs accumulated across multiple cases. For instance, directly using open-source LLMs may result in limited reasoning capabilities, making it challenging to generate effective HDRs based on current cases and prompts quickly. This is especially true when evolutionary reflection prompts produce flawed or nonsensical inputs, causing both local LLMs and online LLMs like ChatGLM4 to fail to generate better HDRs. Similarly, inaccurate LLM outputs can lead to erroneous scheduling plans. For these reasons, we evaluate SeEvo's performance from three dimensions: (1) the accuracy of the generated HDRs, (2) reasoning ability, and (3) the quality of rapid inference on the test set.

As shown in Fig. 2a, qwen2.5-sft-7b achieves HDR generation accuracy close to gpt3.5. Although not perfectly accurate, this represents a significant improvement over the pre-fine-tuned qwen2.5-coder-7b. In Fig. 2b, a comparative experiment of the SeEvo framework is presented across 50 generations and 10 test cases, benchmarked against traditional methods such as GP, GEP, as well as online LLMs like gpt3.5 (gpt-3.5-turbo-ca), glm3 (GLM-3-Turbo), glm4 (GLM-4), the pre-fine-tuned qwen2.5-coder-7b, and qwen2.5-sft-7b. The results demonstrate that qwen2.5-sft-7b outperforms both traditional automatic algorithm design methods and online LLMs, including the pre-fine-tuned qwen2.5-coder-7b. Additionally, to evaluate the generalization and robustness of the framework, we conduct a benchmark comparison on a test set of 200 cases against DRL, GP, GEP, and 10 HDRs. Performance is measured by the relative deviation of each method's solution from the current best solution. The boxplot results (Fig. 2c) indicate that LLM-guided scheduling methods consistently outperform traditional approaches. Notably, qwen2.5-sft-7b demonstrates exceptional performance, with a median relative deviation close to zero and a Gap ratio below 1% for the majority of cases. This indicates both high stability and superiority, significantly surpassing other non-LLM-guided methods, and suggests its potential as a valuable assistant for HRC manufacturing.

Additionally, to validate the effectiveness of the novel individual self-evolution mechanism in SeEvo, we perform an ablation study. By comparing the complete SeEvo strategy with a simplified version (denoted ReEvo, which lacks the individual self-evolution mechanism), we find that under the same LLM conditions, the SeEvo strategy outperforms the ReEvo, as demonstrated by a smaller Gap ratio between its solutions and the best makespan (Fig. 2d). The combination of qwen2.5-sft-7b and the SeEvo strategy yields the best performance, with the most concentrated distribution of Gap ratios. In contrast, for any given LLM, SeEvo consistently outperforms ReEvo. Among the three API-based models, glm3 shows slightly better results. It is worth noting that while glm4 demonstrates strong exploration performance in Fig. 2b, its performance significantly decreases in Fig. 2c–e. This is mainly attributable to our experimental design, where each LLM performs a rapid iteration on the test cases based on its knowledge base of 20 HDRs, making the outcome highly dependent on the quality of that specific knowledge base. We do not explore multiple knowledge bases further, as the primary focus of this paper is the design of the SeEvo method and the local fine-tuning pipeline. Online LLMs are included only to validate the effectiveness of our method and framework, as their adoption in manufacturing is often limited by data privacy concerns. Finally, we further compare the number of best solutions and the average makespan by different LLM methods across the 200 test cases. The results show that qwen2.5-sft-7b (SeEvo) not only finds the highest number of best solutions but also achieves the lowest average makespan, once again confirming its superior overall performance.

Machining flexible manufacturing system scheduling performance testing

To validate the effectiveness of the SeEvo method in a real-world FMS, tests are conducted using operational data from the flexible production line at the headquarters of Guangzhou MINO Equipment Co., Ltd. This production line consists of seven vertical computer numerical control centers capable of mixed-line manufacturing. The system processes a variety of mechanical products, including angle seats, bases, sliding plates, connecting blocks, three-axis bases, roller bases, tray bases, and manual lubrication tables, across multiple orders. The line also features a stacker crane, a palletizer, three flexible handling robots, and two workers (Fig. 3a). In this study, the workshop processes four types of products, each with multiple orders of varying quantities. Orders arrive dynamically based on random user demand and are entered by an operator, who supplies the corresponding materials to a line-side buffer. Machine faults, captured by sensors, occur unpredictably. The optimization objective is to minimize the makespan. The architecture of the FMS is depicted in Fig. 3b. At its core is an intelligent management and control system driven by the scheduling algorithm, which enables centralized resource management, intelligent allocation, efficient process scheduling, and adaptive production in response to dynamic disturbances.

In 54 test cases simulating a real-world production scenario, the SeEvo framework is benchmarked against several baseline methods. The results demonstrate that the SeEvo framework, integrated with the fine-tuned qwen2.5-sft-7b model, outperforms the others. It achieves the lowest median Gap Ratio with the most concentrated distribution, indicating superior solution quality and stability. Moreover, for all tested LLMs, the SeEvo method consistently outperforms its simplified version, ReEvo (Fig. 4a).

A comprehensive performance evaluation further confirms this advantage. On key metrics, such as the number of best solutions obtained and the average minimum makespan, the qwen2.5-sft-7b configuration consistently performs better than the others. In contrast, the HDR originally used by the production line (denoted "Before") achieves the best solution only once in a scenario (3×13) and performs poorly in the majority of scenarios, indicating its limited generalization capability (Fig. 4b). A heatmap analysis visually represents the performance of different methods across various production scales. The qwen2.5-sft-7b (SeEvo) configuration maintains the smallest relative Gap (indicated by dark green) in almost all scale combinations, demonstrating its robust performance (Fig. 4c). Most importantly, all LLM-generated methods consistently and significantly outperform the original HDR across all test scenarios, emphasizing the practical value of the framework (Fig. 4d).

Beyond its quantitative performance advantages, the HDR generated by qwen2.5-sft-7b (Fig. 4e) is structurally clear and logical, effectively integrating sub-policies with physical meanings. For example, in workpiece selection, it combines the principle of selecting the job with the highest completion percentage while using a forward-looking term for fine-tuning, thus balancing the current state with future trends. For machine selection, it aims to assign a workpiece to a machine with the most balanced load and stability, taking into account the processing complexity of the workpiece itself, potentially prioritizing a "long-duration task" for a machine with greater idle capacity.

In contrast, the HDRs generated by traditional automated design methods, such as GEP and GP (Fig. 4f, g), while effective, are mathematically complex and verbose, with limited interpretability. This black-box nature complicates understanding and debugging in real-world production environments. This comparison highlights the unique advantage of the SeEvo framework: it not only discovers high-performance scheduling strategies but also ensures these strategies are human-understandable, facilitating their application in other complex FMS scenarios, such as aerospace skin manufacturing.

Discussion

This paper presents an LLM-based SeEvo framework for FMS production scheduling, which integrates three stages: individual co-evolution, individual self-evolution, and collective evolution. During the application and testing phases, we input the prompts and 20 pre-collected HDRs into the fine-tuned LLM for inference using the SeEvo framework, with the results directly applied to the FMS. The results significantly outperform existing HDR-based scheduling approaches. Additionally, the framework demonstrates the ability to generate high-performance HDRs within just one

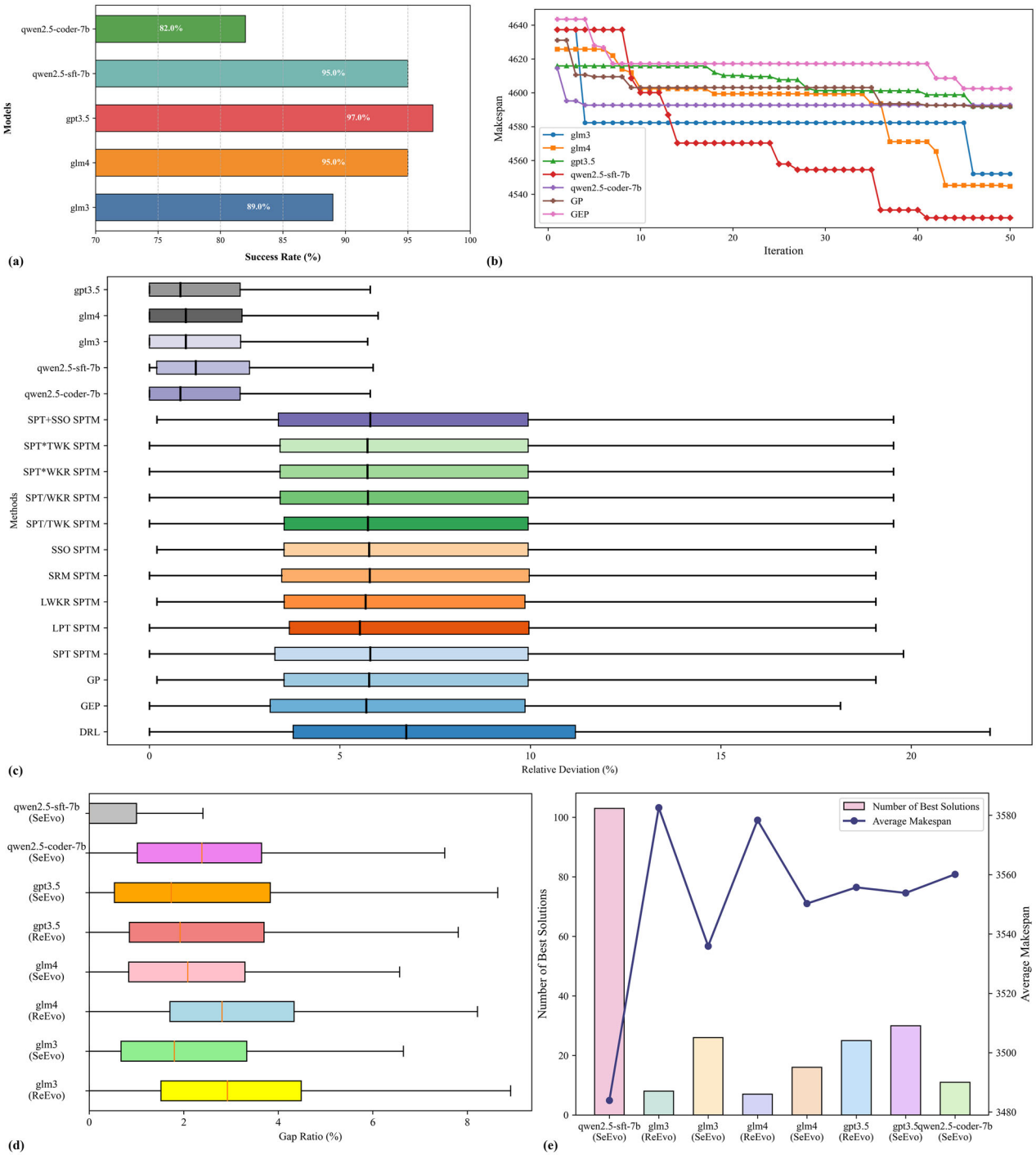


Fig. 2 | Performance evaluation of the scheduling method for dynamic FMS based on LLMs. a Comparison of the success rates of different LLMs in generating HDRs. The results show that gpt3.5 achieves the highest success rate, followed by glm-4 and qwen7b-sft. **b** Convergence performance curves for various methods in 10 random cases. The qwen2.5-sft-7b method demonstrates stronger convergence, indicating its powerful search and exploration capabilities. **c** Box plot of the relative ratio for different scheduling methods across 200 test cases. The performance of qwen7b-sft is significantly superior to all other methods. **d** Box plot of the Gap ratio for different LLM methods and their associated evolutionary strategies (SeEvo vs. ReEvo). The combination of qwen2.5-sft-7b and the SeEvo strategy yields the best performance. **e** Comparison of the number of best solutions obtained (bar chart) and the average makespan (line chart) by each LLM method across 200 test cases. Qwen2.5-sft-7b (SeEvo) performs best on both key metrics, achieving the highest number of best solutions and the lowest average makespan.

minute, offering a novel solution for the application of LLMs in intelligent manufacturing. Notably, the fine-tuned LLM requires only a single 4090D GPU to complete the inference experiments, significantly reducing the cost of utilizing online LLM APIs. When two GPUs are used, the inference speed is comparable to proprietary closed-source models, such as ChatGLM3. These findings underscore the considerable potential of the SeEvo method in FMS scheduling and confirm its effectiveness as a tool for generating scheduling plans.

Despite its numerous advantages, the SeEvo method faces several challenges during the inference process. The selection of training cases, as well as the inherent limitations of the evolutionary logic underlying large-scale test solutions, complicates the evaluation of the SeEvo

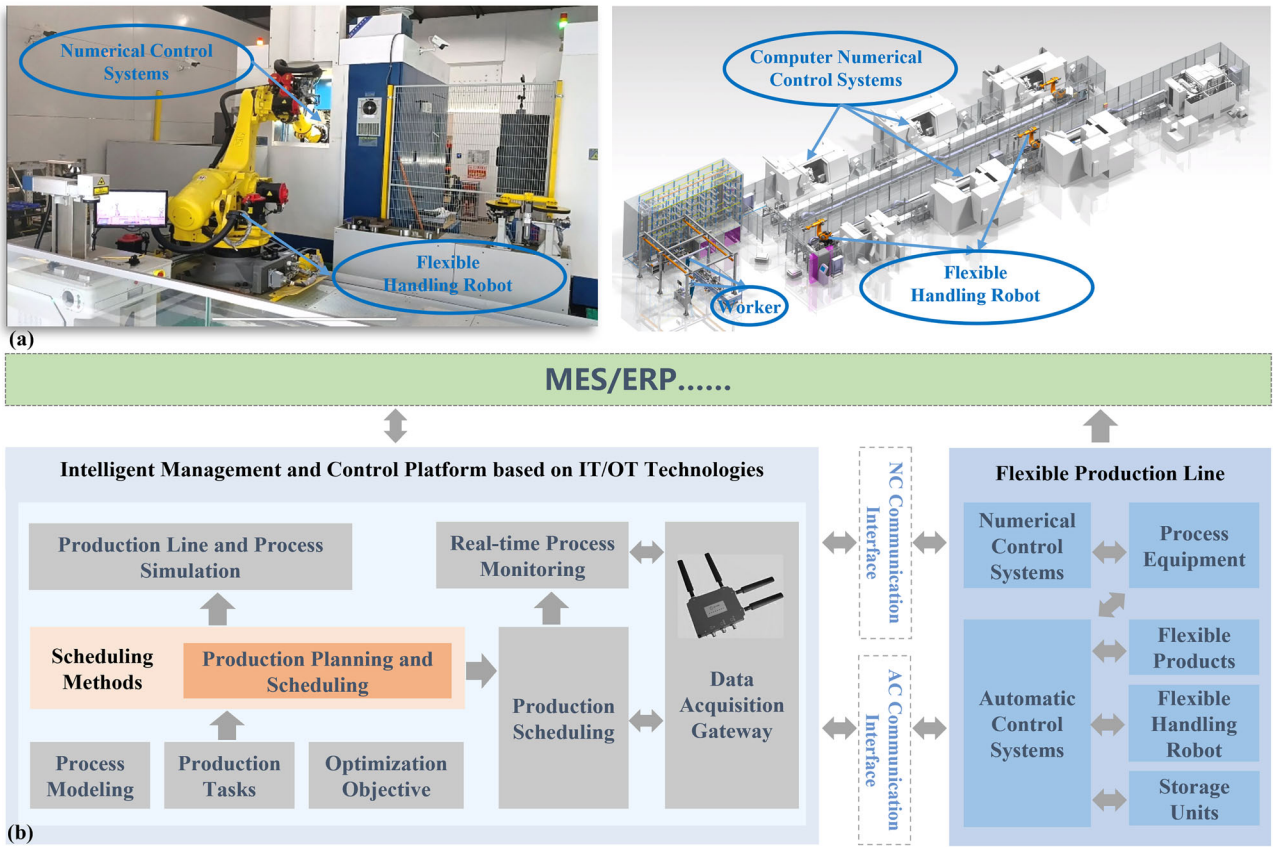


Fig. 3 | Architecture of the FMS. a Schematic of the physical entity (left) and the 3D virtual environment (right) of the FMS line, including the core components of the production line: computer numerical control systems, flexible handling robots, and workers. In a typical process, the worker secures a workpiece onto a pallet, which is then transported by the robot to a computer numerical control system for machining. **b** The cyber-physical integration architecture of the system. An intelligent management and control platform is established under the manufacturing execution system, with the production planning and scheduling module at its core. This module generates scheduling plans by integrating inputs such as process modeling, production tasks, and optimization objectives. Through a data acquisition gateway and communication interfaces, it dispatches commands to the physical production line and collects real-time data, thus creating a closed-loop control system.

method. Furthermore, the model's dependence on a predefined set of HDRs may limit its scalability and adaptability in more realistic scenarios. Future research should focus on constructing larger-scale knowledge bases and developing efficient retrieval mechanisms through a knowledge augmented generation (KAG)[49] framework. Additionally, it will be crucial to explore advanced fine-tuning techniques, such as group relative policy optimization (GRPO), which could further enhance the model's performance, particularly in domains that require more complex and adaptive capabilities.

Method

Evolutionary mechanism based on LLMs

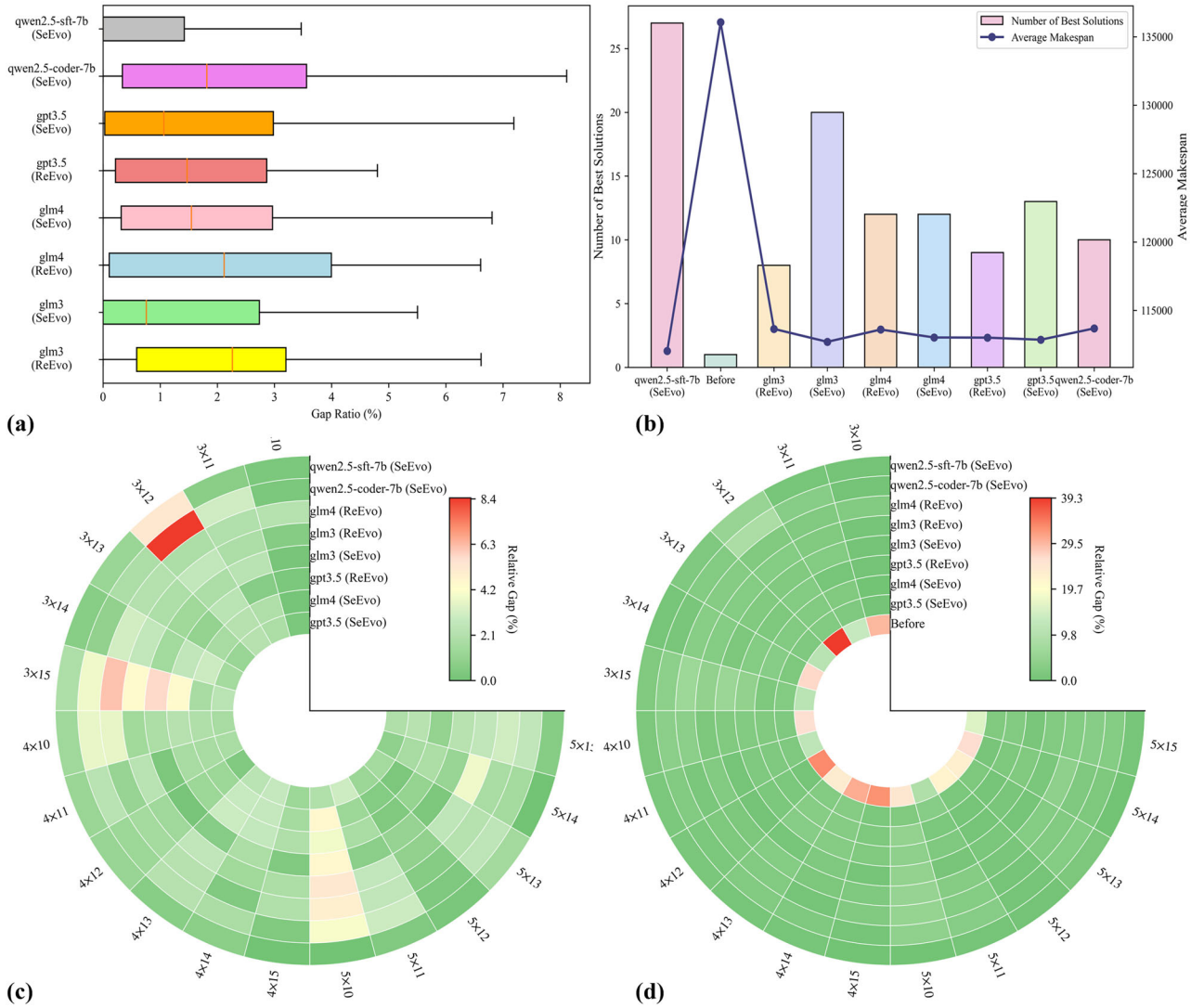
Within the SeEvo framework, LLMs perform two key roles: the Reflector LLM, which generates guiding prompts for individuals, and the Generator LLM, which produces individual HDRs. Unlike traditional hyper-heuristic algorithms such as GP and GEP, which rely on fixed encoding structures and function sets, each individual in SeEvo is a code block generated directly by the LLM. These individuals are only required to adhere to a predefined function signature, including the function name, inputs, and outputs, thus overcoming the limitations associated with fixed encoding length and complexity.

The overall evolutionary process of SeEvo (Fig. 5) is implemented through an iterative loop consisting of three core stages: individual co-evolution, individual self-evolution, and collective evolution. The specific implementation details are as follows:

Population initialization. The LLM generates an initial population of HDRs based on the task specifications and a seed HDR. The prompt engineering process used for this initialization is depicted in Fig. 6.

Individual co-evolution and crossover. Two parent HDRs are randomly selected from the current population for performance comparison. The evaluation result (e.g., superior or inferior performance on test cases) is fed back to the Reflector LLM. The system guides the LLM to analyze the performance differences in depth and to generate instructive recommendations for improvement. This comparative mechanism provides feedback akin to a language gradient, even in the absence of a continuous reward signal. The analysis and recommendations serve as evolutionary instructions, guiding the Generator LLM to produce two new offspring HDRs based on this parent pair.

Individual self-evolution and crossover. In this stage (Fig. 5b), the system feeds the performance trajectory of each individual before and after co-evolution back to the Reflector LLM. The LLM is prompted to reflect on the changes in performance: if performance has declined or stagnated, the LLM analyzes potential causes and generates reverse prompts to prevent further failures. If performance has improved, the LLM synthesizes successful experiences to generate optimization prompts that amplify strengths. These targeted recommendations guide the individual's self-optimization, resulting in the generation of a new offspring. The crossover operation in this stage mirrors that in the co-evolution stage.




```
job_select_expression = rm / (rm + 0.5 * so)

stability_factor = 1.5 - abs(0.5 - ur)
machine_select_expression =  (ur * stability_factor) * (ptm / np.mean(ptm[ptm != 0]))
```

(e)

```
job_select_expression = (((wkr + wkr) * so) - (wkr + wkr))

machine_select_expression = (ptm - ((ur + ur) + ur))
```

(f)

```
job_select_expression = (sin((-(pt + wkr))) + ((((((-sin(sin(sin((rm / pt))))) + (((cos((wkr - twk)) * ((twk / so) + cos(pt))) * cos(((so - twk) - (rm * wkr)))) * sin(cos(((rm - 1) - (--1)))))) * (((((-((twk + rm) + (pt - pt))) + (cos((rm * so)) + sin((wkr - twk)))) - (-(((pt * -1) / sin(pt)) * ((so * ((so * pt) + sin(rm))) * (rm / wkr))))) / (((cos(sin(wkr)) * sin((pt / rm))) * (((wkr * so) + (twk * twk)) / (-(pt * rm)))) / ((cos((1 - rm)) / ((pt * pt) / (rm * so))) + sin((sin(rm) * cos(wkr)))))))) / rm) / sin(rm)) * (-(-(so - wkr)))))

machine_select_expression = ptm + ptm
```

(g)

Fig. 4 | Performance evaluation and analysis of generated HDRs in real-world manufacturing scenarios. a Box plot comparing the optimization performance of different LLM methods across 54 scheduling scenarios. Qwen2.5-sft-7b (SeEvo) exhibits the best performance, with the smallest median and distribution range for the Gap Ratio. **b** Performance of each method across the same 54 real-world scenarios on best-known solutions and average makespan. **c** Heatmap of the Relative Gap for different LLM methods across various problem scales (from 3 × 10 to 5 × 15). The heatmap shows that qwen2.5-sft-7b (SeEvo) consistently maintains the lowest relative Gap across most tested scales. **d** Heatmap comparing the performance of all LLM-based methods against the HDR originally used by the factory (Before). The results indicate that all LLM-generated methods significantly outperform the original HDR. Comparison of HDRs generated by different methods. The HDR (**e**) generated by qwen2.5-sft-7b (SeEvo) is well-structured and highly interpretable. In contrast, the HDRs generated by GEP (**f**) and GP (**g**), while effective, are mathematically complex, lack intuitive physical meaning, and exhibit poor interpretability.

Collective evolution and mutation. This stage (Fig. 5c) provides macro-level control over the population's evolutionary trajectory. Long-term reflection data from previous iterations, co-evolutionary reflections from the current round, and self-evolutionary reflections are integrated. The Reflector LLM synthesizes this global information to generate insights into the overall evolutionary direction. This high-level guidance directs the mutation of the best-performing parent individual in the current population, encouraging a more thorough exploration of the current optimal solution. The number of HDRs generated depends on the mutation probability.

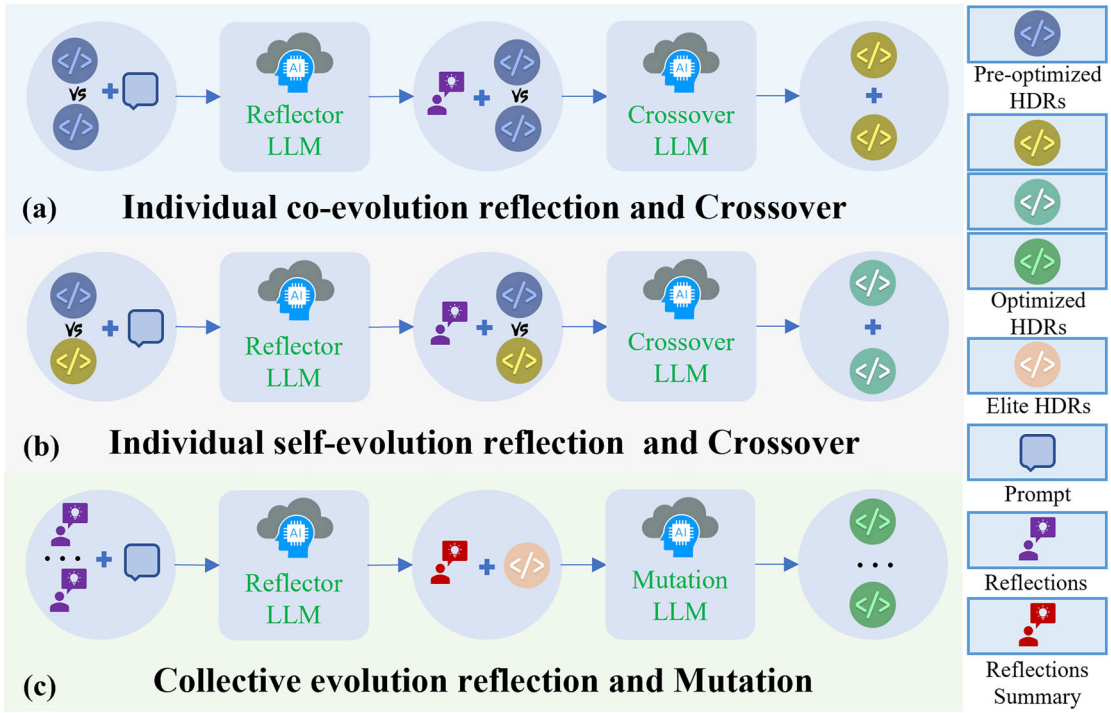


Fig. 5 | The SeEvo framework: an evolutionary process for heuristic algorithms driven by LLMs. a Individual co-evolution reflection and crossover: The system randomly selects two parent individuals for performance comparison. This comparison then serves as input for the Reflector LLM to generate an in-depth analysis of their respective strengths and weaknesses. The resulting analysis acts as an evolutionary instruction, guiding the two parents to generate new offspring. **b** Individual self-evolution reflection and crossover: The system provides feedback to the Reflector LLM on the performance trajectory of each individual before and after an evolution step. Based on this, the LLM generates targeted suggestions for improvement, guiding the individual to self-optimize and produce a new offspring. **c** Collective evolution reflection and mutation: This stage integrates all co-evolutionary and self-evolutionary knowledge accumulated in stages (**a**) and (**b**). This global information is submitted to the Reflector LLM to generate a macro-level insight into the overall evolutionary direction. This high-level guidance is specifically used to direct the mutation of the best parent individual in the current population, facilitating deeper exploration from the current best-known solution.

Model training, knowledge base construction, and fast inference

To enhance the LLM's performance on the specific scheduling problem and facilitate rapid deployment, a comprehensive pipeline is designed, encompassing data generation, cleaning, model fine-tuning, knowledge base construction, and fast inference (Fig. 7).

Data generation and cleaning. Initially, the SeEvo framework is executed using the base qwen2.5-coder-7b model on 200 randomly generated scheduling cases, with each case iterated for 50 rounds. During this process, complete interaction records from each evolutionary step are systematically collected, including all reflective prompts (inputs) and their corresponding generated HDRs (outputs), which form the raw dataset (Fig. 7a). Each HDR is evaluated in an isolated subprocess, with up to 20 subprocesses running in parallel. If any subprocess encounters an error or times out, it is terminated without affecting the main evolutionary process. The main process then evaluates the outcomes by reading results from designated text files.

The data is then cleaned by filtering for datasets that result in performance improvements. Specifically, when the performance of an offspring HDR exceeds that of its parent, the corresponding instruction-response pair is retained (Fig. 7b). Any failed executions result in very low fitness scores for those HDRs, ensuring that they are excluded during the selection phase. This process ensures that only high-quality, performance-enhancing instances are retained in the dataset, contributing to more reliable results in subsequent evaluations.

Supervised fine-tuning. Using the curated high-quality data, SFT is performed on the base model. The fine-tuning process is conducted with the Llama Factory[50] (https://github.com/hiyouga/LLaMA-Factory) on a single A800-80GB GPU. During SFT, successful evolutionary instructions (reflective prompts) are treated as the "instruction", and the improved individual HDRs as the "output". This process yields the qwen2.5-sft-7b model, which exhibits enhanced problem-solving capabilities.

High-quality HDRs knowledge base construction. To provide a high-quality initial population for the fast inference stage, another 50 rounds of deep evolution are performed using the fine-tuned qwen2.5-sft-7b on 20 representative training instances. The high-quality HDRs generated during these iterations are collected to form an elite knowledge base consisting of 20 HDRs (Fig. 7c). While the current knowledge base is limited, primarily due to the suboptimal performance of traditional vector-matching methods in this context, future work will focus on constructing larger-scale knowledge bases and enabling their efficient retrieval through a KAG[49] framework.

Fast HDR generation. During the online application or testing phase (Fig. 7d), the system invokes the fine-tuned qwen2.5-sft-7b model and uses HDRs from the knowledge base as the initial population. A single complete iteration of the SeEvo framework (i.e., sequential individual co-evolution, individual self-evolution, and collective evolution) generates a high-quality solution for a new scheduling problem in under one minute.

Intelligent HRC flexible manufacturing system

System architecture. The FMS (Fig. 3b) comprises three primary modules: the flexible production line, the management and control system, and the LLM-based scheduling module. The production line hardware includes multiple computer numerical control systems,

Individual Self-Evolution Reflection:

```
Below are two {func_name} functions for {problem_desc}
{func_desc}

You are provided with two code versions below. The second version is generated from the first version based on the following hint, and the result of the second version became more {result} compared to the first.

[First version code]
{first_version_code}

[Second version code]
{second_version_code}

[hint]
{hint}

You respond with some hints for designing better heuristics for first version code, based on the two code versions and the prior hints, using less than 50 words.
```

Individual Co-Evolution Reflection:

```
Below are two {func_name} functions for {problem_desc}
{func_desc}

You are provided with two code versions below, where the second version performs better than the first one.

[Worse code]
{worse_code}

[Better code]
{better_code}

You respond with some hints for designing better heuristics, based on the two code versions and using less than 50 words.
```

problem_desc:

```
Solving Flexible Job Shop Scheduling Problem (FJSSP) with constructive heuristics. FJSSP requires achieving the minimum makespan through effective production arrangement of machines, workpieces and operations.
```

user_generator:

```
Write a {func_name} function for {problem_desc}
{func_desc}
```

Crossover:

```
{user_generator}

[Worse code]
{func_signature0}
{worse_code}

[Better code]
{func_signature1}
{better_code}

[Reflection]
{reflection}

[Improved code]
Please write an improved function `{func_name}_v2`, according to the reflection. Output code only and enclose your code with Python code block: ```python ... ```..
```

Mutation:

```
{user_generator}

[Prior reflection]
{reflection}

[Code]
{func_signature1}
{elitist_code}

[Improved code]
Please write a mutated function `{func_name}_v2`, according to the reflection. Output code only and enclose your code with Python code block: ```python ... ```.
```

Collective Evolution Reflection:

```
Below is your prior long-term reflection on designing heuristics for {problem_desc}
{prior_reflection}

Below are some newly gained insights, with corresponding new reflection results either improving (better) or worsening (bad) compared to before.
{new_reflection}
{new_reflection_result}

Write constructive hints for designing better heuristics, based on prior reflections and new insights and using less than 50 words.
```

func_desc:

```
The composite expression for scheduling operations in a Flexible Job Shop Scheduling Problem (FJSSP) is determined by the following variables:

Operations Scheduling Variables (used when select_composite == True):
  pt: Processing time of the current operation for all workpieces.
  wkr: Remaining processing time for all workpieces.
  rm: Remaining processing time for all workpieces except the current operation.
  so: Processing time of the succeeding operation for all workpieces.
  twk: Total processing time for all workpieces.

Machine Selection Variables (used when select_composite == False):
  ptm: Processing time of the current operation on a specific machine.
  ur: Utilization rate of each machine.

Function Behavior:
- If `select_composite == True`, the composite expression will be based on the **operations scheduling variables** (`pt`, `wkr`, `rm`, `so`, `twk`).
- If `select_composite == False`, the composite expression will be based on the **machine selection variables** (`ptm`, `ur`).

The composite expression can include combinations of operations like `+`, `-`, `*`, and `/` to represent the relationships between these variables. The expression is designed to help make decisions in both workpiece scheduling and machine selection, depending on the value of `select_composite`.
Please avoid making the composite expression too complex. It should remain manageable and interpretable, focusing on the most relevant relationships between variables without introducing unnecessary complexity.
```

Seed_func:

```
import numpy as np
def get_combined_expression_v1(pt: np.ndarray = None, wkr: np.ndarray = None, rm: np.ndarray = None, so: np.ndarray = None, twk: np.ndarray = None, ptm: np.ndarray = None, ur: np.ndarray = None, select_composite: bool=True) -> np.ndarray:
    if select_composite == True:
        combined_expression_data = (((twk * wkr) * (wkr * so)) + (rm * (so - so)))
    else:
        combined_expression_data = ((((ur + ptm) + ptm) - (ptm + ptm)) + (ptm - ur))
    return combined_expression_data
```

Fig. 6 | Prompt engineering design. The design of the prompt engineering, including individual co-evolution reflection, individual self-evolution reflection, collective evolution reflection, crossover, and mutation.

handling robots, an automated warehouse, buffer positions, tools, and fixtures. The management system is responsible for data coordination, resource management, and task allocation. The LLM-based scheduling module acts as the core decision-making unit, receiving inputs such as the production line model, process flows, production orders, and optimization objectives, which are processed through SeEvo to generate scheduling solutions.

In our system, we have seven machines, and each workpiece consists of multiple operations. For many of these operations, multiple alternative machines are available for processing, and different types of workpieces have distinct technological routes. These characteristics align closely with the flexible job shop scheduling problem, where jobs (workpieces) require a sequence of operations, each of which can be performed on one of several alternative machines. This routing flexibility and the machine assignment decision-making process are central to the scheduling challenge in our system, making FJSP a natural fit for modeling our manufacturing environment.

The use of FJSP allows us to efficiently address the complexities of machine assignment, job sequencing, and processing uncertainties, all of which are critical in optimizing the performance of our intelligent HRC-FMS.

Scheduling processing. In this flexible manufacturing system, the production flow for a workpiece involves several steps, including manual loading, robotic handling, multi-operation machine processing, and manual fixture changes. This scenario is modeled as a flexible job-shop scheduling problem characterized by dynamic order arrivals, random machine faults, and fuzzy processing times. To capture time fluctuations in real-world production, a processing time ambiguity of 2 min is introduced for each operation in the dataset. The production cycle is computed using an event-driven simulation model, where each subsequent operation is only released into the pool of jobs awaiting processing after the completion of the previous operation (Fig. 8).

Online scheduling execution framework. The proposed framework is divided into two phases: the self-evolution phase and the online application phase (Fig. 8). During the self-evolution phase, SeEvo undergoes extensive evolution across multiple training cases to create a high-quality HDRs knowledge base. In the online application phase, when new production tasks or dynamic disturbances (e.g., urgent orders or machine faults) arise, the system leverages the HDRs from the knowledge base as an initial individual. A single rapid iteration is then performed to generate an optimized scheduling plan. The generated HDRs are primarily used for job selection. When multiple jobs compete for the same machine, the HDR calculates a selection probability for each candidate job, and the system assigns the job with the highest probability for processing. This mechanism enables the system to respond to dynamic events in seconds, maintaining ongoing operations while rescheduling subsequent tasks in real-time. This approach effectively mitigates issues such as the creation of semi-finished products or production disruptions caused by interrupting current operations.

Data generation and experimental design methodology

In the Intelligent HRC flexible manufacturing system, the data generation process is crucial for creating realistic test scenarios to validate and optimize the scheduling system. The experiments utilize two types of generated data:

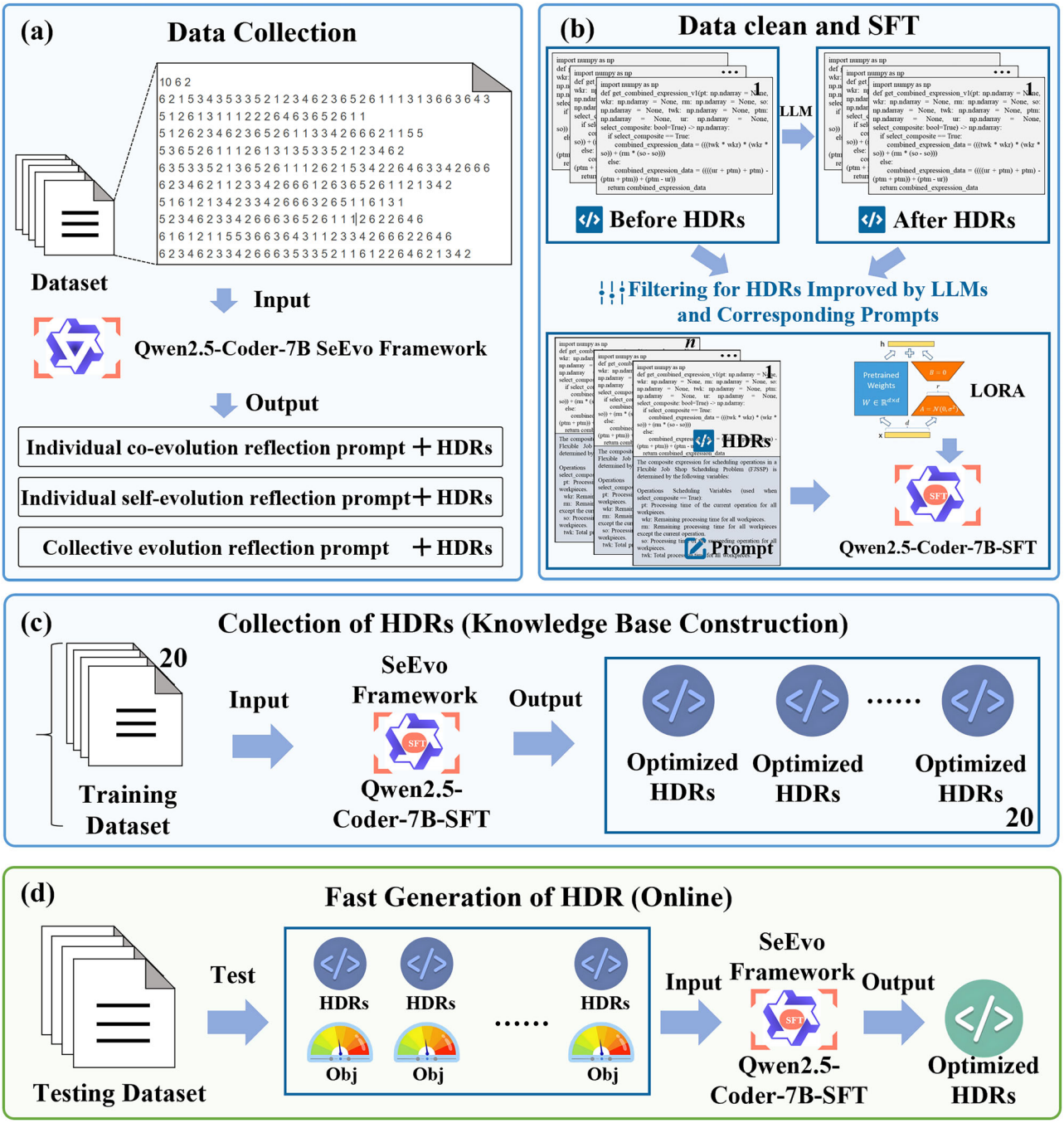


Fig. 7 | Data cleaning, fine-tuning, knowledge base construction, and fast inference framework. a evolutionary data generation: The qwen2.5-coder-7b model is used with the SeEvo framework to run 50 rounds of evolution for each of the 200 random cases. The complete interaction logs from this process, including all successful evolutionary instructions and HDRs. **b** data cleaning and SFT: The raw data is filtered to retain only the lead to performance improvement. Specifically, if an offspring's HDR outperforms its parent, the instruction-response pair that prompted this optimization is selected. These high-quality datasets are then used to perform LoRA fine-tuning on the qwen2.5-coder-7b. **c** HDRs knowledge base collection: The qwen2.5-sft-7b is employed to perform another 50 rounds of deep evolution on 20 training cases. The resulting evolved HDRs are collected to build a high-quality knowledge base, providing high-quality examples for the subsequent fast inference stage. **d** Fast HDR generation: the system calls the qwen2.5-sft-7b and utilizes the HDRs from the knowledge base as the initial individuals. With just a single iteration of the SeEvo framework, the system can generate a high-quality scheduling solution for a new problem within one minute.

randomly generated cases and simulation data based on actual processing information.

Randomly generated cases. The random generation process mimics real-world uncertainties and operational complexities in manufacturing. Here is the breakdown:

1. Order Quantity: The number of orders is randomly chosen between 3 and 10.
2. Workpiece Quantity per Order: Each order consists of a randomly generated number of workpieces, ranging from 10 to 20 pieces.
3. Machine Availability: The number of machines is randomly set between 5 and 10.

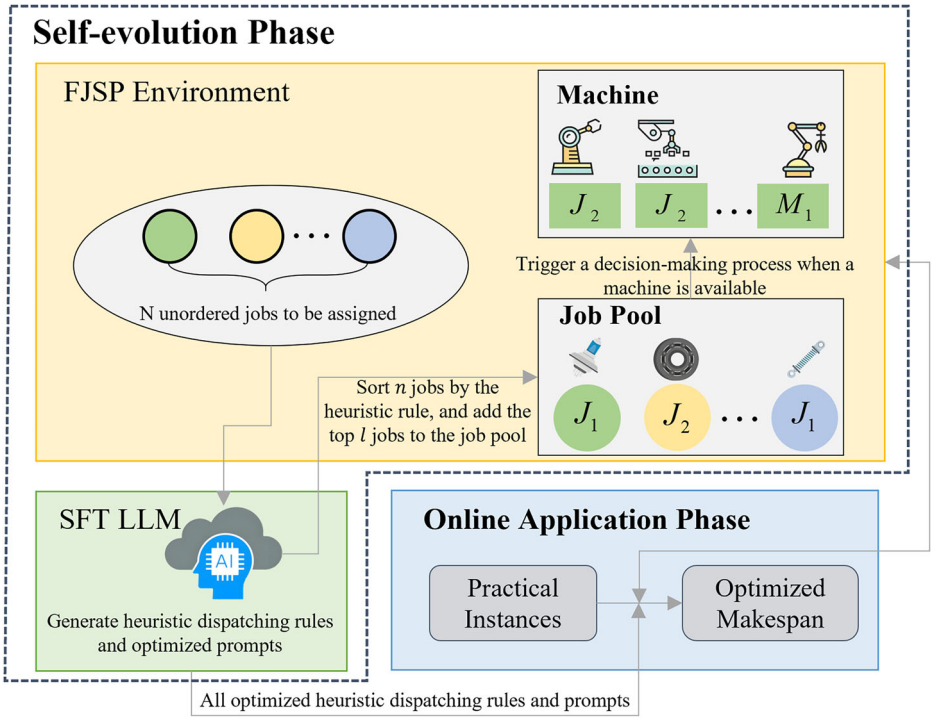


Fig. 8 | HRC flexible manufacturing scheduling evolution framework. The evolutionary framework consists of two phases: the self-evolution phase and the online application phase. The self-evolution phase is divided into two parts: the flexible job shop environment and the SFT LLM. In this phase, HDRs are evolved for 20 cases using the SeEvo framework. In the online application phase, the best HDR is derived through one run of the SeEvo, which is then applied to the HRC flexible manufacturing scheduling system to improve production efficiency.

4. Operations per Workpiece: Each workpiece undergoes a random number of operations, ranging from 5 to 7 steps.
5. Machine Failures: To simulate real-world disruptions, the number of machine failures is randomly generated between 0 and 3 occurrences.
6. Fuzzy Processing Time: Processing time is assigned with a slight variation (± 2 min) to introduce some uncertainty and model real-world fluctuations in time.
7. Machines per Operation: The number of machines available for each operation is randomly chosen between 0 and 3 machines.
8. Processing Time per Workpiece: Each operation has a random processing time between 20 and 60 min, represented as a random integer.

From this methodology, 200 distinct random cases are generated to cover various possible scenarios that the system might encounter in real-world conditions.

Simulation data based on actual processing information. This data type is derived from actual shop floor processing information, ensuring high fidelity to real-world production environments. It helps validate the system by matching the simulation closely with actual production data.

1. Number of Orders: 3 to 5 orders are simulated.
2. Workpiece Quantity per Order: Each order contains 10 to 15 workpieces.
3. Machine Failures: Random machine failures are introduced, mirroring the unpredictability of real manufacturing systems.
4. Processing Information: The specific processing information for each workpiece is derived directly from real-world data collected in a machine shop, ensuring that the simulation accurately reflects actual operational realities.

By combining both randomly generated and real-world simulation data, the system can better respond to dynamic, unpredictable manufacturing conditions, ensuring that the scheduling solutions are not only theoretically sound but also practical and reliable in real-world settings.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Code availability

The underlying code for this study is not publicly available but may be made available to qualified researchers on reasonable request from the corresponding author.

Received: 13 May 2025; Accepted: 10 October 2025;
Published online: 05 November 2025

References

1. Huang, S. et al. Industry 5.0 and society 5.0—comparison, complementation and co-evolution. *J. Manuf. Syst.* **64**, 424–428 (2022).
2. Li, C. et al. Unleashing mixed-reality capability in deep reinforcement learning-based robot motion generation towards safe human–robot collaboration. *J. Manuf. Syst.* **74**, 411–421 (2024).
3. Zafar, M. H., Langås, E. F. & Sanfilippo, F. Exploring the synergies between collaborative robotics, digital twins, augmentation, and industry 5.0 for smart manufacturing: a state-of-the-art review. *Rob. Comput. Integr. Manuf.* **89**, 102769 (2024).
4. Wang, T., Zheng, P., Li, S. & Wang, L. Multimodal human–robot interaction for human-centric smart manufacturing: a survey. *Adv. Intell. Syst.* **6**, 2300359 (2024).
5. Cini, F., Banfi, T., Ciuti, G., Craighero, L. & Controzzi, M. The relevance of signal timing in human-robot collaborative manipulation. *Sci. Rob.* **6**, eabg1308 (2021).
6. Wang, L. et al. Symbiotic human-robot collaborative assembly. *CIRP Ann.* **68**, 701–726 (2019).
7. Ding, P. et al. Dynamic scenario-enhanced diverse human motion prediction network for proactive human-robot collaboration in customized assembly tasks. *J. Intell. Manuf.* **36**, 4593–4612 (2024).
8. Wang, Y., Ma, H.-S., Yang, J.-H. & Wang, K.-S. Industry 4.0: a way from mass customization to mass personalization production. *Adv. Manuf.* **5**, 311–320 (2017).

9. Faccio, M. et al. Human factors in cobot era: A review of modern production systems features. *J. Intell. Manuf.* **34**, 85–106 (2023).
10. Bänziger, T., Kunz, A. & Wegener, K. Optimizing human–robot task allocation using a simulation tool based on standardized work descriptions. *J. Intell. Manuf.* **31**, 1635–1648 (2020).
11. Leng, J. et al. Towards resilience in industry 5.0: a decentralized autonomous manufacturing paradigm. *J. Manuf. Syst.* **71**, 95–114 (2023).
12. Rungtusanatham, M. J. & Salvador, F. From mass production to mass customization: Hindrance factors, structural inertia, and transition hazard. *Prod. Oper. Manage.* **17**, 385–396 (2008).
13. Lee, E., Barthelmey, A., Reckelkamm, T., Kang, H. Son, J. A study on human-robot collaboration based hybrid assembly system for flexible manufacturing. in *IECON 209--45th Annual Conference of the IEEE Industrial Electronics Society* Vol. **1**, 4197–4202 (2019).
14. Gamila, M. A. & Motavalli, S. A modeling technique for loading and scheduling problems in FMS. *Rob. Comput. Integr. Manuf.* **19**, 45–54 (2003).
15. Baruwa, O. T., Piera, M. A. & Guasch, A. Deadlock-free scheduling method for flexible manufacturing systems based on timed colored petri nets and anytime heuristic search. *IEEE Trans. Syst. Man Cybern, Syst.* **45**, 831–846 (2015).
16. Ahn, J. & Kim, H.-J. A branch and bound algorithm for scheduling of flexible manufacturing systems. *IEEE Trans. Automat. Sci. Eng.* **21**, 4382–4396 (2024).
17. El Atchi, S., Azzamouri, A. & Gautier, F. Toward sustainable operations: Integrating utility management into short-term scheduling in the phosphate industry. *Technol. Forecast. Soc. Change* **210**, 123861 (2025).
18. Huang, J., Li, X. & Gao, L. A novel GA-CP method for fixed-type multi-robot collaborative scheduling in flexible job shop. *IEEE Trans. Automat. Sci. Eng.* **22**, 13531–13543 (2025).
19. Firme, B., Figueiredo, J., Sousa, J. M. C. & Vieira, S. M. Agent-based hybrid tabu-search heuristic for dynamic scheduling. *Eng. Appl. Artif. Intell.* **126**, 107146 (2023).
20. Liu, Q., Li, X. & Gao, L. A novel MILP model based on the topology of a network graph for process planning in an intelligent manufacturing system. *Engineering.* **7**, 807–817 (2021).
21. Liu, Q., Li, X., Gao, L. & Fan, J. A multi-MILP model collaborative optimization method for integrated process planning and scheduling problem. *IEEE Trans. Eng. Manag.* **71**, 4574–4586 (2022).
22. Yao, Y. et al. A novel mathematical model for the flexible job-shop scheduling problem with limited automated guided vehicles. *IEEE Trans. Automat. Sci. Eng.* **22**, 7449–7462 (2024).
23. Gromicho, J. A. S., van Hoorn, J. J., Saldanha-da-Gama, F. & Timmer, G. T. Solving the job-shop scheduling problem optimally by dynamic programming. *Comput. Oper. Res.* **39**, 2968–2977 (2012).
24. Li, Y., Li, X., Gao, L. & Lu, Z. Multi-agent deep reinforcement learning for dynamic reconfigurable shop scheduling considering batch processing and worker cooperation. *Rob. Comput. Integr. Manuf.* **91**, 102834 (2025).
25. Zhang, F., Mei, Y., Nguyen, S. & Zhang, M. Survey on genetic programming and machine learning techniques for heuristic design in job shop scheduling. *IEEE Trans. Evol. Comput.* **28**, 147–167 (2024).
26. Li, Y., Gu, W., Yuan, M. & Tang, Y. Real-time data-driven dynamic scheduling for flexible job shop with insufficient transportation resources using hybrid deep Q network. *Rob. Comput. Integr. Manuf.* **74**, 102283 (2022).
27. Visentin, A., Prestwich, S., Rossi, R. & Tarim, S. A. Computing optimal (R,s,S) policy parameters by a hybrid of branch-and-bound and stochastic dynamic programming. *Eur. J. Oper. Res.* **294**, 91–99 (2021).
28. Xie, J., Gao, L., Li, X. & Gui, L. A hybrid genetic tabu search algorithm for distributed job-shop scheduling problems. *Swarm Evol. Comput.* **90**, 101670 (2024).
29. Lu, C., Gao, L., Li, X. & Xiao, S. A hybrid multi-objective grey wolf optimizer for dynamic scheduling in a real-world welding industry. *Eng. Appl. Artif. Intell.* **57**, 61–79 (2017).
30. Teng, Y., Li, X., Gao, L., Xie, J. & Li, Y. An improved co-evolutionary memetic algorithm based on novel schedule type and unconditional feasibility for hybrid flow-shop scheduling problem. *Comput. Ind. Eng.* **193**, 110324 (2024).
31. Huang, J.-P., Gao, L. & Li, X.-Y. An end-to-end deep reinforcement learning method based on graph neural network for distributed job-shop scheduling problem. *Expert Syst. Appl.* **238**, 121756 (2024).
32. Sun, X. et al. Deep reinforcement learning-based multi-objective scheduling for distributed heterogeneous hybrid flow shops with blocking constraints. *Engineering* **46**, 278–291 (2025).
33. Zhang, F., Mei, Y., Nguyen, S. & Zhang, M. Survey on genetic programming and machine learning techniques for heuristic design in job shop scheduling. *IEEE Trans. Evol. Comput*. 1–1 https://doi.org/10.1109/TEVC.2023.3255246 (2023).
34. Zhang, L., Li, Z., Królczyk, G., Wu, D. & Tang, Q. Mathematical modeling and multi-attribute rule mining for energy efficient job-shop scheduling. *J. Cleaner Prod.* **241**, 118289 (2019).
35. Holthaus, O. & Rajendran, C. Efficient dispatching rules for scheduling in a job shop. *Int. J. Prod. Econ.* **48**, 87–105 (1997).
36. Huang, J., Li, X., Gao, L., Liu, Q. & Teng, Y. Automatic programming via large language models with population self-evolution for dynamic job shop scheduling problem. Preprint at https://doi.org/10.48550/arXiv.2410.22657 (2024).
37. M Bran, A. et al. Augmenting large language models with chemistry tools. *Nat. Mach. Intell.* **6**, 525–535 (2024).
38. Wang, T., Fan, J. & Zheng, P. An LLM-based vision and language cobot navigation approach for human-centric smart manufacturing. *J. Manuf. Syst.* **75**, 299–305 (2024).
39. Kang, Y. & Kim, J. ChatMOF: An artificial intelligence system for predicting and generating metal-organic frameworks using large language models. *Nat. Commun.* **15**, 4705 (2024).
40. Mouret, J.-B. Large language models help computer programs to evolve. *Nature* **625**, 452–453 (2024).
41. Romera-Paredes, B. et al. Mathematical discoveries from program search with large language models. *Nature* **625**, 468–475 (2024).
42. Boiko, D. A., MacKnight, R., Kline, B. & Gomes, G. Autonomous chemical research with large language models. *Nature* **624**, 570–578 (2023).
43. Yuksekgonul, M. et al. Optimizing generative AI by backpropagating language model feedback. *Nature* **639**, 609–616 (2025).
44. Abhilash, P. M., Luo, X., Liu, Q., Madarkar, R. & Walker, C. Towards next-gen smart manufacturing systems: The explainability revolution. *npj Adv. Manuf.* **1**, 1–20 (2024).
45. Akiba, T., Shing, M., Tang, Y., Sun, Q. & Ha, D. Evolutionary optimization of model merging recipes. *Nat. Mach. Intell.* **7**, 195–204 (2025).
46. Ye, H. et al. ReEvo: large language models as hyper-heuristics with reflective evolution. In *The Thirty-Eighth Annual Conference on Neural Information Processing Systems* (2024).
47. Park, J., Chun, J., Kim, S. H., Kim, Y. & Park, J. Learning to schedule job-shop problems: Representation and policy learning using graph neural network and reinforcement learning. *Int. J. Prod. Res.* **59**, 3360–3377 (2021).
48. Tassel, P., Gebser, M. & Schekotihin, K. A reinforcement learning environment for job-shop scheduling. In *2021 PRL Workshop–Bridging the Gap Between AI Planning and Reinforcement Learning* (2021).
49. Liang, L. et al. KAG: Boosting LLMs in professional domains via knowledge augmented generation. *Proceedings of the ACM on Web Conference* (2025).
50. Zheng, Y. et al. LlamaFactory: Unified Efficient Fine-Tuning of 100+ Language Models. *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (2024)*.

Acknowledgements
This work was supported in part by National Natural Science Foundation of China under Grant no.52188102; in part by National Natural Science Foundation of China under Grant U21B2029; and in part by the Fundamental Research Funds for the Central Universities of China under Grant No 2024BRA004.

Author contributions
J.H. and Y.T. conceived the idea, designed and performed the numerical simulations and theoretical derivations and developed the code to realize the SeEvo. X.Y.L., J.H., Y.T., Q.H.L., C.J.Z., and G.Q.X. designed and performed the experiments. X.Y.L., J.H., Y.T., Q.H.L., C.J.Z. and L.G. wrote the paper and analyzed the numerical and experimental results. X.Y.L. and L.G. supervised the work. All the authors contributed to the discussion and revision of the manuscript.

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to Xinyu Li.