NL ⇒ Schedule: Evaluate Multitask Scheduling Capability of Large Language Models

Wenrui Liao[1] **Weihong Du**[1] **Yi Li**[1] **Hongru Liang**[2]* **Wenqiang Lei**[1]
[1]College of Computer Science, Sichuan University, China
[2]School of Artificial Intelligence, Sichuan University, China
{liaowenrui, duweihong, liyi2}@stu.scu.edu.cn
{lianghongru, wenqianglei}@scu.edu.cn

Abstract

Automated schedule generation for multitask from natural language descriptions has huge potential in modern industry. While classic methods bypass language complexities by using preformatted matrices, and recent LLM+solver approaches introduce new fragilities by relying on solver-specific code generation. This raises critical questions: *Can large language models (LLMs) solve this* NL ⇒ Schedule *task end-to-end well (RQ1)?* If the answer is "no", *where do they fall short (RQ2)?* And *how can their capabilities be enhanced (RQ3)?* To answer these questions, we introduce NL ⇒ Schedule, the first benchmark for this task, equipped with a dataset of 240 description-schedule pairs constructed from real-world materials and a rigorous evaluation suite. Our evaluation of nine state-of-the-art LLMs reveals the limitations of different LLMs in procedure grounding and the strengths of advanced LLMs in global planning via local analysis. To address these shortcomings, we propose MANS, a novel multi-agent framework. Extensive experiments show that MANS achieves more robust performance comparable to six state-of-the-art LLM+solver methods. We hope NL ⇒ Schedule and MANS will serve as a solid foundation for automatic scheduling. The code and dataset are available in https://github.com/SCUNLP/NL2Schedule

1 Introduction

Scheduling, arranging multitask under the constraint of limited resources , is critical to the efficiency of modern manufacturing and service industries (Conway, 1967; Pinedo, 2022). Automated generation of schedules for multitask thus has huge potential, as it reduces the reasoning workload of human experts. However, most existing methods rely on experts who possess both scheduling skills and domain knowledge, resulting in high costs (Powell and Riccardi, 2025).

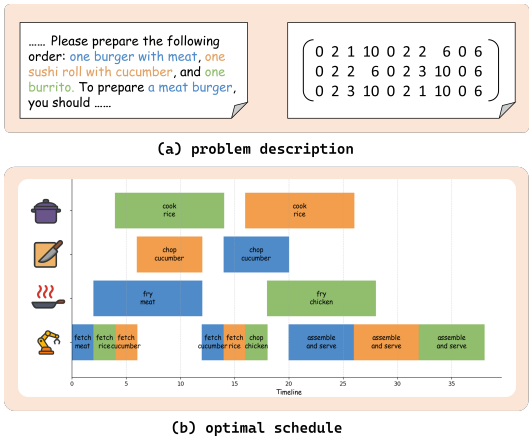

Figure 1: An example of multitask schedule.

Here, we call attention to a vital task, NL ⇒ Schedule. In this task, given a natural language multitask problem description (e.g. Figure 1 (a) left), a model is required to generate a feasible schedule (e.g. Figure 1 (b)). It seems like a NL2SQL task (Lei et al., 2020) but more challenging. This task is not merely extraction, translation or summarization of the description, it requires the model to understand multi goals, identify constraints, and reason over multi steps, so that to induce (⇒) a feasible schedule.

However, most existing researches oversimplify this task (Da Col and Teppan, 2019; Shady et al., 2022; Zhang et al., 2020; Liu et al., 2024b). Instead of the natural language, they directly input synthetic instances formatted in matrices (e.g. Figure 1 (a) right). This creates a big research gap between NL ⇒ Schedule and Matrix2Schedule, as the former is an inductive reasoning task, the latter reduces to a mathematical NP-hard optimization problem. And there also remains a significant gap between real-world scenarios and synthetic scenarios. To mitigate these gaps, a recent line of work has begun exploring the LLM+solver paradigm (Xiao et al., 2024; AhmadiTeshnizi et al., 2024; Huang et al., 2025) on optimization modeling

* Corresponding author

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 35620–35640

task. These methods follow a three-stage pipeline: model formulation, to abstract the text into a mathematical model (e.g. variables, constraints, objectives); code generation, to translate the mathematical model into solver-compatible code; and external solving, where a separate solver computes the final schedule. However, this design introduces new fragilities: the mathematical model and generated code highly depend on the requirements of different solvers, including but not limited to specific API formats, constraints (matrix shape, etc.). This largely hinders the robust deployment of these methods in real world NL ⇒ Schedule scenarios. Recent LLMs have shown advantages in language understanding (Achiam et al., 2023), information extraction (Dagdelen et al., 2024), and planning (Valmeekam et al., 2023). These developments bring new hope to the NL ⇒ Schedule task — *whether LLMs can well solve this problem in an end-to-end manner (RQ1)*, instead of relying on the fragile pipeline. If the answer is "no", we are interested in *where do current LLMs fall short (RQ2)*. Moreover, *how can we improve LLMs to serve as a more reliable tool for* NL ⇒ Schedule *(RQ3)*.

To answer the above questions, we benchmark NL ⇒ Schedule with a novel dataset constructed from real-world materials, a systematical evaluation paradigm, and a series of state-of-the-art baselines. Specifically, we compose 240 description-schedule pairs from manufacturing, construction, pharmaceuticals and cooking. Each pair is well-aligned through a three-stage data2text process. We also design three metrics to measure the step completeness, task completeness and feasibility of the generated schedules, respectively. Further, as replies to RQ1 and RQ2, we test nine state-of-the-art LLMs on NL ⇒ Schedule and conduct in-depth analysis, revealing the limitations of different LLMs in procedure grounding and the strengths of advanced LLMs in global planning via local analysis (e.g., identify the bottleneck etc.). As a reply to RQ3, we propose a novel Multi-Agent framework for NL ⇒ Schedule, called MANS. It collaborates multiple agents to progressively reason about procedures and construct the final schedule. We then compare MANS against six LLM+solver approaches, demonstrating the priority of MANS over the LLM+solver approaches on generating near-optimal feasible schedules in more stable runtime. We hope NL ⇒ Schedule and MANS will serve as a solid foundation for automatic scheduling, and offer insights for complex problem planning in real-world scenarios. In summary, our contributions are as follows.

- We highlight a critical task in manufacturing and service industries, i.e., NL ⇒ Schedule. It requires the model to arrange multitask under limited resources.
- We benchmark NL ⇒ Schedule with a well-aligned dataset, carefully designed metrics and nine baselines.
- We systematically test the baselines in an end-to-end manner, revealing their failure in handling procedures.
- We propose a multi-agent framework, MANS, which suggests more robust performance can be achieved by integrated reasoning methods compared with pipeline baselines.

2 Related Work

2.1 Multitask Scheduling Problem

The Multitask Scheduling Problem is a fundamental NP-hard combinatorial optimization challenge. Existing methods range from fast but often suboptimal Priority Dispatching Rules (PDRs) (Panwalkar and Iskander, 1977; Holthaus and Rajendran, 1997; Shady et al., 2022), and exact solvers (MIP/CP) that guarantee optimality but face scalability limits (Ribeiro et al., 2015; Da Col and Teppan, 2019), to recent DRL-based approaches that learn scheduling strategies but may lack feasibility guarantees (Zhang et al., 2020; Song et al., 2022; Liu et al., 2024b). A critical limitation shared by these methods is their reliance on structured, matrix-formatted inputs, which contrasts with real-world needs often described in natural language (Sarsur et al., 2024).

To address this disconnect, we introduce NL ⇒ Schedule, the first benchmark for automated schedule generation from natural language, providing problem descriptions with corresponding task graphs and optimal schedules to evaluate the completeness and feasibility of generated solutions.

2.2 LLM for Optimization Modeling

Recent research has investigated using LLMs to translate optimization problem descriptions into executable solver code. For instance, (Xiao et al., 2024) proposed a multi-agent framework where specialized agents (e.g., modeler, programmer) solve problems via forward and backward reasoning. (AhmadiTeshnizi et al., 2024) utilizes

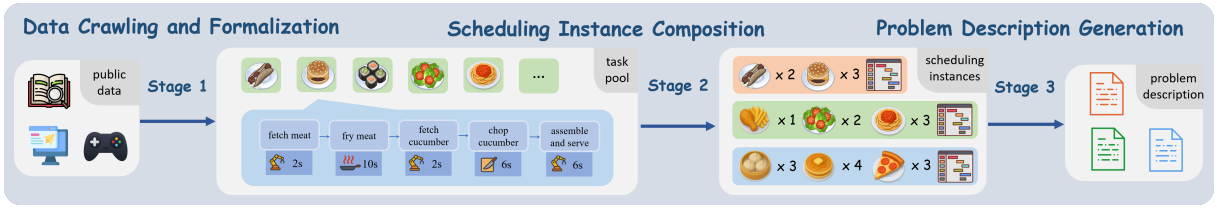

Figure 2: The dataset construction process.

a modular structure to iteratively improve mathematical models and code, enabling it to handle long and complex problem descriptions. Differently, (Huang et al., 2025) introduced a data synthesis framework to train and customize open-source LLMs specifically for optimization modeling and code generation.

This three-stage pipeline requires prior knowledge of the solver to be used, tightly coupling the system to specific solver interfaces and capabilities. This largely hinders the robust deployment of these methods in real world NL $\Rightarrow$ Schedule scenarios.

To address this, we investigated another paradigm where the LLM functions as an end-to-end solver. Our evaluation of state-of-the-art LLMs revealed both remarkable global planning potential and limitations in procedures grounding.

2.3 Datasets

Traditional multitask scheduling datasets (Muth, 1963; Lawrance, 1984; Adams et al., 1988; Applegate and Cook, 1991; Taillard, 1993) consist solely of formatted matrices embedding problem information and the instances are typically generated randomly or via rule-based procedures, making them statistically different from real-world scheduling distributions. Recent studies (Xiao et al., 2024; Huang et al., 2025) introduce datasets with natural language descriptions, but they contain only a handful of toy examples of multitask scheduling tasks.

To address these limitations, we build a dataset in which each scheduling instance includes a problem description, a task graph, and a optimal schedule. The dataset spans six problem scales and four real-world scenarios. To our knowledge, we are the first to construct a large-scale dataset of this kind, specifically designed to bridge natural language descriptions with formal, executable schedules.

2.4 Progress in LLMs

In recent years, LLMs have advanced rapidly and demonstrated excellent performance across a wide range of tasks, including language understanding(Achiam et al., 2023), information extraction (Dagdelen et al., 2024), and planning (Valmeekam et al., 2023). Moreover, reasoning LLMs have demonstrated expert-level performance in mathematics and coding(OpenAI Team, 2025b; Guo et al., 2025; Comanici et al., 2025; Yang et al., 2025). However, whether LLMs can solve the inductive reasoning tasks like multitask scheduling remains poorly understood. To this end, we provides the first comprehensive investigation in multitask scheduling. We introduce the NL $\Rightarrow$ Schedule to systematically evaluate state-of-the-art LLMs and propose our MANS framework, and compare MANS with pipeline-based methods.

3 Dataset Construction

Collecting large-scale scheduling problem descriptions and solutions is challenging, as such data often exist in private industrial cases. To this end, we take an alternative approach: instead of relying on annotated real-world corpora, we first compose real-world tasks into scheduling instances and generate corresponding natural language descriptions via a data2text pipeline inspired by (Du et al., 2024).

Formally, we define each instance by a set of tasks $\mathcal{T} = \{T_1, \ldots, T_n\}$ and a set of machines $\mathcal{M} = \{M_1, \ldots, M_m\}$. Each task T_i consists of a directed acyclic graph of steps, where each step O_{ij} is characterized by a required machine $m_{ij} \in \mathcal{M}$ and a processing duration p_{ij}. The objective is to find a feasible schedule that minimizes the makespan ($C_{\max}$), the completion time of the last step, subject to precedence and machine capacity constraints.

As shown in Figure 2 we construct our dataset via following three-stage pipeline.

3.1 Data Crawling and Formalization

First, we crawl a wide variety of data from public sources across four scenarios: manufacturing, construction, pharmaceuticals, and cooking. We identified 135 basic tasks and represent each task as a directed acyclic graph where nodes represent

Table 1: Details of tasks collected from public data. **#Task** represents the task quantity of the scenario. **#Machine** represents the unique machine types in the tasks. **#Avg_Step** represents the average step quantity in all tasks.

Scenario	#Task	#Machine	#Avg Step
Manufacturing	28	32	5.07
Construction	29	44	8.69
Pharmaceuticals	32	12	4.12
Cooking	46	20	7.42

steps annotated with description, machine requirement and processing duration and edges indicate precedence constraints between steps. The detailed statistical summaries are shown in Table 1, with data sources listed in Appendix A.1.

At this stage, we obtain a task pool consisting of 135 tasks. Only structured graph data is collected, as it is rare to find natural language descriptions that comprehensively specify all the information involved in a task.

3.2 Scheduling Instance Composition

To ensure our scheduling instances are both realistic and challenging, we develop a two-stage composition process. First, we group individual tasks into 13 "workshops" based on shared machine requirements, using the Jaccard similarity metric and K-Medoids clustering. Second, each final scheduling instance is generated by sampling a number of tasks—ranging from 5 to 10—exclusively from within a single pre-defined workshop. We generate 10 distinct instances for each scenario and task quantity, resulting a total of 240 unique scheduling instances. The optimal schedules are annotated by OR-Tools. This two-stage approach guarantees that the tasks in every scheduling instance heavily compete for a limited set of resources. Details are shown in Appendix A.2.

At this stage, we obtain 240 scheduling instances with task graphs and their optimal schedule.

3.3 Problem Description Generation

For each schedule instance, we generate a comprehensive natural language description composed of two parts: scheduling requirements overview, and procedures of each task.

Following the methodology in (Du et al., 2024), we adopt a data2text generation process. We curate 20 scheduling requirement templates from publicly available contracts(TemplateLab, 2025) and order dialogues(Rubino et al., 2022). For procedure descriptions, we summarize 10 templates per domain based on domain-specific documents such as process manuals and recipe instructions(Bień et al., 2020; Niazi, 2019). Details are shown in Appendix A.3 Structured sentence fragments are first generated via template filling. These fragments are then aggregated and refined using GPT-4(Achiam et al., 2023) to produce coherent and fluent paragraphs.

Our final dataset consists of 240 scheduling instances with task graphs, optimal schedule and problem description, covering four real-world scenarios and six problem size (5-10 tasks). We also conduct both automatic and human evaluations and include statistical analysis to ensure its high quality, shown in Appendix A.4 and A.5.

4 Experiment

We conduct systematic experiments to answer three research questions raised in Sec. 1. For RQ1, we introduce three metrics to evaluate nine state-of-the-art LLMs in `NL` $\Rightarrow$ `Schedule`. For RQ2, we conduct in-depth analysis of the failure modes and potential, showing their limitations in procedure grounding and strengths in global planning. For RQ3, we propose a multi-agent framework called MANS and compare with 6 LLM+solver methods, revealing comparable performance and more stable runtime.

4.1 Evaluating LLMs on `NL` $\Rightarrow$ `Schedule` (RQ1)

4.1.1 Model Choice

We evaluate nine state-of-the-art LLMs across two key dimensions: accessibility (open-source vs. closed-source) and reasoning capability (non-reasoning vs. reasoning). Non-reasoning models include: Open-source: Qwen3-235B-A22B-nothinking (Yang et al., 2025), Llama-4-Maverick (Meta AI Llama Team, 2025), DeepSeek-V3 (Liu et al., 2024a); Closed-source: GPT-4.1 (OpenAI Team, 2025a), Gemini-2.5-Flash-nothinking (Comanici et al., 2025). Reasoning models include: Open-source: Qwen3-235B-A22B (Yang et al., 2025), DeepSeek-R1 (Guo et al., 2025); Closed-source: o4-mini (OpenAI Team, 2025b), Gemini-2.5-Pro (Comanici et al., 2025).

4.1.2 Evaluation Metrics

To evaluate the completeness and feasibility of schedules generated by LLMs, we introduce following three metrics.

- **Overall Steps Execution Rate (SER)** The

Table 2: Model performance under different task combination settings in NL ⇒ Schedule. We report **Overall Steps Execution Rate (SER)**, **Average Task Completion Rate (TCR)**, **Schedule Feasibility Rate (SFR)**. #Task=n represents that the scheduling instance contains n tasks. Higher values indicate better performances. The best results are **bolded**, and the second best ones are underlined. The dashed line separates no-thinking models (above) from thinking models (below).

Model	#Task=5			#Task=6			#Task=7		
	SER	**TCR**	**SFR**	**SER**	**TCR**	**SFR**	**SER**	**TCR**	**SFR**
Qwen3-235B-A22B-nothinking	0.298	0.215	-	0.299	0.221	-	0.188	0.082	-
Llama4-Maverick	0.350	0.230	0.025	0.389	0.254	-	0.324	0.214	-
DeepSeek-V3	0.383	0.280	0.050	0.354	0.271	0.050	0.300	0.204	0.025
GPT-4.1	0.422	0.315	0.050	0.499	0.400	0.225	0.374	0.260	0.050
Gemini-2.5-Flash-nothinking	0.547	0.410	0.200	0.454	0.313	0.1	0.479	0.350	0.100
Qwen3-235B-A22B-thinking	0.279	0.195	-	0.243	0.158	-	0.167	0.082	-
DeepSeek-R1	0.612	0.455	0.250	0.594	0.450	0.175	0.505	0.343	0.125
o4-mini	0.614	**0.495**	**0.325**	0.625	**0.475**	0.250	0.532	0.368	0.225
Gemini-2.5-Pro	**0.634**	0.490	**0.325**	**0.653**	0.463	**0.275**	**0.596**	**0.425**	**0.250**
Model	**#Task=8**			**#Task=9**			**#Task=10**		
	SER	**TCR**	**SFR**	**SER**	**TCR**	**SFR**	**SER**	**TCR**	**SFR**
Qwen3-235B-A22B-nothinking	0.158	0.106	-	0.153	0.097	-	0.118	0.070	-
Llama4-Maverick	0.272	0.166	-	0.299	0.192	-	0.182	0.088	0.025
DeepSeek-V3	0.307	0.209	-	0.224	0.144	-	0.187	0.118	-
GPT-4.1	0.312	0.225	0.025	0.288	0.156	-	0.311	0.193	0.025
Gemini-2.5-Flash-nothinking	0.434	0.309	0.125	0.361	0.233	0.050	0.322	0.220	0.075
Qwen3-235B-A22B-thinking	0.186	0.116	-	0.173	0.106	-	0.106	0.050	-
DeepSeek-R1	0.558	0.416	0.125	0.527	0.353	0.100	0.528	0.343	0.050
o4-mini	0.637	0.491	**0.250**	0.524	0.414	**0.200**	0.515	0.385	0.125
Gemini-2.5-Pro	**0.667**	**0.525**	0.200	**0.597**	**0.419**	**0.200**	**0.601**	**0.425**	**0.175**

percentage of correctly executed steps across all tasks in all schedules:

$$\mathrm{SER} = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \frac{1}{|\mathcal{T}_s|} \sum_{t \in \mathcal{T}_s} \frac{|\mathcal{O}_j^c|}{|\mathcal{O}_j|} \tag{1}$$

where $\mathcal{S}$ denotes the set of all generated schedules, and $s \in \mathcal{S}$ denotes a schedule in the schedules set, $\mathcal{T}_s$ denotes the set of all tasks in s, $t \in \mathcal{T}_s$ denotes a task in task sets, $\mathcal{O}_j$ denotes the set of all steps in t, and $\mathcal{O}_j^c \subseteq \mathcal{O}_j$ denotes the subset of tasks in j that are executed successfully.

- **Average Task Completion Rate (TCR)** The percentage of successfully completed tasks across all schedules, even when full feasibility of the schedule isn't achieved:

$$\mathrm{TCR} = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \frac{|\mathcal{T}_s^c|}{|\mathcal{T}_s|} \tag{2}$$

where $\mathcal{T}_s^c \subseteq \mathcal{T}_s$ denotes the subset of tasks in s that are completed.

- **Schedule Feasibility Rate (SFR)** The percentage of generated schedules that are fully executable:

$$\mathrm{SFR} = \frac{|\mathcal{S}_f|}{|\mathcal{S}|} \tag{3}$$

where $\mathcal{S}_\mathrm{f} \subseteq \mathcal{S}$ denotes the feasible schedules subset of all schedules $\mathcal{S}$.

4.1.3 Main Results

As shown in Table 2, our findings reveal a core contradiction: LLMs excel at semantic comprehension but are deficient in procedural grounding and logical reasoning. A clear performance hierarchy emerges, with top-tier models exhibiting superior inductive reasoning. However, the universally low SFR, rarely exceeding 0.3 even for these models, pinpoints the primary bottleneck: generating a logically coherent and conflict-free schedule. Performance also consistently degrades with increasing problem scale, highlighting a critical weakness in handling combinatorial complexity. Furthermore, we observe that chain-of-thought reasoning can be detrimental for weaker models.

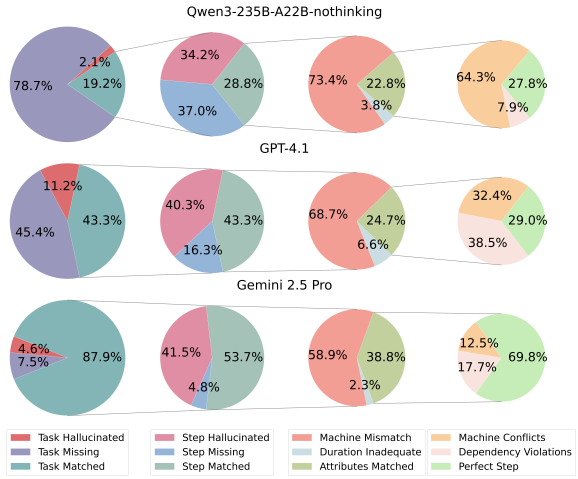

Figure 3: The error distribution across schedule, task, step, and constraint level.

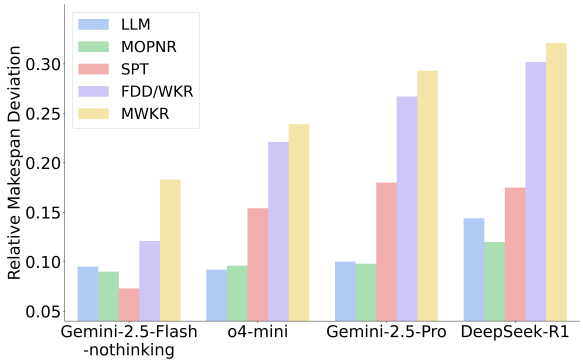

Figure 4: The makespan deviation generated schedules compared with PDRs. Lower value means better efficiency.

4.2 Analysis of Failure and Potential (RQ2)

4.2.1 Failure Analysis

To systematically diagnose model failures, we conduct a four-level analysis (schedule, task, step, and constraint), inspired by (Feitelson et al., 1997), with detailed category definitions in Appendix B.1. The results for three representative models are shown in Figure 3 and reveal a clear hierarchy of errors. Weaker models like Qwen3 fail at the highest level, with severe task omission in 78.7% of schedules. In contrast, top-tier models like Gemini 2.5 Pro are robust in identifying tasks (87.9% accuracy) but still struggle with finer details, frequently generating incomplete step sequences and mismatching machines. Constraint adherence is the ultimate bottleneck; while Gemini 2.5 Pro correctly handles nearly 70% of steps, weaker models are plagued by resource conflicts and dependency violations. In summary, robust procedural grounding remains a major hurdle for all models.

4.2.2 Potential Analysis

To evaluate the efficiency of feasible schedules, we introduce the Average Relative Makespan Deviation (RMD) metric. This approach isolates the assessment of schedule quality from feasibility failures, providing a clear measure of the model's potential to minimize makespan independently of its ability to satisfy hard constraints. It is defined as:

$$\text{RMD} = \frac{1}{|\mathcal{S}_f|} \sum_{s \in \mathcal{S}_f} \frac{M_s - M_s^g}{M_s^g} \tag{4}$$

where M_s denotes the makespan of generated schedule s, M_s^g denotes the makespan of ground truth schedule.

As shown in Figure 4, we found that schedules generated by LLMs are not only valid but highly efficient, achieving an RMD comparable to the best-performing Priority Dispatching Rule (PDR) baselines. This suggests LLMs adopt a holistic, adaptive strategy rather than a single simple rule.

We conduct a deeper analysis of the reasoning tokens from DeepSeek-R1 and thought summary from Gemini-2.5-Pro. As shown in B.2, their success is not based on a single heuristic but on a sophisticated, human-like workflow that dynamically integrates a portfolio of advanced strategies.

We found consistent evidence of three key cognitive abilities: 1) Global Planning, where models identify critical bottlenecks to guide the overall schedule, a hallmark of human experts; 2) Local Optimization, applying greedy heuristics like the Earliest Start Time rule to make efficient step-by-step decisions; and 3) Self-Correction, where they actively detect conflicts and use backtracking to revise the schedule. Furthermore, models like Gemini-2.5-Pro demonstrated even more complex reasoning, such as formulating parallel execution and applying the Critical Path Method (CPM).

In summary, these cognitive traces show that advanced LLMs construct a global plan by synthesizing the results of distinct local analysis. This ability to integrate multiple strategies is the key to their impressive performance on this problem.

4.3 Mitigating LLM Limitations with the MANS Framework (RQ3)

4.3.1 Overall Performance

We designed MANS, a framework that decomposes `NL` $\Rightarrow$ `Schedule` into a structured workflow of specialized agents as shown in 5. This architecture is engineered to mitigate LLMs limitations

Table 3: MANS and LLM+solver performace on NL ⇒ Schedule. We report **Overall Step Execution Rate (SER)**, **Average Task Completion Rate (TCR)** and **Schedule Feasibility Rate (SFR)**. #Task=n represents that the scheduling instance contains n tasks. Higher values indicate better performances. The best results are **bolded**, and the second best ones are underlined.

Method	#Task=5			#Task=6			#Task=7		
	SER	TCR	SFR	SER	TCR	SFR	SER	TCR	SFR
CoE + MOPNR	0.691	0.555	0.375	0.665	0.513	**0.350**	**0.626**	0.521	0.250
CoE + OR Tools	0.675	0.535	0.350	0.665	0.508	0.300	0.614	**0.539**	0.250
CoE + FJSP DRL	0.449	0.305	-	0.404	0.267	-	0.432	0.254	-
Optimus + MOPNR	0.651	0.515	**0.400**	0.521	0.508	**0.350**	0.597	0.426	**0.275**
Optimus + OR Tools	0.602	0.525	**0.400**	0.493	0.489	**0.350**	0.547	0.416	**0.275**
Optimus + FJSP DRL	0.528	0.315	-	0.487	0.258	-	0.456	0.257	-
MANS (ours)	**0.702**	**0.560**	**0.400**	**0.690**	**0.521**	0.325	0.608	0.443	0.250
Method	**#Task=8**			**#Task=9**			**#Task=10**		
	SER	TCR	SFR	SER	TCR	SFR	SER	TCR	SFR
CoE + MOPNR	**0.701**	0.525	0.275	**0.674**	0.401	**0.225**	0.587	0.465	0.225
CoE + OR Tools	0.691	0.525	0.250	0.670	0.398	**0.225**	0.587	0.465	0.250
CoE + FJSP DRL	0.485	0.347	-	0.419	0.314	-	0.422	0.275	-
Optimus + MOPNR	0.687	0.531	**0.300**	0.532	0.394	0.175	0.575	0.458	0.225
Optimus + OR Tools	0.633	0.525	**0.300**	0.532	0.394	0.175	0.535	0.435	0.225
Optimus + FJSP DRL	0.517	0.369	-	0.363	0.239	-	0.468	0.263	-
MANS (ours)	0.697	**0.550**	0.275	0.582	**0.417**	**0.225**	**0.682**	**0.543**	**0.275**

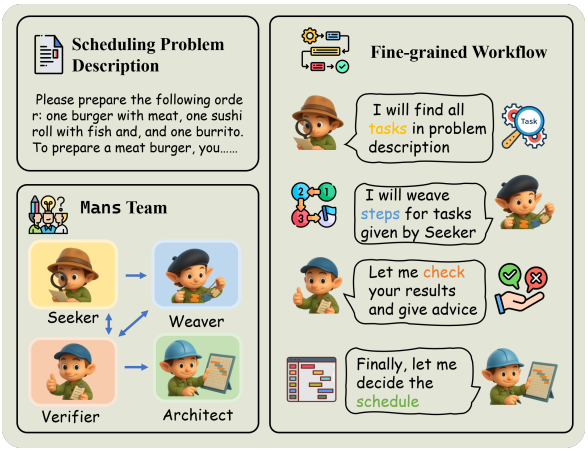

Figure 5: MANS decomposes NL ⇒ Schedule into four distinct stages. (a) Seeker extracts the full task set from text. (b) Weaver parses steps for each task. (c) Verifier checks the extracted structure for fidelity. (d) Architect schedules the verified tasks.

in procedures grounding and amplify the models global planning strengths. More details are shown in Appedix B.6. MANS consists of four agents: **Seeker**: Focuses solely on identifying the full task set, reducing cognitive load and ensuring a correct understanding of the problem scope. **Weaver**: Receives the task list from the Seeker and focuses on extracting each task's steps, machines, durations, and execution order. **Verifier**: Serves as a quality check, turning the framework into a self-correcting system that catches structural errors. **Architect**: This agent embeds planning instructions to guide the model, scaffolding its reasoning with strategies observed in successful traces.

While frameworks like Chain of Experts(CoE) (Xiao et al., 2024) and OptiMUS (AhmadiTeshnizi et al., 2024) introduce multi-agent collaboration, they operate under a "Pipeline Paradigm," using LLMs merely as translators to generate code for external solvers. This dependency creates fragility, where syntax errors lead to failure. MANS shifts the paradigm from "LLM-as-Translator" to "LLM-as-Solver." Instead of delegating computation, MANS internalizes algorithmic reasoning. This end-to-end approach eliminates reliance on external runtime environments. Detailed comparison is shown in AppendixB.4.

We compare MANS with CoE (Xiao et al., 2024) and Optimus(AhmadiTeshnizi et al., 2024). CoE and Optimus are two state-of-the-art methods for natural language optimization modeling. And we pair these methods with three distinct scheduling solvers: MOPNR (Holthaus and Rajendran, 1997), OR Tools(Google, 2023) and FJSP DRL(Song et al., 2023). So finally we get six LLM+solver baselines. We provide more details and the implementation of above methods in Appendix B.4. For a fair and efficient comparison, we adopt Gemini-2.5-Flash-nothinking as the LLM back-

Table 4: MANS and LLM+solver performace on **Worst-case Run Time(WRT)** and **Average Relative Makespan Deviation(RMD)**. Lower values indicate better performances.

Method	WRT(s)	RMD
CoE + MOPNR	128.42	0.192
CoE + OR Tools	154.44	0.023
CoE + FJSP DRL	-	-
Optimus + MOPNR	121.63	0.185
Optimus + OR Tools	133.70	**0.022**
Optimus + FJSP DRL	-	-
MANS (ours)	**79.59**	0.085

bone. We evaluate the completeness and feasibility of these methods, and the metrics is consistent with Sec.4.1.2. We also present experimental results with other models as the backbone in Appendix B.5. To further investigate the practical performance, we introduce RMD to evaluate the efficency of the generated feasible schedules and Worst-case Run Time(WRT) to evaluate stability in practical execution. The results in Table3 and 4 demonstrate MANS not only matches or exceeds the LLM+solver methods in generating complete and feasible schedules but also offers more stable performance profile. And we have following observations: 1) MANS achieves a WRT of 79.59s—40–50% faster than the best LLM+solver methods and achieves a strong average deviation of 0.085. This indicates that MANS maintains high-quality scheduling without sacrificing stability or speed.

2) Both CoE+FJSP DRL and Optimus+FJSP DRL fail to generate any feasible schedules, due to poor generalization of the DRL solver to unseen data in realistic data.

3) LLM+solver methods suffer cascading errors. We noticed that 45% of the cases that LLMs generate incorrect code related to the solver, which requires iterative correction based on code execution errors. And in some cases, the code compiles successfully but ultimately fails to produce a valid solution due to other logic errors. This fragmented pipeline introduces multiple failure points that MANS, with its unified reasoning process, avoids entirely.

4) Much of the high WRT in LLM+solver setups stems from repeated cycles of code generation, execution failure, and debugging. Although solvers like OR-Tools perform well when invoked correctly, accessing that power depends on the brittle success of earlier stages.

Table 5: Ablation study of the MANS framework.'w/o' denotes the removal of the specified agent.

Method	#Task=5		
	SER	TCR	SFR
MANS	**0.702**	**0.560**	**0.400**
MANS w/o seeker	0.612	0.485	0.200
MANS w/o weaver	0.665	0.535	0.375
MANS w/o verifier	0.648	0.495	0.300
Method	**#Task=10**		
	SER	TCR	SFR
MANS	**0.682**	**0.543**	**0.275**
MANS w/o seeker	0.482	0.315	0.075
MANS w/o weaver	0.637	0.468	0.225
MANS w/o verifier	0.594	0.425	0.150

4.3.2 Ablation Study

To validate the contribution of each specialized agent within the MANS, we further conducted ablation studies on #Task=5 and #Task=10 in Table 10. The results demonstrate that each agent serves a critical and indispensable function. Notably, the removal of the Seeker causes the most catastrophic failure, particularly on complex 10-task instances. This confirms that establishing the correct problem scope at the outset is the most critical step in the entire workflow. The substantial performance drops from removing the Weaver and Verifier likewise validate their essential roles in parsing detailed content and ensuring structural fidelity, respectively.

5 Conclusion and Future Work

We benchmark NL $\Rightarrow$ Schedule task with a novel dataset constructed from real-world materials, a systematical evaluation paradigm, and a series of state-of-the-art baselines. Experimental results and in-depth analysis of nine LLMs in the NL $\Rightarrow$ Schedule reveal the limitations of different LLMs in procedure grounding and the strengths of advanced LLMs in global planning via local analysis. To mitigate the main challenge and strengthen the strengths in global planning, we design a novel multi-agent framework MANS to empower the LLMs ability in procedures grounding and amplifying the models emergent reasoning strength. The results suggests more robust performance can be achieved by MANS compared with LLM+solver baselines.

Future work will focus on three directions. First, incorporating authentic real-world data: we aim to integrate raw industrial logs via anonymization and data desensitization to bridge the simulation-

to-reality gap. Second, enhancing scenario complexity: extending NL ⇒ Schedule to include dynamic constraints like resource availability windows, sequence-dependent setup times, and non-sequential workflows. Finally, delving into the nature of LLMs' emergent "algorithmic thinking" to further unlock AI-driven optimization discovery.

6 limitations

While this work establishes a foundational benchmark for the NL ⇒ Schedule task, we acknowledge several limitations that offer avenues for future research.

First, our dataset is constructed through a semi-synthetic process. Due to the significant challenge of obtaining large-scale, real-world data—which is often proprietary and exists in unstructured formats—we opted for a data2text approach that composes scheduling instances from authentic materials. While this methodology ensures a high degree of realism and control, the ultimate goal for the community is to collect and curate truly organic datasets of natural language requests paired with their resultant schedules. We hope our work stimulates such data collection efforts, despite their inherent difficulty.

Second, as the first benchmark in this domain, we deliberately constrained the problem's complexity to establish a tractable baseline for evaluation. Our dataset does not yet encompass the full spectrum of real-world challenges, such as larger problem scales, stochastic durations, sequence-dependent setup times, or dynamic resource availability. Future iterations of the benchmark will be dedicated to systematically incorporating these more complex constraints to better reflect the intricacies of industrial scheduling.

Finally, the evaluation of schedule generation from natural language is still in its nascent stage, and a standardized set of metrics has yet to emerge. The metrics we proposed—focusing on completeness, feasibility, and efficiency—provide a solid foundation but may not capture all facets of a "good" schedule, such as robustness or fairness. The development of a more comprehensive and widely accepted evaluation protocol is a crucial next step for the field.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China (No.62576230 and No.U25B201508 and No.62272330);

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Joseph Adams, Egon Balas, and Daniel Zawack. 1988. The shifting bottleneck procedure for job shop scheduling. *Management science*, 34(3):391–401.

Ali AhmadiTeshnizi, Wenzhi Gao, and Madeleine Udell. 2024. Optimus: scalable optimization modeling with (mi)lp solvers and large language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

David Applegate and William Cook. 1991. A computational study of the job-shop scheduling problem. *ORSA Journal on computing*, 3(2):149–156.

Michał Bień, Michał Gilski, Martyna Maciejewska, Wojciech Taisner, Dawid Wisniewski, and Agnieszka Lawrynowicz. 2020. RecipeNLG: A cooking recipes dataset for semi-structured text generation. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 22–28, Dublin, Ireland. Association for Computational Linguistics.

Paul Bratley, Michael Florian, and Pierre Robillard. 1971. Scheduling with earliest start and due date constraints. *Naval Research Logistics Quarterly*, 18(4):511–519.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Richard Walter Conway. 1967. " theory of scheduling,". *Addison Wesley*.

Giacomo Da Col and Erich C Teppan. 2019. Industrial size job shop scheduling tackled by present day cp solvers. In *International Conference on Principles and Practice of Constraint Programming*, pages 144–160. Springer.

John Dagdelen, Alexander Dunn, Sanghoon Lee, Nicholas Walker, Andrew S Rosen, Gerbrand Ceder, Kristin A Persson, and Anubhav Jain. 2024. Structured information extraction from scientific text with large language models. *Nature communications*, 15(1):1418.

Weihong Du, Wenrui Liao, Hongru Liang, and Wenqiang Lei. 2024. PAGED: A benchmark for procedural graphs extraction from documents. In *Proceedings of the 62nd Annual Meeting of the Association*

for Computational Linguistics (Volume 1: Long Papers), pages 10829–10846, Bangkok, Thailand. Association for Computational Linguistics.

Juliette Faille, Albert Gatt, and Claire Gardent. 2021. Entity-based semantic adequacy for data-to-text generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1530–1540. Association for Computational Linguistics.

Dror G Feitelson, Larry Rudolph, Uwe Schwiegelshohn, Kenneth C Sevcik, and Parkson Wong. 1997. Theory and practice in parallel job scheduling. In *Workshop on job scheduling strategies for parallel processing*, pages 1–34. Springer.

Robert Bosch GmbH. 2022. *Automotive handbook*. John Wiley & Sons.

Google. 2023. OR-Tools. Version 9.9, https://developers.google.com/optimization.

Mikell P Groover. 2010. *Fundamentals of modern manufacturing: materials, processes, and systems*. John Wiley & Sons.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Oliver Holthaus and Chandrasekharan Rajendran. 1997. Efficient dispatching rules for scheduling in a job shop. *International Journal of Production Economics*, 48(1):87–105.

Chenyu Huang, Zhengyang Tang, Shixi Hu, Ruoqing Jiang, Xin Zheng, Dongdong Ge, Benyou Wang, and Zizhuo Wang. 2025. Orlm: A customizable framework in training large models for automated optimization modeling. *Operations Research*.

Paul Jaccard. 1901. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579.

Terry K Koo and Mae Y Li. 2016. A guideline of selecting and reporting intraclass correlation coefficients for reliability research. *Journal of chiropractic medicine*, 15(2):155–163.

S Lawrance. 1984. Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques. *GSIA, Carnegie Mellon University*.

Wenqiang Lei, Weixin Wang, Zhixin Ma, Tian Gan, Wei Lu, Min-Yen Kan, and Tat-Seng Chua. 2020. Re-examining the role of schema linking in text-to-SQL. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6943–6954, Online. Association for Computational Linguistics.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Chien-Liang Liu, Chun-Jan Tseng, and Po-Hao Weng. 2024b. Dynamic job-shop scheduling via graph attention networks and deep reinforcement learning. *IEEE Transactions on Industrial Informatics*, 20(6):8662–8672.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Meta AI Llama Team. 2025. Llama 4 Scout and Maverick: Mixture-of-Experts Multimodal Models. https://github.com/meta-llama/llama-models/blob/main/models/llama4/MODEL_CARD.md. Accessed: 2025-07-24.

Carolyn R Miller. 1979. A humanistic rationale for technical writing. *College English*, 40(6):610–617.

J Muth. 1963. Probabilistic learning combinations of local job-shop scheduling rules. *Industrial scheduling*.

Sarfaraz K Niazi. 2019. *Handbook of pharmaceutical manufacturing formulations: Volume two, uncompressed solid products*. CRC press.

OpenAI Team. 2025a. GPT-4.1. https://platform.openai.com/docs/models/gpt-4.1. Accessed: 2025-07-24.

OpenAI Team. 2025b. O4-mini. https://openai.com/index/introducing-o3-and-o4-mini/. Accessed: 2025-05-10.

Shrikant S Panwalkar and Wafik Iskander. 1977. A survey of scheduling rules. *Operations research*, 25(1):45–61.

Michael L. Pinedo. 2022. *Scheduling: Theory, Algorithms, and Systems*. Springer International Publishing, Cham.

Cheyenne Powell and Annalisa Riccardi. 2025. Generating textual explanations for scheduling systems leveraging the reasoning capabilities of large language models. *Journal of Intelligent Information Systems*, pages 1–51.

LKPJ Rdusseeun and P Kaufman. 1987. Clustering by means of medoids. In *Proceedings of the statistical data analysis based on the L1 norm conference, neuchatel, switzerland*, volume 31, page 28.

Maria SFO de C Ribeiro and 1 others. 2015. Comparing mixed & integer programming vs. constraint programming by solving job-shop scheduling problems. *Independent Journal of Management & Production*, 6(1):211–238.

Melanie Rubino, Nicolas Guenon des Mesnards, Uday Shah, Nanjiang Jiang, Weiqi Sun, and Konstantine Arkoudas. 2022. Cross-TOP: Zero-shot cross-schema task-oriented parsing. In *Proceedings of the Third Workshop on Deep Learning for Low-Resource Natural Language Processing*, pages 48–60, Hybrid. Association for Computational Linguistics.

Norman Sadeh, Katia Sycara, and Yalin Xiong. 1995. Backtracking techniques for the job shop scheduling constraint satisfaction problem. *Artificial intelligence*, 76(1-2):455–480.

Daniel Sarsur, Patrícia N Pena, and Ricardo HC Takahashi. 2024. Automatic translation of blocking flexible job shop scheduling problems to automata using the supervisory control theory. *Journal of Control, Automation and Electrical Systems*, 35(1):12–23.

Salama Shady, Toshiya Kaihara, Nobutada Fujii, and Daisuke Kokuryo. 2022. A novel feature selection for evolving compact dispatching rules using genetic programming for dynamic job shop scheduling. *International Journal of Production Research*, 60(13):4025–4048.

Patrick E Shrout and Joseph L Fleiss. 1979. Intraclass correlations: uses in assessing rater reliability. *Psychological bulletin*, 86(2):420.

Nanua Singh. 1996. Systems approach to computer-integrated design and manufacturing.

Wen Song, Xinyang Chen, Qiqiang Li, and Zhiguang Cao. 2022. Flexible job-shop scheduling via graph neural network and deep reinforcement learning. *IEEE Transactions on Industrial Informatics*, 19(2):1600–1610.

Wen Song, Xinyang Chen, Qiqiang Li, and Zhiguang Cao. 2023. Flexible job shop scheduling via graph neural network and deep reinforcement learning. *IEEE Transactions on Industrial Informatics*, 19(2):1600–1610.

Eric Taillard. 1993. Benchmarks for basic scheduling problems. *european journal of operational research*, 64(2):278–285.

TemplateLab. 2025. 18 free purchase order templates in word, excel, pdf. https://templatelab.com/purchase-order-templates/. Accessed: 2025-07-28.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems*, 36:75993–76005.

Ziyang Xiao, Dongxiang Zhang, Yangjun Wu, Lilin Xu, Yuan Jessica Wang, Xiongwei Han, Xiaojin Fu, Tao Zhong, Jia Zeng, Mingli Song, and 1 others. 2024. Chain-of-experts: When llms meet complex operations research problems. In *ICLR*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Cong Zhang, Wen Song, Zhiguang Cao, Jie Zhang, Puay Siew Tan, and Xu Chi. 2020. Learning to dispatch for job shop scheduling via deep reinforcement learning. *Advances in neural information processing systems*, 33:1621–1632.

A Details of Dataset

A.1 Dataset Source

To construct a benchmark that reflects the complexity and nuance of real-world scheduling problems, we adopted a synthetic, multi-source data collection methodology. A single source, such as a handbook or a website, rarely provides a complete, self-contained description of a complex task with all its requisite stepal details, resource constraints, and durations. Therefore, instead of relying on individual sources, we composed each complete task description by synthesizing and cross-referencing information from multiple, varied sources. For example, the fundamental process flow and precedence constraints for a manufacturing task might be derived from an engineering handbook, while the specific machine names and realistic processing times could be informed by industry websites or technical documentation. This synthetic approach allowed us to create rich, detailed, and coherent task graphs that are more representative of real-world industrial challenges than any single source could provide alone. The primary sources that informed this synthesis process are detailed in Table 6.

A.2 Details of Scheduling Instance Composition

We compose complete schedule instances by combining multiple task graphs. A simple random combination would fail to generate challenging or realistic scenarios. The core difficulty in scheduling arises from resource contention, where multiple tasks compete for the shared, limited machines.

To ensure that the scheduling instances in NL $\Rightarrow$ Schedule are both realistic and computationally challenging, we developed a principled, two-stage process for composing instances from our pool of individual task graphs. This process is designed to maximize resource contention, which is the primary driver of scheduling complexity.

A.2.1 Machine Overlap Clustering

First, we sought to identify natural workshop scenarios where a specific set of machines is frequently used together. This was achieved through a formal clustering process based on machine usage patterns.

- **Vector Representation**: Each unique task graph in our pool was represented as a binary vector $\mathbf{v} \in \{0,1\}^{|M|}$, where $|M|$ is the total number of unique machines across all tasks in a scenario. An element v_i is 1 if the task graph requires machine M_i , and 0 otherwise.

- **Similarity Metric**: We used the Jaccard Distance(Jaccard, 1901) to measure the similarity between any two task vectors, $\mathbf{v}_a$ and $\mathbf{v}_b$. The Jaccard Distance is well-suited for this purpose as it measures the similarity of sparse sets:

$$J(\mathbf{v}_a, \mathbf{v}_b) = \frac{|\mathbf{v}_a \cap \mathbf{v}_b|}{|\mathbf{v}_a \cup \mathbf{v}_b|} = \frac{\sum_i \min(v_{a_i}, v_{b_i})}{\sum_i \max(v_{a_i}, v_{b_i})}$$

  A higher Jaccard Distance indicates that two tasks share a larger proportion of their required machines.

- **Clustering Algorithm**: Using the pairwise Jaccard similarity matrix, we applied K-Medoids Clustering(Rdusseeun and Kaufman, 1987) to group the task graphs from each scenario. Each resulting cluster represents a workshop where the constituent tasks heavily compete for a common set of machines. Finally we get 13 workshops that listed in Table 7

A.2.2 Resource-Centric Instance Composition

With the workshop clusters defined, we generate each final scheduling instance using a deterministic, cluster-based composition rule:

- **Select a Core Cluster**: To generate a new scheduling instance of size n, we first select one of the 13 workshop clusters to serve as the core workshop.

- **Intra-Cluster Task Sampling**: We then sample n unique task graphs exclusively from within this selected core cluster.

This deterministic, intra-cluster sampling strategy guarantees that every generated instance is composed of tasks that have a high degree of machine overlap. This, in turn, ensures high resource contention, which is the primary driver of scheduling complexity. By preventing the inclusion of tasks from outside the cluster, we create focused, non-trivial scheduling problems that rigorously test a model's ability to resolve conflicts and manage shared resources effectively. This principled composition method is critical to the quality and challenge of the NL $\Rightarrow$ Schedule dataset.

Table 6: Details of the public data sources used for constructing the NL ⇒ Schedule benchmark.

Source Name	Source Type	Reference / Link
Scheduling: Theory, Algorithms, and Systems	Book	(Pinedo, 2022)
Systems approach to computer-integrated design and manufacturing	Book	(Singh, 1996)
Fundamentals of modern manufacturing: materials, processes, and systems	Book	(Groover, 2010)
Automotive handbook	Book	(GmbH, 2022)
SAE Books	Website	https://www.sae.org/publications/books
DMG MORI	Website	https://en.dmgmori.com/news-and-media/news/dmg-mori-technology-excellence-automotive
TOYOTA VIRTUAL PLANT TOUR	Website	https://global.toyota/en/company/plant-tours/stamping
Overcooked! 2	Video Game	https://store.steampowered.com/app/728880/Overcooked_2/
Big Pharma	Video Game	https://store.steampowered.com/app/344850/Big_Pharma/
Construction Simulator	Video Game	https://store.steampowered.com/app/1273400/Construction_Simulator/

Table 7: Details of the public data sources used for constructing the NL ⇒ Schedule benchmark.

Scenario	Number	Worshop
Manufacturing	7	Brake_Shop, Chassis_Shop, Stamping_Shop, Engine_Shop, Cooling_Shop, Electrical_Shop, Transmission_Shop
Construction	4	Residential_Community, Campus, Commercial_Complex, Industrial_Park
Pharmaceuticals	1	-
Cooking	1	-

A.3 Dataset Template

To generate the realistic natural language scheduling problem descriptions for NL ⇒ Schedule, we developed a template-based approach to ensure that the resulting texts exhibit both structural consistency and linguistic diversity, mirroring the style of real-world documents.

We curated two distinct sets of templates by first collecting and then summarizing real-world data.

A.3.1 Scheduling Requirement Templates

To create realistic high-level problem descriptions, we began by scraping a corpus of authentic order and dialogue data from public websites(TemplateLab, 2025) and datasets(Rubino et al., 2022). From this raw data, we analyzed common phrasing, request structures, and objective statements. We then synthesized these recurring patterns into 20 distinct scheduling requirement templates.

A.3.2 Procedure Description Templates

Similarly, to ensure the procedural instructions for each task were realistic, we scraped a diverse set of public procedural documents from manufacturing, construction, pharmaceuticals, and cooking domains (Bień et al., 2020; Niazi, 2019).By analyzing these texts, we identified common linguistic patterns for describing sequential steps, machine usage, and processing times. These patterns were then distilled into 10 distinct procedure description templates per domain.

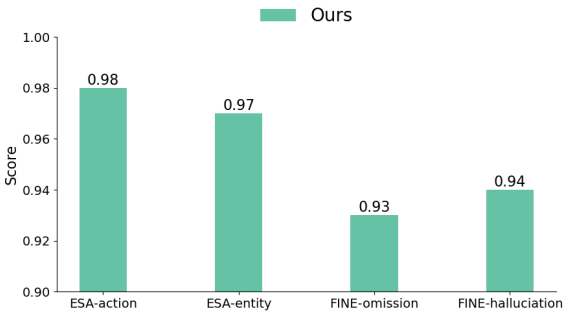

Figure 6: Automatic Evaluation results of ours dataset.

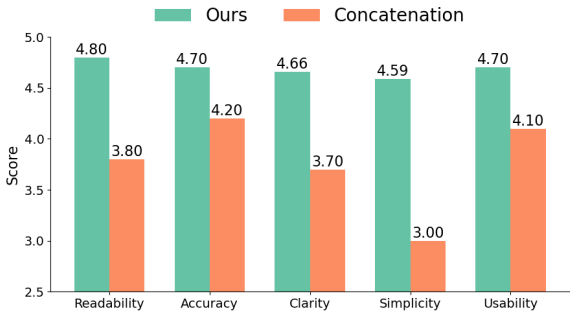

Figure 7: Human Evaluation results of the variant and ours dataset.

A.4 Dataset Analysis

We analyze our datasets to investigate whether the generated order texts are consistent with the original scheduling instances, whether the texts are qualified according to human standards Therefore, we conduct both automatic and human evaluations to the dataset constructed by our three-stage pipeline.

A.4.1 Automatic Evaluation

To evaluate the generated documents, we employ 2 standard Data2Text metrics, **FINE** and **ESA** (Faille et al., 2021). FINE uses a natural language inference model(Liu et al., 2019) to check for factual inconsistencies, specifically identifying hallucinations and omissions. ESA, in turn, quantifies the coverage of key entities and actions. For both metrics, higher values denote superior quality. The central goal of applying these metrics is to quantitatively verify the semantic equivalence between our ground-truth task graphs and the final scheduling problem descriptions. This automated evaluation provides a rigorous, objective measure of how effectively our generation pipeline preserves the informational integrity of the source data.

The results presented in Figure 6 provide strong quantitative evidence for the high quality and faithfulness of our generated documents. Specifically, the exceptionally high scores on the FINE metric confirm that our generation process is factually grounded, exhibiting a minimal rate of both hallucinations and omissions. This is further corroborated by the excellent performance on the ESA metric, which demonstrates that nearly all key entities and actions from the source task graphs are successfully preserved in the final text.

A.4.2 Human Evaluation

we created a variant of documents for comparison: concatenation, which directly concatenates all template-filling sentences as the final description. No LLM-based polishing was performed. A central goal of this evaluation is to validate the necessity and effectiveness of our generation&refinement process. Inspired by (Miller, 1979), we ask three workers to score the document from 1 to 5 on five criteria: readability, the quality of the document to be understood easily; accuracy, the quality of the document accurately describing the information in the procedure graph; clarity, the quality of the document expressing the complex execution of actions logically; simplicity, the quality of the document not containing redundant information; usability, the quality of the document aiding users in accomplishing this procedure. To ensure the reliability of human evaluation, we hire three domain experts to evaluate the generated documents and calculate the ICC (Intraclass Correlation Coefficient) (Shrout and Fleiss, 1979) score between different experts. The higher ICC score indicates higher consistency between different experts and higher reliability of the evaluation. Generally an ICC of 0.75 or higher indicates that the evaluation is reliable (Koo and Li, 2016). ICC is calculated as follows:

$$\mathrm{ICC} = \frac{\mathrm{MS}_{\text{between}} - \mathrm{MS}_{\text{within}}}{\mathrm{MS}_{\text{between}} + (k-1) \times \mathrm{MS}_{\text{within}}}$$

where $\mathrm{MS}_{\text{between}}$ is the mean square for between groups variability, $\mathrm{MS}_{\text{within}}$ is the mean square for within groups variability and k is the number of groups. And we ensure that the ICC scores for all of our evaluations are ≥ 0.75.

The results is shown in 7, it confirms the high quality of our generated dataset and validate our generation&refinement process. The concatenation scored exceptionally well in Accuracy and Usability, indicating that our underlying template-filling method successfully preserves the factual integrity and core utility of the source data. Furthermore, ours data demonstrated a significant improvement across all five metrics, with particularly notable

gains in Readability, Clarity, and Simplicity. This proves that the LLM refinement stage is crucial for transforming the factually correct but potentially rigid text into a coherent, natural, and easily digestible narrative.

A.5 Statistical Analysis

The statistical results in Table 8 demonstrate that the dataset is highly diverse, challenging, and representative of real-world industrial scenarios from three key perspectives:

- **Linguistic Variability**: While intra-domain embedding similarity (EmbSim) remains relatively high (0.60–0.75) due to the necessary use of domain-specific technical terminology (e.g., "Excavator" in Construction), the overall global similarity drops significantly to 0.4150. This sharp contrast indicates that the dataset covers a broad semantic spectrum across the four workshops. This low inter-domain similarity ensures that models cannot rely on memorizing simple linguistic templates and must generalize across vastly different lexical and contextual environments.

- **Task Complexity**: The dataset features varied task structures across domains. The Construction workshop represents "Deep Logic" scenarios with the highest number of steps (8.67±1.48) and machine diversity (8.34), requiring models to handle long-chain temporal dependencies. Conversely, the Pharmaceuticals workshop represents high-frequency, short-cycle tasks. The high standard deviation in workloads and steps (e.g., Cooking range: 3–12 steps) further ensures that the scheduling algorithms are tested against highly heterogeneous task distributions.

- **Inter-task Resource Constraints**: The difficulty of the scheduling problem is evidenced by intense resource competition. The *Cooking* domain reaches a **Max Resource Load of 1.00**, indicating a zero-idle bottleneck where resources are utilized to their theoretical limit, leaving no room for scheduling errors. The **Manufacturing** domain exhibits the highest *Task Overlap Ratio* (80.49%) and *Sharing Degree* (5.98 tasks per machine), forcing the scheduler to resolve severe conflicts among nearly identical resource requirements across multiple tasks.

These metrics collectively demonstrate that the dataset provides a rigorous benchmark for evaluating a model's ability to perform complex combinatorial reasoning under strict resource limitations.

B Details of Experiments

B.1 Details of Failure Analysis

B.1.1 Definition of Error Category

To systematically diagnose model failures, we conduct a four-level analysis inspired by (Feitelson et al., 1997) aligned with the hierarchical structure of a schedule: schedule level, task level, step level, and constraint level. At the schedule level, we evaluate whether the set of tasks included in each generated schedule matches the ground truth. Each case is categorized as *task matched* (all required tasks are present), *task missing* (some tasks are omitted), or *task hallucinated* (irrelevant or spurious tasks are introduced). At the task level, we examine each task in the task matched set to assess whether its full sequence of steps is correctly specified. A task is labeled as *step matched*, *step missing*, or *step hallucinated* similar with schedule level. Only tasks with fully matched steps are passed to the next level. At the step level, we check whether each step has the correct attribute assignments. Steps are marked as *attribute matched*, *machine mismatch* (assigned to the wrong machine), or *duration inadequate* (assigned a shorter-than-required duration). Finally, at the constraint level, we analyze steps with correct attributes for violations of fundamental scheduling rules, including *dependency violations* (incorrect step order) and *machine conflicts* (overlapping steps on the same machine). A step that passes all four levels without error is counted as a *perfect step*.

B.1.2 Full results about failure analysis

We list the other six models failure modes distribution in 8.

We can draw similar conclusion with 4.2.1. We observe the same consistent patterns and performance hierarchies:

- **Macro-Level Fidelity**: The trend holds true that non-reasoning models consistently struggle to identify the correct set of tasks, exhibiting high rates of both tasks missing and tasks hallucinated. Reasoning models mirror the strong performance of Gemini 2.5 Pro, demonstrating a significantly more robust ability to correctly grasp the overall problem scope.

Table 8: Statistical Analysis of the Dataset: Linguistic Variability, Task Complexity, and Inter-task Resource Constraints. **EmbSim** measures semantic closeness between instances (lower values indicate higher linguistic diversity); **Steps (Avg)** represents the mean number of sequential operations within a task; **Workload** denotes the total processing time required to complete a task; **Mach. Div.** indicates the variety of unique machine types required per task; **Sharing Deg.** reflects the average number of tasks competing for the same resource; **Max Load** (Bottleneck Index) is the ratio of the most utilized machine's workload to the total makespan; **Overlap %** represents the probability of two tasks requiring identical sets of machines.

Scenario	**Linguistic**	**Task Complexity (per Task)**			**Inter-task Resource Constraints**		
	EmbSim ↓	**Steps (Avg)**	**Workload**	**Mach. Div.**	**Sharing Deg.**	**Max Load**	**Overlap %**
Construction	0.7559	8.67 ± 1.48	51.36 ± 11.63	8.34	4.02 (Max 10)	0.77	54.98%
Cooking	0.7326	7.19 ± 2.03	24.85 ± 9.37	3.16	2.98 (Max 10)	1.00	48.36%
Manufacturing	0.6056	5.05 ± 0.22	23.30 ± 7.77	4.28	5.98 (Max 10)	0.96	80.49%
Pharmaceuticals	0.6819	4.17 ± 1.06	7.19 ± 2.81	4.14	4.65 (Max 10)	0.93	64.06%
All Combined	**0.4150**	6.27 ± 2.31	26.68 ± 17.52	4.98	4.41 (Avg)	0.915 (Avg)	61.97%

- **Micro-Level Content**: The challenge of parsing fine-grained details is universal. For nearly every model evaluated, machine mismatch is confirmed as the single most dominant attribute error, often accounting for the majority of all micro-level failures.
- **Scheduling Logic**: The models' underlying reasoning strategies are also consistent. non-reasoning models are almost exclusively defeated by global resource conflicts. Reasoning models, however, consistently demonstrate a more balanced and effective capability.

B.2 Details of Potential Analysis

Several key snippet from reasoning tokens from DeepSeek-R1 and thought summary from Gemini-2.5-Pro:

- *"We note that the RA [Robot Arm] is heavily used. We must sequence the RA steps appropriately."*

  This statement shows the model identifying the Robot Arm as a bottleneck—a strategy typical of expert human schedulers (Adams et al., 1988). It suggests the model adopts a global plan that prioritizes managing critical resources to guide downstream decisions.

- *"Task1 steps: step0: RA (1) → step1: CS (6) → (because after step0, we can start step1 at 1) step2: F (12) → (because after step1, we can start step2 at 7)"*

  This statement reveals a greedy use of the Earliest Start Time rule (Bratley et al., 1971), placing each step as early as possible.

- *"This is a conflict! We have two steps on CS at the same time... We must fix this."*

  This statement shows that LLM does not blindly apply rules; it simulates the outcome of its decisions and actively checks for constraint violations. When a potential conflict is detected, it triggers a backtracking search (Sadeh et al., 1995), returns to previous decision points and adjust the solution path.

B.3 Details of Priority Dispatching Rules

To provide strong, established baselines for our makespan deviation analysis, we implemented four widely-used Priority Dispatching Rules (PDRs). These rules are heuristics that make a local, greedy decision whenever a machine becomes free and has one or more steps waiting in its queue. Below are the detailed descriptions of each rule employed in our study.

B.3.1 Shortest Processing Time (SPT)

This rule prioritizes the step that can be completed the fastest. When a machine becomes available, it examines the queue of all steps waiting for it. It selects and processes the step with the minimum (shortest) processing duration. SPT is a classic heuristic known for its effectiveness in reducing average task flow time and work-in-process inventory. It is a highly localized, greedy strategy that aims to free up machines as quickly as possible. Its primary weakness is that it can "starve" steps with long processing times, potentially delaying the completion of certain tasks indefinitely if a continuous stream of shorter tasks is available.

B.3.2 Most steps Remaining (MOPNR)

This rule prioritizes the task that has the longest sequence of steps still to be completed. When a machine becomes available, it considers the parent tasks of all waiting steps. It selects the step

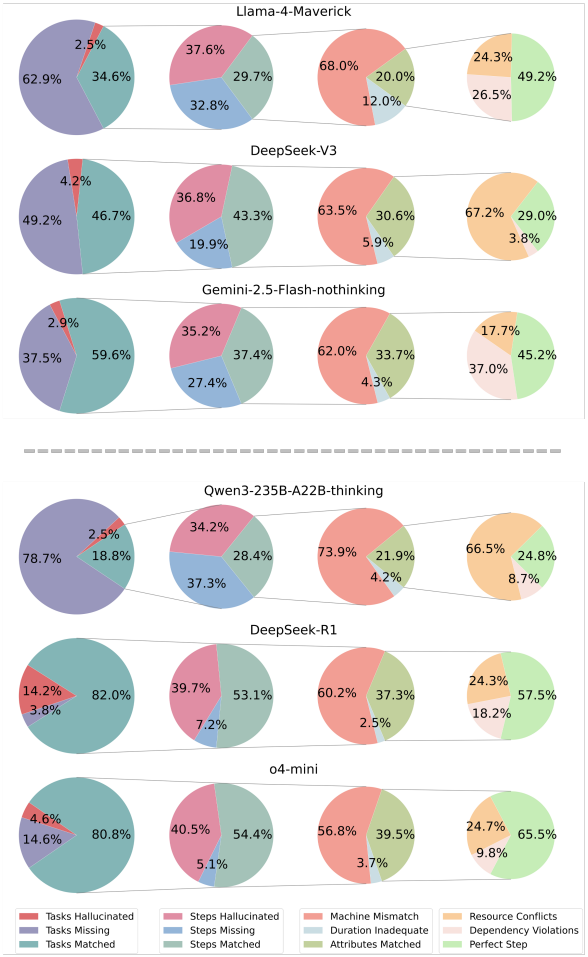

Figure 8: The failure analysis of the other six models. The dashed line separates non-reasoning models (above) from reasoning models (below).

belonging to the task that has the highest number of steps remaining in its process plan. MOPNR is a lookahead heuristic designed to prevent complex, multi-step tasks from being left until the end. By prioritizing tasks that are "further from completion," it attempts to ensure that all tasks progress through the system in a balanced manner. It focuses on the structural complexity of a task rather than the duration of its individual steps.

B.3.3 Most Work Remaining (MWKR)

This rule prioritizes the task that has the largest amount of total processing time left. Similar to MOPNR, this rule considers the parent tasks of waiting steps. For each task, it calculates the sum of the durations of all its remaining steps. It then selects the step belonging to the task with the highest total work remaining. MWKR is a more refined version of MOPNR. Instead of simply counting the number of future steps, it accounts for their duration, providing a more accurate measure of the "work content" left in a task. It is a robust global heuristic that effectively prioritizes larger, more demanding tasks.

B.3.4 First Due Date / Work Remaining (FDD/WKR)

This rule prioritizes tasks based on their urgency, using remaining work as a tie-breaker. When a machine becomes available, it selects the step belonging to the task with the earliest assigned due date. In the case where multiple waiting tasks share the same earliest due date, the tie is broken by applying the MWKR rule. In classic JSSP benchmarks where explicit due dates are not provided, a common practice is to estimate a pseudo due date for each task, often calculated as a multiple of its total processing time. Our implementation follows this approach to simulate an urgency-based scheduling environment. This rule therefore tests the ability to schedule based on a measure of overall task completion targets.

B.4 Details of LLM+solver Implementation

To create robust and fair comparisons against the mediated solving paradigm, we adapted two state-of-the-art frameworks, Chain-of-Experts (CoE) (Xiao et al., 2024) and Optimus (AhmadiTeshnizi et al., 2024), to the `NL` ⇒ `Schedule` task. Our adaptation philosophy was to preserve the core collaborative intelligence of these frameworks while ensuring a fair informational basis for comparison against our MANS.

B.4.1 Adaptation of CoE

The CoE framework utilizes a team of specialized agents to collaboratively solve optimization problems. To adapt it for our task under fair conditions, we made one primary modification to its agent roster:

We deliberately removed the Terminology Interpreter, Modeling Knowledge Supplement Expert, LP File Generation and the Programming Example Provider from the agent team. In the original framework, these agents leverage external knowledge bases (e.g., GAMS modeling examples, Gurobi code snippets) to provide expert-level shortcuts and pre-canned solutions. Removing them was crucial for fairness, as it ensures that the CoE baseline, like MANS, must derive its solution based solely on the provided natural language text and its general world knowledge, rather than relying on an external library of optimization tricks or code examples. And we modify the Modeling Expert and

Framework	Iterative Collaboration	Structured Context	Solver Independence	Intrinsic Reasoning
ORLM (Huang et al., 2025)	✗	✗	✗	✗
CoE (Xiao et al., 2024)	✓	✗	✗	✗
OptiMUS (AhmadiTeshnizi et al., 2024)	✓	✓	✗	✗
MANS (Ours)	✓	✓	✓	✓

Table 9: Comparison of MANS with state-of-the-art frameworks.

Programming Expert since it provides Gurobi Reference knowledge, which is not used in our solver.

All other agents were retained with slight modifications to preserve the framework's core collaborative reasoning capabilities.

B.4.2 Adaptation of Optimus

The Optimus framework employs a highly structured, modular workflow that strictly separates the mathematical formulation of a problem from its final implementation in code. Our primary adaptation focused on aligning its core formulation logic with the nature of the Job Shop Scheduling Problem.

The original Optimus is hard-coded to guide its Formulation Generation agent to produce models in Mixed-Integer Linear Programming form. Given that text-to-schedule is often more naturally and efficiently modeled using Constraint Programming(CP), we modified the core prompts for the Formulation Generation and Formulation Fixing agents. Instead of instructing them to produce linear equations, the modified prompts guide them to define the problem in terms of core CP concepts: decision variables (e.g., the start and end times of times), their domains, and the logical constraints that govern them (e.g., precedence and no-overlap resource constraints).

The core modular workflow of management, formulation, coding, and debugging was otherwise preserved with slight modifications.

B.4.3 Solver-Specific Adaptations for Programming Agents

After the problem has been understood and modeled by the upstream agents in both the CoE and Optimus frameworks, the final stage is to generate an executable program for a backend solver. We adapted the "programmer" agents within both frameworks to target our three distinct solver backends.

MOPNR is the highest-performing PDR in Sec. 4.2.2, a fast but myopic heuristic. OR Tools represents the industrial-strength "gold standard." It provides an upper bound on solution quality. FJSP DRL represents the AI-driven approach, which uses Deep Reinforcement Learning to train a scheduling policy.

This required creating three specialized versions of the final code generation logic.

- **Adaptation for Google OR-Tools** The programming agents' primary task was to generate a complete and syntactically correct python script that uses the `ortools.sat.python.cp_model` API[1].
- **Adaptation for MOPNR** Since MOPNR is a heuristic, not a solver that accepts a formal model, the programming agents' task was redefined. They were required to write a Python script that implements the MOPNR scheduling logic from scratch.
- **Adaptation for FJSP DRL** For the DRL backend, the agents' task was to act as a "tool-user" that prepares the correct input for a pre-trained model. This involved generating a python script that interfaces with the provided DRL library. The agents were provided with a curated API documentation page (derived from the original paper's repository[2]) and were instructed to write code to load the pre-trained DRL policy model and translate the structured task graph information into the precise JSON-based state representation required by the model's inference function.

Detailed prompt template implementations for CoE:

Modeling Expert:
You are a modeling expert specialized in the field of Job Shop Scheduling problem. Your expertise lies in Constraint Programming(CP) models, and you possess an in-depth under-

[1] https://developers.google.com/optimization/scheduling/job_shop
[2] https://github.com/songwenas12/fjsp-drl

Table 10: Ablation study of the MANS framework.'w/o' denotes the removal of the specified agent.

Method	#Task=5		
	SER	TCR	SFR
MANS	**0.702**	**0.560**	**0.400**
MANS w/o seeker	0.612	0.485	0.200
MANS w/o weaver	0.665	0.535	0.375
MANS w/o verifier	0.648	0.495	0.300
Method	**#Task=10**		
	SER	**TCR**	**SFR**
MANS	**0.682**	**0.543**	**0.275**
MANS w/o seeker	0.482	0.315	0.075
MANS w/o weaver	0.637	0.468	0.225
MANS w/o verifier	0.594	0.425	0.150

standing of various modeling techniques within the realm of operations research. At present, you are given an Job Shop Scheduling problem, alongside additional insights provided by other experts. The goal is to holistically incorporate these inputs and devise a comprehensive model that addresses the given production challenge. Now the origin problem is as follow: {problem}. And the modeling is as follow: {comments} Give your model of this

Programming Expert:

You are a Python programmer in the field of Job Shop Scheduling problem. Your proficiency in utilizing third-party libraries such as {solver} is essential. In addition to your expertise in {solver}, it would be great if you could also provide some background in related libraries or tools, like NumPy, SciPy, or. You are given a specific problem and comments by other experts. You aim to develop an efficient Python program that addresses the given problem. Now the origin problem is as follow: problem And the experts along with there comment are as follow: comments Give your Python code directly.

Detailed prompt template implementations for Optimus:

Formulation generation prompt

You are an expert mathematical formulator and an optimization professor at a top university. Your task is to model a specific constraint ({clausType}) of a Job Shop Scheduling Problem (JSSP) using a Constraint Programming (CP) formulation.

Here is a {clausType} we need you to model: {targetDescription}

Here is some context on the problem: {background}

Here is the list of available decision variables (e.g., step start/end times): {variables}

And finally, here is the list of input parameters (e.g., step durations): {parameters}

First, take a deep breath and explain how we should define the {clausType} for this JSSP. Feel free to define new decision variables if you think it is necessary.

Formulation generation prompt

You are a mathematical formulator working with a team of optimization experts. The objective is to tackle a complex Job Shop Scheduling problem, and your role is to fix a previously modelled {target}. Recall that the {target} you modelled was {constraint} and your formulation you provided was {formulation} The error message is {error} Here are the variables you have so far defined: {variables} Here are the parameters of the problem {parameters} Your task is carefully inspect the old {target} and fix it when you find it actually wrong. After fixing it modify the formulation. Please return the fixed JSON string for the formulation. The current JSON is {json} Take a deep breath and solve the problem step by step.

B.5 Backbone Comparison Experiment

B.6 Details of MANS

Below are the prompts of MANS.

Seeker prompt:

You are the Seeker, an expert in identifying high-level objectives from complex documents. Your sole task is to read the provided scheduling problem description and identify the complete and accurate set of all tasks that need to be completed.

The full order text is as follows: {scheduling_problem_description}

Your instructions are:

Table 11: Performance of different LLM backbones on NL ⇒ Schedule across varying problem sizes. We report **Overall Step Execution Rate (SER)**, **Average Task Completion Rate (TCR)**, and **Schedule Feasibility Rate (SFR)**. #Task=n represents that the scheduling instance contains n tasks. Higher values indicate better performances. The best results are **bolded**, and the second best ones are underlined.

Backbone	#Task=5			#Task=6			#Task=7		
	SER	**TCR**	**SFR**	**SER**	**TCR**	**SFR**	**SER**	**TCR**	**SFR**
Qwen3-235B-A22B-nothinking	0.346	0.232	-	0.299	0.247	-	0.234	0.129	-
Llama4-Maverick	0.532	0.324	0.050	0.574	0.336	0.025	0.428	0.274	-
DeepSeek-V3	0.597	0.336	0.050	0.534	0.369	0.025	0.400	0.324	0.075
GPT-4.1	0.638	0.423	0.200	**0.692**	**0.532**	0.250	0.596	0.378	0.125
Gemini-2.5-Flash-nothinking	**0.702**	**0.560**	**0.400**	0.690	0.521	**0.325**	**0.608**	**0.443**	**0.250**
Backbone	**#Task=8**			**#Task=9**			**#Task=10**		
	SER	**TCR**	**SFR**	**SER**	**TCR**	**SFR**	**SER**	**TCR**	**SFR**
Qwen3-235B-A22B-nothinking	0.177	0.131	-	0.142	0.117	-	0.121	0.096	-
Llama4-Maverick	0.412	0.305	-	0.296	0.184	-	0.209	0.097	-
DeepSeek-V3	0.399	0.294	-	0.334	0.221	-	0.216	0.163	-
GPT-4.1	0.588	0.515	0.075	0.426	0.314	0.025	0.352	0.275	0.050
Gemini-2.5-Flash-nothinking	**0.697**	**0.550**	**0.275**	**0.582**	**0.417**	**0.225**	**0.682**	**0.543**	**0.275**

Focus exclusively on identifying the tasks. Extract their names or unique identifiers as described in the text.

Handle quantities. If a task needs to be performed multiple times (e.g., "produce 3 widgets"), create a unique entry for each instance (e.g., widget_1, widget_2, widget_3).

Crucially, you must IGNORE all other details. Do not extract the steps, machines, or durations. That is not your responsibility. This focus is critical to ensure the overall scope of the problem is captured correctly.

Weaver prompt:

You are the Weaver, a specialist in parsing detailed procedural workflows from text. Your task is to take a list of pre-identified tasks and extract the precise sequence of steps required for each one.

The full order text for context is: {scheduling_problem_description}

The list of tasks you must process is: {task_list_from_seeker}

Your instructions are:

For every task in the provided list, find its description in the text and extract its sequence of steps.

For each step, you must extract its execution order (e.g., step 1, step 2), the specific machine it requires, and its processing duration.

Verifier prompt:

You are the Verifier, a meticulous quality assurance expert. Your task is to perform a rigorous audit of the structured data extracted from a problem description, ensuring it is 100% faithful to the original text.

The original, ground-truth text is: scheduling_problem_description

The extracted task and step structure you must verify is: {task_list_from_seeker} {structured_data_from_weaver}

Your instructions are to perform a complete check and then output a corrected version:

Verify task Completeness: Does the number of tasks in the structured data exactly match the number of tasks described in the text?

Verify step Completeness: For each task, does the number of steps in the structured data exactly match what is described in the text?

Verify Attribute Accuracy: For every single step, cross-reference the text to confirm that the assigned machine and duration are completely accurate.

Correct any discrepancies.

Architect prompt:

You are the Architect, an expert scheduling strategist. Your task is to take a perfectly structured and verified problem definition and create an optimal, conflict-free schedule that com-

pletes all tasks in the shortest possible time (minimizes makespan).

The verified problem structure is: verified_task_graphs

To create your schedule, you must follow the Scheduling Strategy Guide below, using it to inform your step-by-step deliberation process.

Scheduling Strategy Guide:

Identify Bottlenecks: Before scheduling, analyze all steps to identify the most heavily contended machines. The efficient sequencing of tasks on these bottleneck resources is critical to achieving an optimal makespan.

Think Ahead: Avoid purely greedy decisions. A choice that seems best locally (e.g., starting a short task on an idle machine) may create a major delay for a more critical task later.

Allow Strategic Idling: Do not be afraid to leave a machine idle if it is waiting for an step that is part of the critical path. A short, strategic wait can prevent a much longer delay later.