R-MALS: Role-Specialized Multi-Agent Scheduling with LLMs under Dynamic Disruptions

Yun Liu
College of Computer Science
Sichuan University
Chengdu, China
yliu@stu.scu.edu.cn

Yanan Sun*
College of Computer Science
Sichuan University
Chengdu, China
ysun@scu.edu.cn

***Abstract*—Dynamic Flexible Job-shop Scheduling plays a critical role in intelligent manufacturing, where scheduling decisions must be continuously adapted to dynamic disruptions. However, existing scheduling algorithms typically focus on either global rescheduling or local decision-making under dynamic disruptions. As a result, these algorithms often lack the ability to explicitly reason about the structural impact of disruptions on schedules, which may limit their effectiveness in dynamic environments. To address this limitation, we propose R-MALS, a role-specialized multi-agent framework that typically integrates large language models (LLMs) into the entire scheduling process to enable explicit reasoning about disruption propagation. Specifically, R-MALS decomposes scheduling into role-specialized agents for initial schedule generation, disturbance analysis, and localized rescheduling, allowing LLMs to reason about disruption propagation and adjust affected operations while preserving stable parts of the schedule. Furthermore, a two-stage training strategy that combines role-specific supervised fine-tuning with constraint-aware direct preference optimization is adopted to better align LLMs reasoning with scheduling constraints. We conduct extensive experiments on the Brandimarte and Fattahi benchmark against state-of-the-art algorithms. The results demonstrate that R-MALS achieves competitive makespan performance in static settings and maintains reliable rescheduling under machine breakdowns, highlighting the effectiveness of integrating LLMs into the entire scheduling process.**

***Index Terms*—Dynamic flexible job-shop scheduling, large language model, rescheduling, multi-agnet framework**

I. Introduction

Dynamic flexible job shop scheduling (DFJSS) is a complex combinatorial optimization problem that involves assigning multiple jobs to a set of machines with diverse processing capabilities [1], aiming to optimize objectives such as makespan [2] and production profit [3]. In DFJSS, each job consists of a sequence of operations, each processed on a set of candidate machines [4]. DFJSS involves two key decisions, i.e., machine assignment (selecting machines for operations) and operation sequencing (ordering operations on machines) [5]. DFJSS is often subject to dynamic disruptions, such as job arrivals and machine breakdowns [6], making it a challenging problem in modern manufacturing [7].

Existing DFJSS algorithms mainly include meta-heuristics, heuristic rules, and deep reinforcement learning (DRL). Meta-heuristics can produce high-quality solutions, but their iterative rescheduling process makes real-time response to frequent disruptions difficult [8]. Heuristic rules improve efficiency through hand-crafted dispatching rules, yet often suffer from limited adaptability [9]. DRL further enhances adaptability by learning scheduling policies from interactions with the environment [10]. However, DRL typically follows a stepwise decision-making paradigm and relies primarily on numerical state representations, which may limit explicit reasoning about how disruptions affect the schedule as a whole [11]. As a result, DRL-based schedulers tend to respond to disruptions in a locally reactive manner, resulting in unnecessary rescheduling actions and reduced stability in dynamic environments.

Large Language Models (LLMs) [12] have shown promise for structured reasoning over complex symbolic representations, leading to increasing interest in their use for DFJSS [13]. Existing studies mainly employs LLMs as auxiliary modules, such as generating dispatching rules and search operators [14], [15], or providing high-level strategy selection for DRL-based schedulers [16], [17]. While these studies improve adaptability, they still leave core scheduling decisions to heuristics or learned policies, limiting explicit reasoning about disruption propagation under dynamic disruptions. This motivates more direct integration of LLMs into the scheduling process.

Motivated by this observation, we propose R-MALS, an end-to-end LLM-driven scheduling framework for adaptive decision-making under dynamic disruptions. Unlike existing algorithms that treat LLMs as auxiliary components, R-MALS integrates LLMs directly into the scheduling process to explicitly reason about disruption impacts and support stable schedule adaptation in dynamic environments. The main contributions of this paper are as follows:

- We propose R-MALS, a novel end-to-end LLM-driven scheduling paradigm for adaptive decision-making under dynamic disruptions. In R-MALS, LLMs are directly integrated into the entire scheduling process to reason about schedule structures and disruption propagation, rather than being used as auxiliary components.
- We design a role-specialized multi-agent strategy that decomposes scheduling into initial generation, disruption analysis, and localized rescheduling. By assigning each agent a well-defined decision scope and output structure, the proposed strategy avoids unnecessary global

* Corresponding author: Yanan Sun

rescheduling under frequent disruptions.

- We introduce a two-stage training strategy that combines role-specific supervised fine-tuning with constraint-aware direct preference optimization. By constructing negative samples that violate scheduling constraints, the strategy effectively bridges the gap between the latent reasoning of LLMs and the rigid symbolic constraints of DFJSS, facilitating reliable and executable scheduling decisions.
- Extensive experiments on standard benchmark datasets demonstrate the superior performance of R-MALS. Additional ablation studies and detailed analyses further verify the effectiveness of the core design elements.

II. Related Work

A. Scheduling Algorithms

Existing scheduling algorithms can be categorized into meta-heuristics, heuristics, and DRL-based algorithms. Meta-heuristics, such as genetic algorithms [18] and particle swarm optimization [19], typically address disruptions by triggering global rescheduling. These algorithms are computationally expensive and may unnecessarily modify unaffected schedule regions [11]. Heuristics improve real-time efficiency via dispatching rules [20], but their myopic decision logic makes it difficult to capture disruption propagation across operational dependencies and resource constraints [15]. DRL-based algorithms sequentially generate scheduling actions based on the job-shop state [10]. However, reliance on state representations and local rewards may limit reasoning about the structural impact of disruptions on the schedule [21]. As a result, existing algorithms either rely on expensive global rescheduling or react to disruptions locally, motivating algorithms that explicitly reason about disruption effects without full rescheduling.

B. Combination Optimization based on LLMs

LLMs have attracted increasing attention in combinatorial optimization due to their strong generative and reasoning capabilities [15], [22]. Existing studies mainly integrate LLMs with classical optimization algorithms in two ways. One line of research leverages LLMs to assist the construction of solutions by generating heuristic rules, as shown in Figure 1(a). For example, LLMs have been combined with evolutionary search frameworks to guide the generation of heuristics [14] or search guidance for evolutionary optimization [23]. Another line treats LLMs as auxiliary operators within existing optimization pipelines, as shown in Figure 1(b). Representative works employ LLMs to dynamically guide operator selection in evolutionary algorithms [24] or to adapt state representations and reward functions in DRL [17]. Despite their potential, LLMs are not yet tightly integrated into the core optimization process, motivating the integration of LLMs into the optimization loop for direct decision-making, as shown in Figure 1(c).

III. The Proposed Algorithm

A. Overall Framework

This paper presents a role-specialized multi-agent scheduling algorithm, named R-MALS, which integrates LLMs into

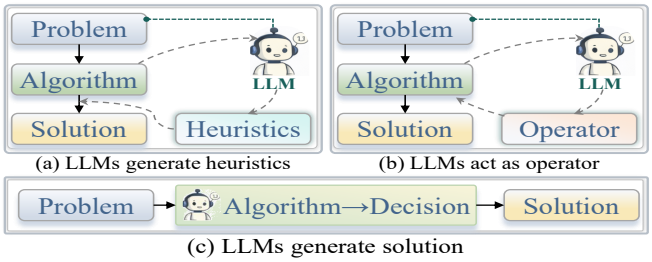


Fig. 1. Primary paradigms for using LLMs in optimization.

core decision-making for DFJSS. As shown in Figure 2, R-MALS follows a generate–analyze–repair paradigm, decomposing scheduling into initial generation, disruption analysis, and localized adaptation. Specifically, a Scheduler constructs an initial feasible schedule. When dynamic disruptions occur, an Analyzer identifies impacted operations, and a Rescheduler performs localized updates while preserving unaffected regions. To ensure feasibility, a differential repair strategy is applied to correct residual violations with minimal edits. The entire pipeline is further supported by a two-stage training strategy for role-consistent and constraint-aware generation. The following subsections will detail the role-specialized multi-agent design, differential repair, and training strategy.

B. Role-Specialized Multi-Agent Design

To address the combinatorial complexity and frequent disruptions in DFJSS, we adopt a role-specialized multi-agent design that decomposes scheduling into three coordinated agents, i.e., $\mathcal{R} = \{\textsc{Scheduler}, \textsc{Analyzer}, \textsc{Rescheduler}\}$. All agents follow a unified LLM-based agent paradigm but differ in their output space. This design is motivated by the distinct reasoning requirements of different scheduling stages. Specifically, initial scheduling requires structured generation, disturbance analysis demands causal reasoning, and rescheduling requires minimal and stable modifications to existing solutions.

Scheduler: Initial schedule generation. The Scheduler is responsible for generating an initial feasible schedule for a static scheduling instance. Directly generating explicit start and completion times poses inherent challenges for LLMs, as maintaining global temporal consistency across a large number of interdependent operations is particularly difficult under long scheduling horizons. To mitigate this limitation, we avoid direct time prediction and instead formulate scheduling as a structured generation problem over discrete machine-wise operation sequences, as defined in Eq. (1):

$$\mathcal{S}_0 = \{\mathcal{O}_m\}_{m\in\mathcal{M}}, \mathcal{O}_m = [(j_1, o_1), \ldots, (j_n, o_l)]. \quad (1)$$

By constraining the LLM to output sequences rather than absolute timestamps, the search space is reduced from a continuous time domain to a discrete combinatorial structure. After that, the sequences are subsequently mapped to executable schedules via a deterministic decoding procedure $\Gamma : \mathcal{S}_0 \to \mathbb{R}^+$ that enforces precedence and machine exclusivity constraints.

Analyzer: Disturbance Analysis. The Analyzer is triggered by dynamic events such as machine breakdowns. Rather than

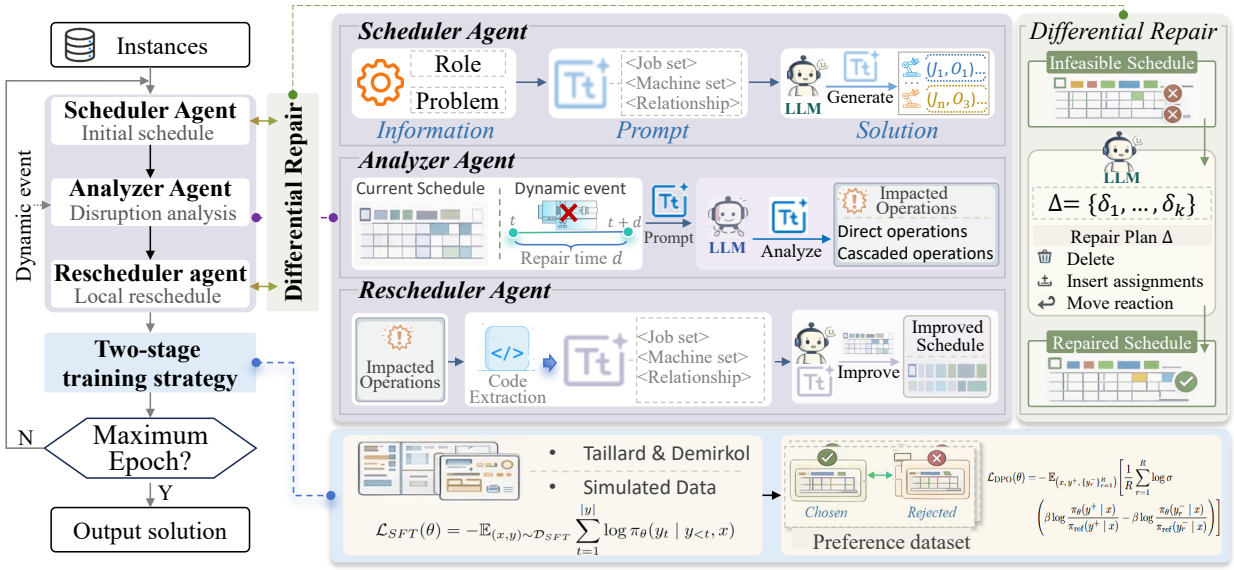


Fig. 2. Framework of the proposed R-MALS algorithm.

regenerating schedules, its role is to diagnose the structural impact of a disturbance on the current schedule. Given a disruption event $\mathcal{E}$, the Analyzer takes as input a structured scheduling context $\Sigma_{\text{anl}} = \{\mathcal{S}_{\text{curr}}, \mathcal{E}, \mathcal{J}\}$, where $\mathcal{S}_{\text{curr}}$ is the current executable schedule, $\mathcal{E}$ encodes the disruption attributes, and $\mathcal{J}$ specifies job-level precedence relations. Based on this context, the Analyzer outputs an affected operation set $\mathcal{O}_{\text{impacted}} \subseteq \mathcal{O}$ to determine the rescheduling scope. Specifically, the Analyzer distinguishes direct and cascaded impacts. Directly affected operations are those assigned to the failed machine whose execution intervals overlap with the disruption window, defined in Eq. (2):

$$\mathcal{I}_{\text{dir}} = \{o \in \mathcal{O} \mid \text{Alloc}(o) = M_{\text{fail}} \ \wedge \ [S_o, C_o] \cap [t_f, t_f + d_r] \neq \emptyset\}. \quad (2)$$

Operations that are not directly blocked may still become infeasible due to violated precedence constraints, yielding cascaded impacts $\mathcal{I}_{\text{cas}}$. The final impacted set is therefore defined as $\mathcal{O}_{\text{impacted}} = \mathcal{I}_{\text{dir}} \cup \mathcal{I}_{\text{cas}}$. Operations outside $\mathcal{O}_{\text{impacted}}$ are treated as stable, allowing the Rescheduler to perform localized repair while preserving unaffected schedule regions.

Rescheduler: Localized schedule adaptation. Given the affected operation set $\mathcal{O}_{\text{impacted}}$, the Rescheduler is responsible for updating the current schedule to mitigate disruption effects. To ensure that the Rescheduler produces feasible and stable updates, we design a structured prompt (shown in Figure 3) that explicitly specifies (i) the role and task, (ii) the impacted operations annotated with their impact levels, and (iii) the stable operations. Directly affected operations are treated as mandatory rescheduling targets, while cascaded ones are modified only when necessary to satisfy precedence or machine exclusivity constraints. Operations outside $\mathcal{O}_{\text{impacted}}$ remain fixed during rescheduling. Under these prompt constraints, the Rescheduler generates an updated machine-wise operation sequence, which is subsequently decoded into an executable schedule via deterministic constraint enforcement.

C. Differential Repair Strategy

Although the Scheduler and Rescheduler are trained to generate structured scheduling decisions, infeasible solutions may still occur due to the brittleness of autoregressive generation and the tight coupling of scheduling constraints. Regenerating the entire schedule to correct such local errors is inefficient. To address this issue, we introduce a Differential Repair (DR) strategy that performs minimal incremental edits on infeasible schedules. Given a locally infeasible schedule $\mathcal{S}$, DR first applies deterministic validation to identify constraint violations. Based on the resulting violation summary, an LLM generates a repair plan $\Delta = \{\delta_1, \ldots, \delta_K\}$, where each atomic repair action is defined as Eq. (3):

$$\delta = < a, o, m_{src}, m_{dst}, p >, \quad (3)$$

where $a \in \{\text{DELETE}, \text{INSERT}, \text{MOVE}\}$ and $o = (j, o)$ denoting an operation. DELETE removes invalid assignments, INSERT places missing operations on feasible machines, and MOVE reassigns operations to resolve local conflicts. The generated actions are then applied sequentially to obtain the repaired schedule $\mathcal{S}'$, followed by deterministic re-validation. If violations remain, the repair process is repeated. In this way, DR corrects infeasible schedules by editing only the affected parts while preserving already valid regions.

D. Two-stage Training Strategy

To improve the reliability of LLM-generated scheduling decisions under strict feasibility constraints, we adopt a two-stage training strategy, denoted as $\mathcal{P} = \{\mathcal{L}_{SFT}, \mathcal{L}_{DPO}\}$, which transitions LLM from basic instruction-following to constraint-aware preference alignment.

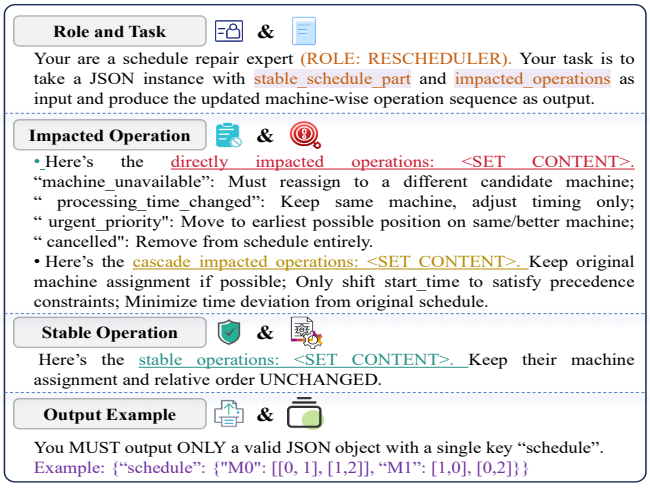


Fig. 3. A simple example illustrates the prompt used by the Rescheduler.

Stage 1: Role-Specific Supervised Fine-Tuning (SFT). In the first stage, we construct an SFT dataset $\mathcal{D}_{SFT} = \{(x, y)\}$ from FJSP benchmarks [25], [26] and additional simulation instances [27], where x denotes the problem description and y is the target machine-wise sequence. The LLM is trained with the standard cross-entropy loss in Eq. (4):

$$\mathcal{L}_{SFT}(\theta) = -\mathbb{E}_{(x,y)\sim\mathcal{D}_{SFT}} \sum_{t=1}^{|y|} \log \pi_\theta(y_t \mid y_{<t}, x). \tag{4}$$

This stage mainly teaches the model to generate role-consistent and structurally valid scheduling outputs.

Stage 2: Constraint-Aware Direct Preference Optimization (DPO). To further reduce infeasible generations, we construct a preference dataset $\mathcal{D}_{pref} = \{(x, y_w, \{y_r\}_{r=1}^{R})\}$, where y_w is a feasible schedule and $\{y_r\}_{r=1}^{R}$ are infeasible schedules with representative constraint violations. The DPO objective encourages LLMs to assign a higher likelihood to feasible outputs than infeasible ones relative to a reference policy π_{ref}. Formally, the loss is defined as Eq. (5):

$$\mathcal{L}_{\text{DPO}}(\theta) = -\,\mathbb{E}_{\left(x,\, y^+,\, \{y_r^-\}_{r=1}^{R}\right)} \left[\frac{1}{R} \sum_{r=1}^{R} \log \sigma \left(\beta \log \frac{\pi_\theta(y^+ \mid x)}{\pi_{\text{ref}}(y^+ \mid x)} - \beta \log \frac{\pi_\theta(y_r^- \mid x)}{\pi_{\text{ref}}(y_r^- \mid x)} \right) \right], \tag{5}$$

where β controls the strength of preference alignment. In this way, hard scheduling constraints are gradually absorbed into the generation behavior of the model, reducing infeasible outputs and the need for subsequent repair.

IV. Experiments

A. Experimental Details

Datasets. To evaluate R-MALS in static and dynamic scenarios, we use the Brandimarte benchmark (MK01–MK10) [28] for static scheduling [4] and the Fattahi benchmark (MFJS01–MFJS10) [6], [29] for dynamic scheduling with machine breakdowns. Each MFJS instance simulates a random machine failure, with repair duration proportional to busy time with coefficients ($\beta_m^1 = 0.15$, $\beta_m^2 = 0.25$) [30], modeling different unavailability levels [6].

Peer Competitors. We evaluate R-MALS against four categories of competitors: (1) Heuristic: FIFO+SPT [2] as a baseline; (2) Meta-heuristics: GP [31] and IGAR [32] for their global search capability; (3) DRL: PPO [33], a representative RL-based scheduler; and (4) LLM-enhanced: LLM4EO [22], representing LLMs as auxiliary components in optimization.

Parameter Setting. We implement the Scheduler and Rescheduler using Qwen2.5-7B, fine-tuned via LoRA-based SFT followed by DPO with learning rates 1×10^{-5} and 5×10^{-4} [34]. DPO adopts sigmoid loss with a KL penalty coefficient $\beta = 0.1$, trained for 3 epochs with batch size 64 and 6k-token sequences. The Analyzer uses frozen Qwen3-Max without fine-tuning. All experiments are conducted on a server with six NVIDIA RTX 3090 GPUs.

B. Overall Results

Table I reports the makespan of R-MALS versus five competitors on MFJS instances under two machine repair time settings. R-MALS consistently achieves the minimum makespan. Under moderate repair durations (β_m^1), it outperforms FIFO+SPT by 6.9%–16.9% and improves over meta-heuristics and PPO by up to 7.35%. Compared with LLM4EO, R-MALS demonstrates that direct LLM-driven scheduling can match or exceed LLM-assisted optimization, with advantages increasing under longer repair times (β_m^2). For example, in MFJS07, the gaps against IGAR and PPO reach 7.04% and 6.55%, respectively. In complex instances like MFJS10, R-MALS maintains a 4% lead over GP. Overall, R-MALS shows robust performance across disruption levels and problem scales, highlighting its promise for dynamic rescheduling.

C. Ablation Study

(1) Performance of the Scheduler: Table II compares R-MALS with classic meta-heuristics (GA, SLABC [35]), multi-agent systems (DMAS [36], GMAS [37]), and the LLM-assisted LLM4EO [22] on MK01–10. R-MALS achieves the best or competitive makespan, obtaining the best results on 6 instances and matching the best on MK08. R-MALS consistently outperforms GA and DRL baselines, especially on medium-scale instances (MK06, MK07), and generally matches or surpasses LLM4EO, demonstrating that direct LLM integration can generate effective initial solutions when properly constrained and trained.

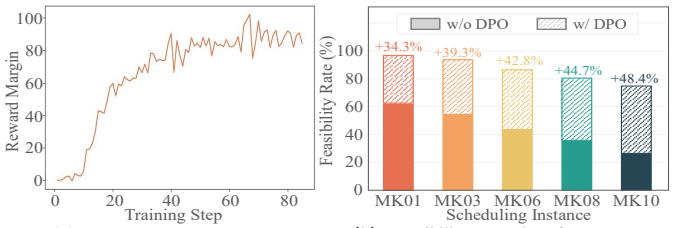
(2) Effectiveness of Constraint-Aware DPO: Figure 4 presents the ablation study on the effect of the constraint-aware DPO, where w/o DPO denotes R-MALS with SFT only, and w/ DPO is R-MALS with the two-stage training strategy. As shown in Figure 4(a), the reward margin increases steadily, indicating that the LLM effectively learns to prioritize feasible schedules over infeasible ones. Moreover, Figure 4(b) reports that incorporating DPO yields substantial feasibility

TABLE I
MAKESPAN RESULTS OF R-MALS WITH PEER COMPETITORS ON MFJS01–MFJS10 DATASETS WITH TWO DIFFERENT REPAIR TIME SETTINGS.

	Instances	MFJS01	MFJS02	MFJS03	MFJS04	MFJS05	MFJS06	MFJS07	MFJS08	MFJS09	MFJS10
β_m^1	FIFO+SPT	497.00	482.00	528.00	568.00	625.00	730.00	935.00	952.00	1243.00	1384.00
	GP	468.86	448.00	466.08	565.23	519.46	641.05	910.57	950.96	1171.46	1303.23
	IGAR	468.71	468.77	468.63	**545.30**	514.18	634.10	881.46	884.09	1097.85	1275.07
	PPO	469.39	448.96	466.32	568.71	527.33	635.87	889.10	905.67	1109.74	1258.28
	LLM4EO	467.86	446.22	**462.03**	565.52	**512.83**	634.37	880.21	882.21	1104.60	**1234.07**
	R-MALS	**462.56**	**448.33**	462.53	554.26	519.66	**631.71**	**879.03**	**881.00**	**1088.37**	1236.81
β_m^2	FIFO+SPT	575.00	503.00	619.00	731.00	731.00	883.00	1371.00	1332.00	1571.00	1821.00
	GP	524.30	500.72	583.53	728.87	719.30	829.70	1203.67	1225.15	1496.83	1745.22
	IGAR	555.70	**497.53**	614.00	713.93	719.81	839.70	1206.63	1222.35	1506.57	1748.61
	PPO	545.63	496.57	564.26	711.52	716.70	833.29	1200.27	1255.05	1496.59	1745.22
	LLM4EO	523.61	501.00	562.31	686.97	709.65	828.86	**1121.03**	1180.37	1485.93	**1675.36**
	R-MALS	**520.30**	**497.53**	**558.25**	**686.52**	**703.71**	**826.33**	1121.66	**1178.61**	**1483.26**	1678.33

TABLE II
MAKESPAN OF R-MALS AND COMPETITORS ON MK01–MK10.

Instances	GA	SLABC	DMAS	GMAS	LLM4EO	R-MALS
MK01	40.80	43.00	42.00	39.00	40.60	**40.00**
MK02	28.00	31.50	29.00	**26.00**	27.60	**26.00**
MK03	**204.00**	219.40	213.00	**204.00**	**204.00**	205.00
MK04	64.10	75.60	87.00	60.00	62.60	**61.00**
MK05	176.00	191.90	184.00	**172.00**	175.80	**172.00**
MK06	65.20	89.40	73.00	58.00	65.20	**56.00**
MK07	144.90	159.30	162.00	137.00	145.10	**135.00**
MK08	**523.00**	525.30	555.00	**523.00**	**523.00**	**523.00**
MK09	313.20	379.10	348.00	307.00	313.00	**305.00**
MK10	229.00	294.00	250.00	**197.00**	225.50	**197.00**

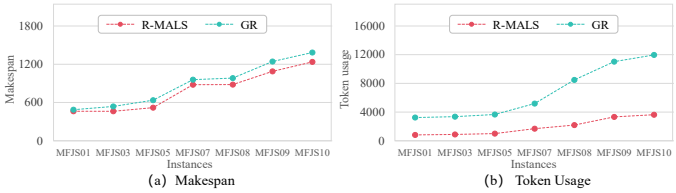


(a) Reward margin of DPO trainer (b) Feasibility rate gains of DPO over SFT

Fig. 4. The effectiveness of the DPO trainer.

improvements across five representative instances, with gains of $+34.3\%$ to $+48.4\%$ over the SFT-only baseline. These results demonstrate that DPO is critical for internalizing hard scheduling constraints into the generation behavior of LLM.

(3) Impact of the Analyzer: Figure 5 compares R-MALS with a variant (i.e., GR) that feeds initial schedules and events directly into the Rescheduler without disturbance analysis. As shown in Figure 5(a), R-MALS achieves lower makespans across all scales, indicating that identifying impacted operations helps avoid unnecessary modifications and preserves solution quality under machine breakdowns. Meanwhile, Figure 5(b) shows that R-MALS requires significantly fewer tokens, confirming that the Analyzer effectively constrains the decision scope by filtering unaffected operations.



(a) Makespan (b) Token Usage

Fig. 5. The impact of the Analyzer.

TABLE IV
COMPARISON OF THE TIME COMPLEXITY.

Category	Algorithms	Time Complexity
Heuristic	FIFO+SPT	$O(\|\mathcal{O}\|k)$
Meta-heuristics	GP	$O(GP\|\mathcal{O}\|k\|T\|)$
	IGAR	$O(GPr\|\mathcal{O}\|k)$
DRL	PPO	$O(Ld\|\mathcal{O}\|^2)$
LLMs-based	LLM4EO	$O(GP\|\mathcal{O}\|k + N_{\text{evo}}T_{\text{LLM}})$
	R-MALS	$O((2E+1)T_{\text{LLM}} + (E+1)\|\mathcal{O}\|)$

D. Further Analysis of Time Complexity

Let $|\mathcal{O}|$ be the number of operations, k the feasible actions per decision, G and P the generations and population size, E the frequency of dynamic events, N_{evo} the number of evolution triggers, and T_{LLM} the LLM inference cost. Table III shows that FIFO+SPT has linear complexity, PPO incurs a higher polynomial cost from graph-based encoding, and meta-heuristics add overhead from iterative population search. LLM4EO further increases cost via LLM-driven operator evolution. In contrast, R-MALS avoids iterative search, with complexity dominated by T_{LLM}. Despite the LLM overhead, R-MALS still demonstrates industrial potential because industrial disruptions are sporadic, and LLM overhead is manageable.

V. CONCLUSION

This paper investigates the potential of LLMs in the entire scheduling process under dynamic disruptions. To this end, we propose R-MALS, a role-specialized multi-agent framework

coordinating LLMs in schedule generation, disturbance analysis, and localized rescheduling. With a two-stage training strategy for constraint alignment, R-MALS achieves competitive performance on benchmark datasets. Future work will explore scalability to larger instances and more complex disruptions.

REFERENCES

[1] C. Su, C. Zhang, D. Xia, B. Han, C. Wang, G. Chen, and L. Xie, "Evolution strategies-based optimized graph reinforcement learning for solving dynamic job shop scheduling problem," *Applied Soft Computing*, vol. 145, p. 110596, 2023.

[2] K. Lei, P. Guo, Y. Wang, J. Xiong, and W. Zhao, "An end-to-end hierarchical reinforcement learning framework for large-scale dynamic flexible job-shop scheduling problem," in *2022 International joint conference on neural networks (IJCNN)*. IEEE, 2022, pp. 1–8.

[3] C. Zhang, M. Dong, and K. Ota, "Enabling computational intelligence for green internet of things: Data-driven adaptation in lpwa networking," *IEEE Computational Intelligence Magazine*, vol. 15, no. 1, pp. 32–43, 2020.

[4] G. Zhang, X. Lu, X. Liu, L. Zhang, S. Wei, and W. Zhang, "An effective two-stage algorithm based on convolutional neural network for the bi-objective flexible job shop scheduling problem with machine breakdown," *Expert Systems with Applications*, vol. 203, p. 117460, 2022.

[5] L. Zhu, F. Zhang, X. Zhu, K. Chen, and M. Zhang, "Phenotype and genotype based sample aware surrogate-assisted genetic programming in dynamic flexible job shop scheduling," *IEEE Transactions on Artificial Intelligence*, vol. 6, no. 12, pp. 3232–3247, 2025.

[6] C. Fan, W. Wang, and J. Tian, "Flexible job shop scheduling with stochastic machine breakdowns by an improved tuna swarm optimization algorithm," *Journal of manufacturing systems*, vol. 74, pp. 180–197, 2024.

[7] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, and M. Zhang, "Multitask genetic programming-based generative hyperheuristics: A case study in dynamic scheduling," *IEEE Transactions on Cybernetics*, vol. 52, no. 10, pp. 10 515–10 528, 2021.

[8] L. He, R. Chiong, W. Li, S. Dhakal, Y. Cao, and Y. Zhang, "Multiobjective optimization of energy-efficient job-shop scheduling with dynamic reference point-based fuzzy relative entropy," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 1, pp. 600–610, 2021.

[9] K. Salamun, I. Pavić, H. Džapo, and M. Đurasević, "Evolving scheduling heuristics with genetic programming for optimization of quality of service in weakly hard real-time systems," *Applied soft computing*, vol. 137, p. 110141, 2023.

[10] X. Wu, X. Yan, D. Guan, and M. Wei, "A deep reinforcement learning model for dynamic job-shop scheduling problem with uncertain processing time," *Engineering applications of artificial intelligence*, vol. 131, p. 107790, 2024.

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

[12] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian, "A comprehensive overview of large language models," *ACM Transactions on Intelligent Systems and Technology*, vol. 16, no. 5, pp. 1–72, 2025.

[13] H.-j. Oh, J.-w. Baek, K.-y. Cho, and J.-h. Woo, "Feasibility study on the interactive ai scheduling with llm technology for human-centric industry 5.0," in *IFIP International Conference on Advances in Production Management Systems*. Springer, 2025, pp. 419–434.

[14] J. Huang, X. Li, L. Gao, Q. Liu, and Y. Teng, "Automatic programming via large language models with population self-evolution for dynamic job shop scheduling problem," *arXiv preprint arXiv:2410.22657*, 2024.

[15] R. Li, L. Wang, H. Sang, L. Yao, and L. Pan, "Llm-assisted automatic memetic algorithm for lot-streaming hybrid job shop scheduling with variable sublots," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2025.

[16] J. Ding, J. Xia, Y. Ye, and M. Chen, "Effective reinforcement learning-based dynamic flexible job shop scheduling using two-stage dispatching," *Journal of Systems Architecture*, vol. 172, p. 103664, 2026.

[17] T.-Y. Hong and K.-H. Li, "Large language model driven adaptive deep reinforcement learning with dynamic definition refinement for flexible job shop scheduling problem," *Applied Soft Computing*, p. 114253, 2025.

[18] W. Yu, L. Zhang, and N. Ge, "An adaptive multiobjective evolutionary algorithm for dynamic multiobjective flexible scheduling problem," *International Journal of Intelligent Systems*, vol. 37, no. 12, pp. 12 335–12 366, 2022.

[19] Q. Zhang and B. Zhang, "Research on dynamic flexible job shop scheduling problem based on discrete particle swarm optimization with readjustment strategy," in *Tenth International Conference on Mechanical Engineering, Materials, and Automation Technology (MMEAT 2024)*, vol. 13261. SPIE, 2024, pp. 1093–1097.

[20] W. Weng, J. Chen, M. Zheng, and S. Fujimura, "Realtime scheduling heuristics for just-in-time production in large-scale flexible job shops," *Journal of Manufacturing Systems*, vol. 63, pp. 64–77, 2022.

[21] Y. Gui, D. Tang, H. Zhu, Y. Zhang, and Z. Zhang, "Dynamic scheduling for flexible job shop using a deep reinforcement learning approach," *Computers & Industrial Engineering*, vol. 180, p. 109255, 2023.

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

[23] X. Wu, D. Wang, C. Wu, L. Wen, C. Miao, Y. Xiao, and Y. Zhou, "Efficient heuristics generation for solving combinatorial optimization problems using large language models," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, 2025, pp. 3228–3239.

[24] Y. Zhang and G. Yi, "Laos: Large language model-driven adaptive operator selection for evolutionary algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2025, pp. 517–526.

[25] E. Taillard, "Benchmarks for basic scheduling problems," *european journal of operational research*, vol. 64, no. 2, pp. 278–285, 1993.

[26] E. Demirkol, S. Mehta, and R. Uzsoy, "Benchmarks for shop scheduling problems," *European Journal of Operational Research*, vol. 109, no. 1, pp. 137–141, 1998.

[27] M. Xu, Y. Mei, F. Zhang, and M. Zhang, "Genetic programming with lexicase selection for large-scale dynamic flexible job shop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 5, pp. 1235–1249, 2024.

[28] P. Brandimarte, "Routing and scheduling in a flexible job shop by tabu search," *Annals of Operations research*, vol. 41, no. 3, pp. 157–183, 1993.

[29] P. Fattahi, M. Saidi Mehrabad, and F. Jolai, "Mathematical modeling and heuristic approaches to flexible job shop scheduling problems," *Journal of intelligent manufacturing*, vol. 18, no. 3, pp. 331–342, 2007.

[30] N. Al-Hinai and T. Y. ElMekkawy, "Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm," *International Journal of Production Economics*, vol. 132, no. 2, pp. 279–291, 2011.

[31] R. Braune, F. Benda, K. F. Doerner, and R. F. Hartl, "A genetic programming learning approach to generate dispatching rules for flexible shop scheduling problems," *International Journal of Production Economics*, vol. 243, p. 108342, 2022.

[32] M. K. Amjad, S. I. Butt, and N. Anjum, "Improved genetic algorithm integrated with scheduling rules for flexible job shop scheduling problems," in *E3S Web of Conferences*, vol. 243. EDP Sciences, 2021, p. 02010.

[33] L. Lv, C. Zhang, and W. Shen, "A deep reinforcement learning-based rescheduling method for flexible job shops under machine breakdowns," in *2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, 2024, pp. 1734–1739.

[34] Z. Liu, X. Sun, and Z. Zheng, "Enhancing llm safety via constrained direct preference optimization," *arXiv preprint arXiv:2403.02475*, 2024.

[35] R. Chen, B. Yang, S. Li, and S. Wang, "A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem," *Computers & industrial engineering*, vol. 149, p. 106778, 2020.

[36] A. Maoudj, B. Bouzouia, A. Hentout, A. Kouider, and R. Toumi, "Distributed multi-agent scheduling and control system for robotic flexible assembly cells," *Journal of intelligent manufacturing*, vol. 30, no. 4, pp. 1629–1644, 2019.

[37] X. Jing, X. Yao, M. Liu, and J. Zhou, "Multi-agent reinforcement learning based on graph convolutional network for flexible job shop scheduling," *Journal of Intelligent Manufacturing*, vol. 35, no. 1, pp. 75–93, 2024.