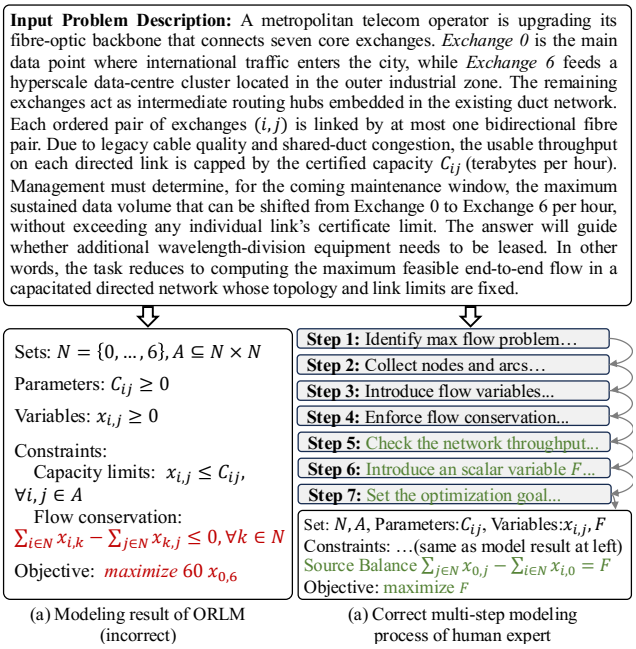
DeepOR: A Deep Reasoning Foundation Model for Optimization Modeling

Ziyang Xiao[1], Yuan Jessica Wang[3], Xiongwei Han[2], Shisi Guan[1], Jingyan Zhu[1], Jingrong Xie[1], Lilin Xu[1], Han Wu[2], Wing Yin Yu[2], Zehua Liu[2], Xiaojin Fu[2], Gang Chen[1], Dongxiang Zhang[1]*

[1] Zhejiang University
[2] Huawei Noah's Ark Lab
[3] School of Business, Singapore University of Social Sciences
{xiaoziyang, zhangdongxiang}@zju.edu.cn, {hanxiongwei, rocket.yuwingyin}@huawei.com

Abstract

Optimization modeling plays a critical role in supporting optimal decision-making across various domains. Previous works have demonstrated that large language models (LLMs) tailored for optimization modeling have significantly automated and simplified this process. However, these models typically employ a straightforward input-output paradigm and struggle with challenging instances. In contrast, recent advances in general-purpose reasoning LLMs (RLLMs), such as DeepSeek-R1, have shown impressive capabilities in complex domains like mathematics and coding. In this paper, we introduce DeepOR, the first RLLM specifically designed for optimization modeling. Instead of directly outputting solutions, DeepOR explicitly performs multiple intermediate reasoning steps. To adapt a base LLM into an RLLM, we begin by synthesizing long chain-of-thought (CoT) data guided by a flowchart, which is automatically generated using a self-exploration algorithm. Once the training data are prepared, we employ supervised fine-tuning on the base LLM to endow it with reasoning capabilities tailored for optimization modeling. To fully leverage the model's reasoning potential, we further apply reinforcement learning with reward-shaping derived from solver feedback. Experimental results on benchmarks confirm that DeepOR consistently and significantly outperforms existing state-of-the-art approaches.



(a) Modeling result of ORLM (incorrect)

(a) Correct multi-step modeling process of human expert

Figure 1: An example of an optimization modeling task.

Introduction

Optimization modeling is fundamental in solving real-world decision-making problems across diverse industries, including supply chain management (Cuthbertson 1998), healthcare (Delgado et al. 2022), and air traffic flow management (de Matos and Ormerod 2000). Traditionally, constructing an optimization model has relied heavily on domain expertise and manual effort, making the process labor-intensive and prone to errors. Advances in large language models (LLMs) have opened new avenues for automating this process, with early studies demonstrating the potential of LLMs to generate optimization models directly from natural language descriptions (Xiao et al. 2024; AhmadiTeshnizi, Gao, and Udell 2024; Tang et al. 2024).

Recently, foundation models in LLMs have been rapidly progressing. OpenAI o1 (OpenAI 2024) and Deepseek R1 (Guo et al. 2025) have sparked a growing body of research into deep reasoning LLMs (RLLMs) (Chen et al. 2025b). RLLMs are models that produce explicit chains-of-thought (CoT) prior to generating the final answers, substantially benefiting tasks such as mathematical problems, programming, and multidisciplinary knowledge tasks (Team et al. 2025; Chen et al. 2025a; Pfister and Jud 2025).

Optimization modeling also requires mathematical reasoning, programming implementation, and specialized domain knowledge, and thus could benefit from reasoning ability. Figure 1 provides an example of a maximum-flow optimization problem. In this example, the correct modeling process, as given by human experts, involves multiple steps, such as identifying the problem type and essential components, and carefully considering network constraints. A critical modeling step is the explicit introduction of an intermediate scalar variable, F, which enables the clear definition of the source and sink balance constraints to facilitate accurate

*Corresponding author

model construction. In contrast, ORLM, as a typical foundation model in optimization modeling, fails to formulate the problem correctly. This is likely because ORLM generates the result directly without any intermediate reasoning steps.

The above example indicates the need for foundation models with domain-specific reasoning capabilities to achieve reliable optimization modeling. However, training a reasoning model is a challenging task because of the absence of datasets with CoT annotations. Existing datasets only contain final optimal values, and CoT annotations by human experts are expensive. Moreover, the optimization modeling process demands rigorous, structured reasoning steps, while existing training methods are general-purpose and may not adequately address the requirements of such a knowledge-intensive domain.

In this work, we propose DeepOR, a deep reasoning foundation model tailored for optimization modeling, which is capable of imitating the expert's thought trajectories in optimization modeling. To achieve this, we introduce a two-stage training framework: (i) **Expertise Tuning**, which endows the model with domain-specific reasoning capabilities through supervised fine-tuning (SFT), and (ii) **Self-Improvement Learning**, which fully explores the reasoning potential via reinforcement learning.

Specifically, in the expertise tuning stage, we first introduce a structured representation known as the **expert flowchart**, which mimics an expert's thought processes when constructing modeling. This flowchart is used to guide the synthesis of CoT data, which is subsequently used to train the foundation model via SFT. Moreover, to reduce reliance on human expert, we propose an algorithm that automatically generates the flowchart. In the self-improvement learning stage, we further enhance the reasoning capabilities acquired previously using reinforcement learning. To provide more informative and stable reward signals, we introduce a **modeling checklist** for reward-shaping. The modeling checklist comprises a series of objective yes-or-no questions, each evaluated by an LLM based on the problem description, modeling output, ground truth and solver feedback. This approach significantly promotes training stability.

We evaluate our approach on 6 diverse OR modeling benchmarks of varying difficulty levels. Experimental results demonstrate that DeepOR consistently outperforms prior learning-based baselines on both easy and hard benchmarks, achieving improvements of approximately 1.4% on EasyLP and 3.7% on ComplexLP. Our contributions can be summarized as follows:

- We introduce **the first deep reasoning foundation model** for optimization modeling.
- We propose a **large-scale data synthesis pipeline** for generating long CoT data for optimization modeling, guided by an automatically generated flowchart.
- We design a **reward-shaping mechanism** based on a modeling checklist that provides comprehensive, structured feedback to increase stability of RL process.
- We empirically validate the effectiveness of our proposed method across diverse OR modeling benchmarks.

Related Work

LLMs for Optimization Modeling

To automate the optimization modeling process, NL4Opt (Natural Language for Optimization) has emerged as a challenging task (Ramamonjison et al. 2023). Its objective is to translate textual descriptions of operations research (OR) problems into formal mathematical modelings. Large language models, prized for their language understanding and reasoning capabilities, have become central to this endeavor. Early work mainly focused on training-free inference frameworks using commercial LLMs, including prompt-based method (Chen, Constante-Flores, and Li 2023; JU et al. 2024; Jiao et al. 2024; Li et al. 2023b), multi-agent collaboration systems (Xiao et al. 2024; AhmadiTeshnizi, Gao, and Udell 2024; Li et al. 2023a; Zhang et al. 2025) and chain-of-thought variants (Deng et al. 2024; Astorga et al. 2024). Recent work has shifted toward training-based methods. For example, ORLM (Tang et al. 2024) and OptMATH (Lu et al. 2025) strengthen base models through instruction tuning on high-quality synthetic data, while LLMOPT (Jiang et al. 2024) further boosts performance via multi-instruction alignment learning. In addition to supervised learning, SIRL (Chen et al. 2025c) explores reinforcement learning strategies, aiming to increase model robustness and mitigate hallucination errors. Despite these advancements, existing models still struggle with complex optimization modeling due to limitations in their multi-step reasoning capabilities.

Reasoning in Large Language Models

Reasoning has become a defining capability for LLMs. Chain-of-Thought (CoT) prompting (Wei et al. 2022) was a pioneering approach that breaks a task into explicit step-by-step intermediate thoughts. This sparked a wave of research with a series of methods that employ simple prompt engineering, such as ReAct (Yao et al. 2023), Reflexion (Shinn, Labash, and Gopinath 2023), and Tree of Thoughts (Hulbert 2023). More recently, training-based approaches have proven even more effective. OpenAI's o1 model set a new benchmark on challenging mathematical tasks (OpenAI 2024). subsequent work replicated its success via supervised fine-tuning using tree-guided synthesis data (Qin et al. 2024) and distillation data (Huang et al. 2024b). Then, DeepSeek-R1 demonstrated that reinforcement learning can teach a model to discover its own reasoning paths, yielding impressive gains on complex problems (Guo et al. 2025). Kimi k1.5 further validated that LLMs could scale their training data by explore tasks guided by reward signals (Chen et al. 2025c). Despite this momentum, a dedicated reasoning model for optimization modeling remains absent.

Proposed Method

Starting from a pretrained model $\mathcal{M}$, our goal is to fine-tune it into a reasoning model $\mathcal{M}_{\mathrm{R}}$ specifically for optimization modeling. Our training framework is presented in Figure 2, which consists of three parts: **(i) Expert Flowchart Generation**, **(ii) Expertise Tuning**, and **(iii) Self-Improvement Learning**.

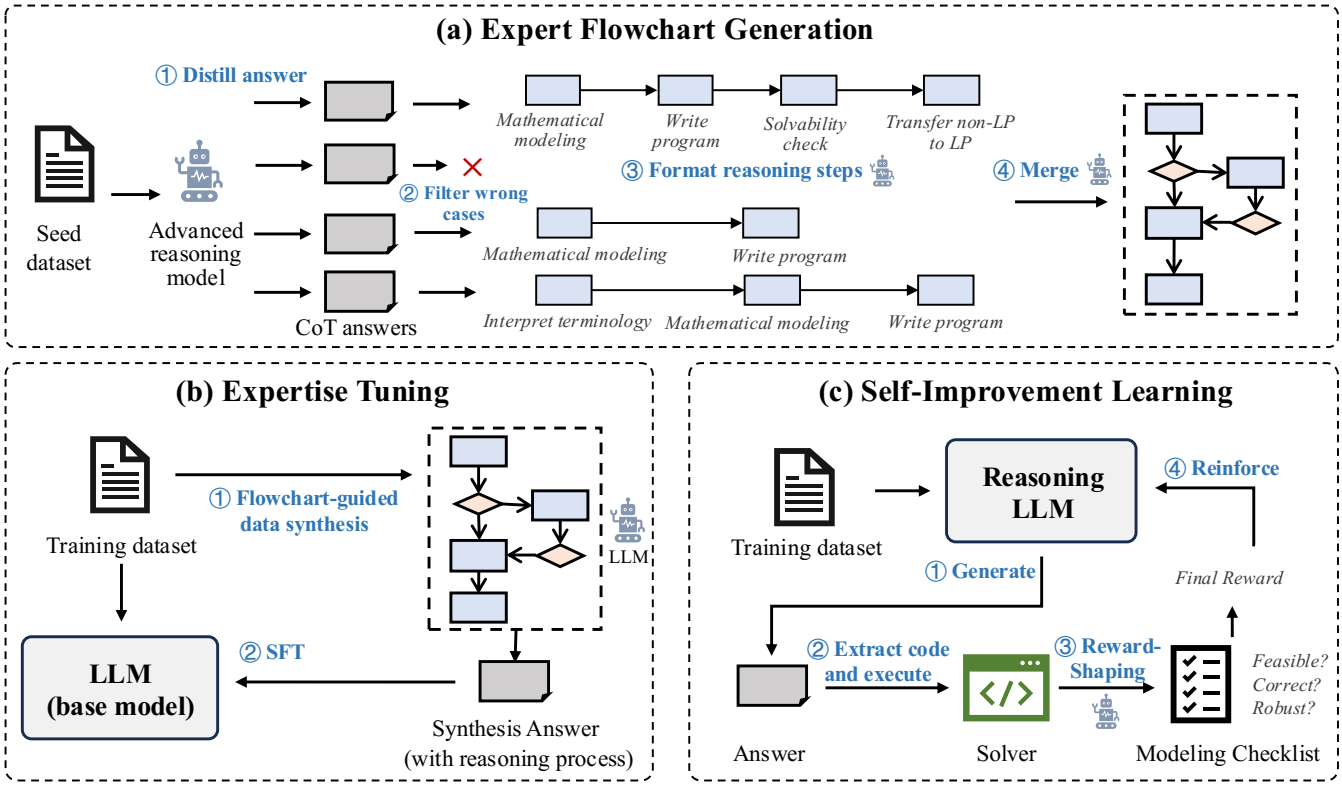


Figure 2: An illustration of DeepOR framework. The bot mark represents the processes using LLM as auxiliary.

Expert Flowchart Generation

Unlike ordinary mathematical or coding tasks, optimization modeling requires a highly domain-specific solving process that differs from general CoT. To synthesize data with CoT for optimization modeling, we first introduce an *expert flowchart* that explicitly describes the problem-solving procedure similar to human experts.

Expert flowchart is a directed acyclic graph (DAG) that represents the process of an LLM solving an optimization modeling problem. Each node within the flowchart falls into one of two categories. Blue rectangular nodes denote *thought steps*, in which the LLM is prompted to generate an intermediate thought. Each thought step corresponds to a distinct action within the problem-solving sequence, such as interpreting terminology or writing code segments. Yellow diamond-shaped nodes represent *decision steps*, where the LLM generates a boolean condition (e.g., checking if the model contains quadratic constraints) to determine the appropriate subsequent reasoning path.

Manually drafting such a flowchart is time-consuming and heavily relies on expert knowledge. Thus, we propose a method to automate the creation of expert flowchart. Let $\mathcal{D}$ be the training dataset. We first select a diverse seed dataset $\mathcal{D}'$ (30 instances in our implementation) and query the most advanced reasoning LLM (*OpenAI o3*) to generate CoT solutions $\mathcal{A}'$. Incorrect outputs are discarded. For each correct solution, a secondary LLM summarizes the textual answer into a list of labels $(l_0, l_1, \ldots, l_n)$, where each l_t represents a concise description of a thought step. These sequences are then merged into a unified flowchart: recurring steps are merged into thought nodes, and decision nodes are inserted wherever identical thought nodes branch into multiple subsequent steps. Each node is assigned a corresponding prompt template that guides the LLM's reasoning or judgment when used for data synthesis. The merging procedure is automated by an LLM. The seed dataset and prompts we used are detailed in the Appendix.

The automatically generated expert flowchart comprises 34 nodes, including 21 thought steps and 13 decision steps, covering the reasoning processes of a wide range of problems. Details of the flowchart are provided in the Appendix.

Expertise Tuning

Once the expert flowchart is generated, we synthesize CoT annotations **at scale**. For each problem instance $p \in \mathcal{D}$, the algorithm traverses the flowchart from the start node. At each thought node t, a non-reasoning LLM generates the intermediate thought z_t guided by the prompt template in the node. At decision steps, the LLM returns a binary indicator $\mathcal{I}_i$ to select the outgoing edge. This produces a synthesized reasoning trace $\tau = (z_0, z_1, \ldots, z_n)$ for each problem.

The synthesized data are utilized for supervised fine-tuning (SFT), transforming the base model $\mathcal{M}$ into a reasoning model $\mathcal{M}'_{\mathrm{R}}$. In our implementation, we randomly sample only $10\,k$ problem instances from the full $210\,k$ corpus to balance training scale and synthesis cost.

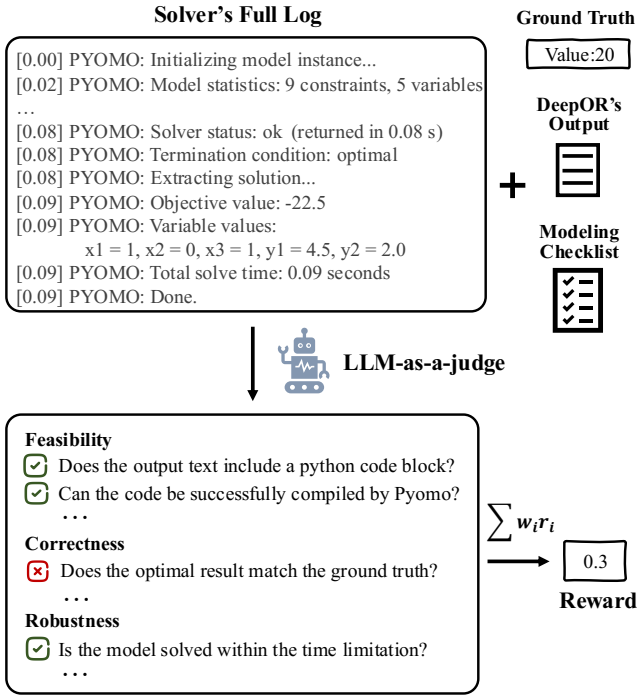


Figure 3: Modeling checklist for reward-shaping.

Self-Improvement Learning

Let the policy be denoted by $\pi_{\mathcal{M}'_R}$. At each step t, the policy selects an action to generate a thought step z_t based on the current state $s_{t-1} = (p, z_0, z_1, \ldots, z_{t-1})$. The newly generated thought z_t is appended to the sequence, forming the updated state $s_t = (p, z_0, z_1, \ldots, z_t)$.

Modeling Checklist for Reward-Shaping The reward signal is crucial for effective reinforcement learning. Although SIRL (Chen et al. 2025c) proposes a reward mechanism leveraging solver feedback, we find that its objective-oriented approach does not necessarily correlate a higher reward with higher modeling quality. This limitation is particularly pronounced in our long CoT setting, where any biases can accumulate throughout the reasoning process.

This observation raises a broader question: **How to effectively assess modeling quality?** The design of any reward-shaping strategy must answer this question.

To this end, we introduce a *modeling checklist*, a comprehensive framework consisting of a series of questions, each targeting a single aspect of model quality. As illustrated in Figure 3, we evaluate three dimensions:

- **Feasibility**: Is the model feasible? For example, does the generated program contain compilation errors, or is the optimization model solvable?
- **Correctness**: Does the model provide the correct answer? For instance, is the result optimal, or is there a gap between the model's solution and the ground truth?
- **Robustness**: How robust is the model? For instance, are the constraints appropriately tight, or do variables correctly reflect integrality constraints in an MIP?

To avoid ambiguity, each question in the modeling checklist is formulated as a yes-or-no question. And each question is designed to be objective and atomic, which means that it evaluates only a single, specific aspect of modeling quality. The complete checklist can be found in the Appendix.

Checking these questions in the checklist can be trivial. Some questions may require reading numerical value from the solver, while others might require LLM judgment. Here, we propose a simple yet effective approach: we provide the original problem, modeling result, solver's entire log, ground truth and the modeling checklist to an LLM, prompting it to return an evaluation result in JSON format. The final reward is computed as a weighted sum of the individual items. Due to the objective nature of checklist questions, we find that the LLM's judgment is robust. The specific prompts and weight used, along with an empirical validation of the LLM's judgment accuracy are detailed in the Appendix.

Training Strategy We use Group Relative Policy Optimization (GRPO) as our RL training algorithm. For each question q, GRPO samples a group of G candidate answers $(o_1, o_2, \ldots, o_G)$ and assigns rewards $(r_1, r_2, \ldots, r_G)$ to each answer using the reward-shaping mechanism.

Furthermore, we introduce an adaptive resampling scheme to boost training efficiency. Within every group we compute the correctness of each answer (already available from reward-shaping) and derive the group-level accuracy Acc. If $Acc = 0$, the entire group is treated as hard and passed to the flowchart-based synthesizer, which produces $\hat{G}$ additional answers $(\hat{o}_1, \hat{o}_2, \ldots, \hat{o}_{\hat{G}})$. These hard examples are then used for SFT. The self-improvement loss combines policy optimization and SFT for resampling instances:

$$\mathcal{L}(\theta) = -\mathcal{J}_{GRPO}(\theta) + \mathcal{L}_{SFT}(\theta).$$

Here, the GRPO term is:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}\big[q \sim P(Q),\ \{o_i\}_{i=1}^{G} \sim \pi_{\theta'}(O|q)\big]$$
$$\frac{1}{G}\sum_{i=1}^{G} \min\Big(\frac{\pi_\theta(o_i|q)}{\pi_{\theta'}(o_i|q)} A_i,\ \mathrm{clip}\big(\frac{\pi_\theta(o_i|q)}{\pi_{\theta'}(o_i|q)}, 1-\epsilon, 1+\epsilon\big) A_i\Big)$$
$$-\beta\, \mathrm{D}_{\mathrm{KL}}\big(\pi_\theta \,\|\, \pi_{\mathrm{ref}}\big),$$

where $\dfrac{\pi_\theta(o_i|q)}{\pi_{\theta'}(o_i|q)}$ is the importance weight and A_i denotes the advantage. The SFT term on hard examples is:

$$\mathcal{L}_{SFT}(\theta) = -\mathbb{E}_{(p,o)\sim\hat{G}} \sum_{i=1}^{\hat{G}} \log_{\pi_\theta}(o_i|o_{0:i-1}, p; \theta).$$

Experiments

Experimental Setup

Datasets. We employ four relatively simple benchmarks to evaluate performance: (i) **NL4Opt**: The first operations research modeling dataset from the NL4Opt competition (Ramamonjison et al. 2023), containing $1,101$ elementary-level linear programming problems; (ii) **EasyLP**: The easier subset of the MAMO dataset (Huang et al. 2024a), comprising 688 instances; (iii) **NLP4LP**: The benchmark utilized in

Methods	NL4Opt	NLP4LP	ReSocratic	EasyLP	ComplexOR	IndustryOR	ComplexLP
gpt-4o	84.5%	70.6%	48.4%	70.3%	42.9%	38.1%	57.7%
Deepseek R1	94.8%	78.6%	70.1%	90.1%	50.0%	43.5%	61.5%
OpenAI o3	96.2%	81.0%	**74.8%**	92.4%	60.7%	47.8%	65.8%
Chain-of-Experts	87.3%	**83.9%**	71.2%	91.2%	57.1%	32.6%	50.6%
OptiMUS	78.8%*	72.0%*	-	-	-	-	-
CAFA	89.2%	54.5%	40.1%	71.2%	46.4%	41.1%	44.5%
ORLM	85.9%	76.4%	61.8%	90.4%	50.0%	41.3%	59.5%
LLMOpt	87.3%	72.0%	54.5%	88.5%	53.6%	42.2%	60.9%
OPTMath	94.3%	73.9%	58.7%	87.0%	50.0%	43.5%	62.1%
SIRL	96.2%	80.6%	72.6%	91.8%	53.6%	45.7%	63.4%
DeepOR	**97.7%**	82.9%	73.8%	**93.2%**	**64.3%**	**52.2%**	**67.1%**

Table 1: Performance comparison of models across benchmarks. Results marked with $*$ for OptiMUS are sourced from the original publication due to input format mismatches encountered during reproduction; all other results are reproduced.

OptiMUS (AhmadiTeshnizi, Gao, and Udell 2024), which is regularly updated and currently comprises 269 instances; and (iv) **ReSocratic**: A dataset consisting of 605 instances, filtered using a quality control framework.

Furthermore, we use three relatively challenging benchmarks to evaluate performance on complex modeling problems: (i) **IndustryOR**: Introduced in ORLM (Tang et al. 2024), consisting of 100 real-world industrial cases; (ii) **ComplexOR**: Used in Chain-of-Experts (Xiao et al. 2024), comprising 37 instances sourced from both industrial and academic scenarios; and (iii) **ComplexLP**: The more challenging subset of MAMO, containing 211 instances.

To enhance the reliability of experimental results, we adopt data source that have been manually cleaned or corrected based on a survey of optimization modeling [1].

Baselines. We compare our proposed method with the following baselines: (i) general-purpose LLMs, including non-reasoning models (e.g., **GPT-4o**) and reasoning models (e.g., **Deepseek R1**, **OpenAI o3**); (ii) multi-agent frameworks specifically designed for operations research, such as **Chain-of-Experts**, **OptiMUS**, and **CAFA**; and (iii) fine-tuned LLMs tailored for operations research tasks, including supervised fine-tuning models (e.g., **ORLM**, **LLMOpt**, **OPTMath**) and **SIRL**, a model trained with reinforcement learning.

Training setup. We utilize the widely-adopted open-source model Qwen3-8B as our base model in the main experiments. For fair comparison with prior work that employed different base models, we also conduct an ablation study across different base models.

We adopt the training dataset introduced by OptMath (Lu et al. 2025), which comprises $210\,k$ operations research problem-modeling pairs. In expertise tuning stage, we train the model using the AdamW optimizer (learning rate 2×10^{-5}, cosine decay, weight decay 0.1) for 3 epochs. In self-improvement learning stage, we conduct training for 5 epochs on the entire $210\,k$ dataset using the same hyperparameter settings as in the expertise tuning. The group size G in GRPO is set to 5 with a coefficient β of 0.05. We use Qwen3-4B as the LLM judger in reward-shaping.

Evaluation setup. For fairness, we set the temperature to 0.0 and top-p to 1.0. Greedy decoding with repetition penalty 1.0 is applied to all fine-tuning methods. For all multi-agent frameworks (e.g., Chain-of-Experts, OptiMUS and CAFA), we utilize the same version of commercial LLM *gpt-4o* as the base model. The primary evaluation metric is the pass@1 accuracy based on the final objective value.

Overall Performance

The overall experimental results for the baselines are shown in Table 1. Firstly, compared with other fine-tuning models, DeepOR achieves state-of-the-art accuracy across all optimization modeling benchmarks. Among the fine-tuning models, SIRL is the most competitive model, as it also incorporates reinforcement learning in its training process. DeepOR demonstrates a slight superiority over SIRL in the four simpler benchmarks. Furthermore, the advantages of DeepOR become more pronounced in the challenging benchmarks. Specifically, in ComplexOR, DeepOR significantly enhances performance from 53.6% to 64.3%. In IndustryOR and ComplexLP, DeepOR achieves improvements of 6.5% and 3.7%, respectively. This highlights the effectiveness of the tailored CoT reasoning capability in DeepOR for solving complex instances. In addition, we observe that LLMOpt and OptMath outperform ORLM on benchmarks such as NL4Opt, IndustryOR, and MAMO (EasyLP and ComplexLP) but underperform on other benchmarks. This discrepancy may stem from limited generalizability due to their non-reasoning architectures. In contrast, DeepOR exhibits better generalization across diverse benchmarks.

Secondly, compared to state-of-the-art general-purpose reasoning LLMs such as Deepseek R1 and OpenAI o3, DeepOR demonstrates distinct advantages. General-purpose reasoning models also exhibit strong performance in optimization modeling tasks, particularly OpenAI o3. However, DeepOR surpasses OpenAI o3 across most benchmarks, especially in the more challenging ones. For example, in the IndustryOR benchmark, DeepOR achieves an accuracy of

[1] https://llm4or.github.io/LLM4OR/

Methods	EasyLP	ComplexLP
DeepOR (Full)	93.2%	67.1%
w/o Expertise Tuning	**72.8%**	**55.9%**
w/o Self-Improvement	91.0%	62.7%
w/o Flowchart (+line CoT)	89.5%	61.5%
w/o Flowchart (+CAFA CoT)	92.8%	65.8%
w/o Reward-Shaping	92.5%	64.6%

Table 2: Ablation study of DeepOR.

Base Model	Methods	EasyLP	ComplexLP
Qwen3-8B	DeepOR	93.2%	67.1%
LLaMa3-8B	DeepOR	**92.0%**	**62.7%**
	ORLM	89.4%	57.1%
Qwen2.5-7B	DeepOR	**92.5%**	**64.0%**
	ORLM	90.4%	59.5%
	OPTMath	87.0%	62.1%
	SIRL	91.8%	63.4%

Table 3: Sensitivity analysis results on different base models.

52.2%, significantly outperforming Deepseek R1's 43.5% and OpenAI o3's 47.8%. This highlights DeepOR's enhanced reasoning capabilities and domain-specific knowledge in optimization modeling, which general-purpose LLMs typically lack. Notably, OpenAI o3 narrowly outperforms DeepOR in the ReSocratic benchmark by 1%. This minor edge could stem from the fact that ReSocratic is synthesized entirely using GPT with a filtering mechanism, giving OpenAI o3 an inherent advantage. Additionally, it is worth mentioning that DeepOR is significantly smaller in model size compared to these general-purpose LLMs, allowing for faster inference and making it more suitable for practical usage.

Thirdly, multi-agent frameworks also do not exhibit performance comparable to fine-tuning reasoning models. According to the results, the most competitive multi-agent framework, Chain-of-Experts, shows a noticeable performance gap compared to SIRL and DeepOR. When applied to optimization modeling tasks, multi-agent frameworks can be seen as specialized CoT workflows designed explicitly for optimization modeling. In such frameworks, the capability for multi-step reasoning is enforced externally through workflow design rather than inherently developed within the model. The experimental results suggest that internally training a model's CoT reasoning ability is a more efficient approach in optimization modeling tasks.

Ablation Study

We conduct an ablation study to assess the contributions of key components. We select two representative benchmarks: EasyLP (simpler) and ComplexLP (more challenging). As shown in Table 2, expertise tuning is crucial to our training framework, as its removal significantly reduces accuracy by 20.4% on EasyLP and 11.2% on ComplexLP. This substantial drop indicates that, without expertise tuning, the base model lacks the domain-expert reasoning capability required for optimization modeling, rendering self-improvement learning ineffective on its own.

Self-improvement learning also proves important to DeepOR, especially in the ComplexLP benchmark, where accuracy decreases from 67.1% to 62.7% without it. Additionally, removing reward-shaping exhibits varying impacts across benchmarks: EasyLP accuracy declines by a modest 0.7%, while ComplexLP experiences a more noticeable reduction of 2.5%. This suggests that reward-shaping significantly improves model robustness on harder instances.

Furthermore, we explore two alternative CoT data synthesis variants to our flowchart-based method: (i) line CoT, where each line represents an intermediate reasoning step of the LLM; and (ii) CAFA CoT, a manually designed CoT pattern for optimization modeling introduced by CAFA that simplifies the modeling process into four fixed steps. Interestingly, substituting flowchart with line CoT leads to a more significant accuracy drop compared to removing self-improvement learning, highlighting that selecting an appropriate CoT pattern can be even more crucial than employing reinforcement learning. Additionally, CAFA CoT results in a slight accuracy reduction, demonstrating that LLMs' self-explored problem-solving patterns outperform those manually crafted by human experts.

Sensitivity Analysis on Different Base Models

We select Qwen3-8B as our primary base model. However, given the varied release times and implementations, previous fine-tuning approaches have utilized different base models. For fair comparison, we conduct a sensitivity analysis by evaluating our approach across two other popular base models, including LLaMa3-8B and Qwen2.5-7B. Table 3 presents these results. Many baseline methods, including ORLM, OPTMath, and SIRL, are trained on Qwen2.5-7B by default. When we adapt DeepOR to this base model, it still achieves superior performance compared to other methods. Similarly, when comparing DeepOR and ORLM on LLaMa3-8B, DeepOR significantly outperforms ORLM. Thus, we attribute the superior performance of DeepOR primarily to its advanced training framework rather than merely the choice of base model.

Additionally, we observe performance degradation for both DeepOR and ORLM when transitioning from Qwen3-8B to Qwen2.5-7B, and further to LLaMa3-8B. This indicates that the choice of base model does indeed impact the performance of fine-tuned models. Generally, better base models lead to improved results. Notably, transitioning from Qwen2.5-7B to LLaMa3-8B results in accuracy drops of 1.0% and 2.4% for ORLM in EasyLP and ComplexLP, respectively, whereas DeepOR's accuracy declines by only 0.3% and 1.3%. This finding shows DeepOR's robustness and relative insensitivity to changes in the base model.

Furthermore, we note a substantial performance improvement in ComplexLP when switching from the non-reasoning base model Qwen2.5-7B to the reasoning-enabled Qwen3-8B. This suggests that incorporating reasoning capabilities

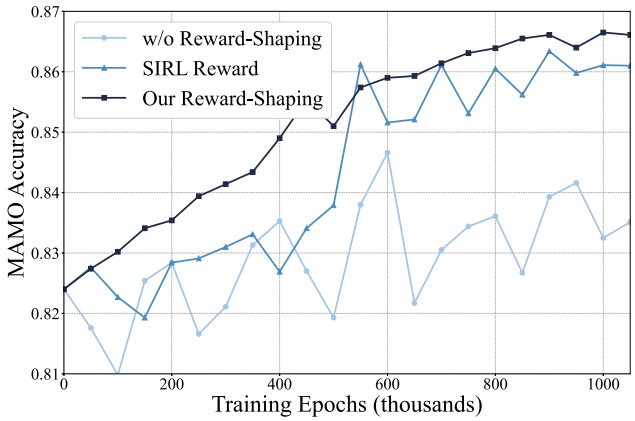


Figure 4: Accuracy on MAMO dataset during self-improvement learning with different reward-shaping strategies.

into the base model significantly enhances performance on more complex optimization modeling tasks.

Analysis of Reward-Shaping Strategies

We also examine how alternative reward-shaping schemes affect RL training. Figure 4 plots the accuracy on the MAMO dataset throughout the self-improvement learning. With vanilla sparse rewards, we observe highly unstable training dynamics. The accuracy experiences dramatic fluctuations, peaking around $600\,k$ epochs before rapidly deteriorating. This instability stems from the sparsity of reward signals, which impedes the model's ability to learn robust modeling patterns. When incorporating SIRL rewards, training stability improves notably. However, the learning trajectory displays a distinct "mutagenic" pattern, where performance stagnates at a low level for an extended period before abruptly improving around $550\,k$ epochs and subsequently plateauing.

The performance of our checklist-based method increases smoothly and consistently. Although SIRL improves more rapidly in early stages, our method ultimately achieves a significantly higher terminal accuracy. We attribute this superior performance to the checklist's fine-grained feedback, which jointly evaluates feasibility, correctness, and robustness, rather than focusing solely on the final objective.

In contrast, our approach demonstrates remarkably stable and consistent performance improvements throughout training. Although SIRL improves rapidly during a brief period, our method ultimately achieves a higher terminal accuracy. We attribute this superior performance to the checklist's dense and fine-grained feedback, which jointly evaluates feasibility, correctness, and robustness, rather than focusing solely on the final objective value.

Case Study

Figure 5 presents a typical case study comparing the modeling results of SIRL and DeepOR. Although both methods yield the same final optimal solution, the model formulations differ significantly. DeepOR's generated code is notably clearer in structure and more sophisticated. For example, it provides more descriptive comments, uses lambda expressions to make code clear, and pays attention to method encapsulation. Furthermore, DeepOR's modeling introduces an intermediate scalar variable F to explicitly manage the total flow. While this difference does not affect the optimal solution, the explicit total-flow variable may reduce model-building overhead and help the solver's presolve on larger instances. The modeling result reveals that DeepOR produces a more robust and concise formulation.

```
…
# Constraints
for n in nodes:
  in  = gp.quicksum(flow[i, j] for (i, j) in arcs if j==n)
  out = gp.quicksum(flow[i, j] for (i, j) in arcs if i==n)
  if n == source:
    m.addConstr(in - out == total, "balance_source")
  elif n == sink:
    m.addConstr(in - out == total, "balance_sink")
  else:
    m.addConstr(in == out)
for (i, j), cap in capacity.items():
    m.addConstr(flow[i, j] <= cap)
…
```

(a) Modeling Result of SIRL

```
…
def _balance_rule(m, n):
  inflow = sum(m.Flow[i, j] for (i, j) in m.Arcs if j==n)
  outflow= sum(m.Flow[i, j] for (i, j) in m.Arcs if i==n)
  if   n == SOURCE:
    return outflow - inflow  == m.F  # net supply
  elif n == SINK:
    return inflow  - outflow == m.F  # net demand
  else:
    return inflow == outflow         # transshipment
m.Balance = Constraint(m.Nodes, rule=_balance_rule)
# ----- Capacity constraints ------
m.CapacityLimit = Constraint(
    m.Arcs,
    rule=lambda m, i, j: m.Flow[i, j] <= m.Cap[i, j],
    doc="Cannot exceed certified link limit")
…
```

(b) Modeling Result of DeepOR

Figure 5: Case study. The input problem is the same as the example in the introduction. Due to page limitation, only the constraints part is shown. The full response can be found in the Appendix.

Conclusion

In this paper, we introduced DeepOR, the first deep reasoning foundation model designed for optimization modeling. By systematically synthesizing long CoT data guided by an automatically generated expert flowchart, and subsequently employing supervised fine-tuning and reinforcement learning with a novel checklist-based reward-shaping, DeepOR effectively addresses the limitations of existing LLMs in complex optimization tasks. Experimental results across diverse benchmarks demonstrate that DeepOR is superior to all baselines, particularly excelling in more challenging modeling scenarios.

Acknowledgments

This work is supported by the "Pioneer" R&D Program of Zhejiang (2025C01001) and the Fundamental Research Funds for the Central Universities (226-2024-00145, 226-2024-00216).

References

AhmadiTeshnizi, A.; Gao, W.; and Udell, M. 2024. OptiMUS: Scalable Optimization Modeling with (MI)LP Solvers and Large Language Models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Astorga, N.; Liu, T.; Xiao, Y.; and van der Schaar, M. 2024. Autoformulation of Mathematical Optimization Models Using LLMs. arXiv:2411.01679.

Chen, H.; Constante-Flores, G. E.; and Li, C. 2023. Diagnosing Infeasible Optimization Problems Using Large Language Models. *CoRR*, abs/2308.12923.

Chen, J.; Tang, G.; Zhou, G.; and Zhu, W. 2025a. ChatGPT and Deepseek: Can They Predict the Stock Market and Macroeconomy? arXiv:2502.10008.

Chen, Q.; Qin, L.; Liu, J.; Peng, D.; Guan, J.; Wang, P.; Hu, M.; Zhou, Y.; Gao, T.; and Che, W. 2025b. Towards Reasoning Era: A Survey of Long Chain-of-Thought for Reasoning Large Language Models. arXiv:2503.09567.

Chen, Y.; Xia, J.; Shao, S.; Ge, D.; and Ye, Y. 2025c. Solver-Informed RL: Grounding Large Language Models for Authentic Optimization Modeling. arXiv:2505.11792.

Cuthbertson, R. W. 1998. The Logic of Logistics: Theory, Algorithms and Applications for Logistics Management. *J. Oper. Res. Soc.*, 49(9): 1016–1017.

de Matos, P. A. L.; and Ormerod, R. J. 2000. The application of operational research to European air traffic flow management - understanding the context. *Eur. J. Oper. Res.*, 123(1): 125–144.

Delgado, E. J.; Cabezas, X.; Martin-Barreiro, C.; Leiva, V.; and Rojas, F. 2022. An equity-based optimization model to solve the location problem for healthcare centers applied to hospital beds and COVID-19 vaccination. *Mathematics*, 10(11): 1825.

Deng, H.; Zheng, B.; Jiang, Y.; and Tran, T. H. 2024. CAFA: Coding as Auto-Formulation Can Boost Large Language Models in Solving Linear Programming Problem. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*.

Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; and Xiao Bi, e. a. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *CoRR*, abs/2501.12948.

Huang, X.; Shen, Q.; Hu, Y.; Gao, A.; and Wang, B. 2024a. Mamo: a Mathematical Modeling Benchmark with Solvers. arXiv:2405.13144.

Huang, Z.; Zou, H.; Li, X.; Liu, Y.; Zheng, Y.; Chern, E.; Xia, S.; Qin, Y.; Yuan, W.; and Liu, P. 2024b. O1 Replication Journey–Part 2: Surpassing O1-preview through Simple Distillation, Big Progress or Bitter Lesson? *arXiv preprint arXiv:2411.16489*.

Hulbert, D. 2023. Tree of Knowledge: ToK aka Tree of Knowledge dataset for Large Language Models LLM. https://github.com/dave1010/tree-of-thought-prompting.

Jiang, C.; Shu, X.; Qian, H.; Lu, X.; Zhou, J.; Zhou, A.; and Yu, Y. 2024. LLMOPT: Learning to Define and Solve General Optimization Problems from Scratch. *CoRR*, abs/2410.13213.

Jiao, Z.; Sha, M.; Zhang, H.; Jiang, X.; and Qi, W. 2024. City-LEO: Toward Transparent City Management Using LLM with End-to-End Optimization. *CoRR*, abs/2406.10958.

JU, D.; Jiang, S.; Cohen, A.; Foss, A.; Mitts, S.; Zharmagambetov, A.; Amos, B.; Li, X.; Kao, J. T.; Fazel-Zarandi, M.; and Tian, Y. 2024. To the Globe (TTG): Towards Language-Driven Guaranteed Travel Planning. *CoRR*, abs/2410.16456.

Li, B.; Mellou, K.; Zhang, B.; Pathuri, J.; and Menache, I. 2023a. Large Language Models for Supply Chain Optimization. *CoRR*, abs/2307.03875.

Li, R.; Pu, C.; Tao, J.; Li, C.; Fan, F.; Xiang, Y.; and Chen, S. 2023b. LLM-based Frameworks for Power Engineering from Routine to Novel Tasks. arXiv:2305.11202.

Lu, H.; Xie, Z.; Wu, Y.; Ren, C.; Chen, Y.; and Wen, Z. 2025. OptMATH: A Scalable Bidirectional Data Synthesis Framework for Optimization Modeling. arXiv:2502.11102.

OpenAI. 2024. Learning to Reason with LLMs. https://openai.com/index/learning-to-reason-with-llms/. [Accessed 19-09-2024].

Pfister, R.; and Jud, H. 2025. Understanding and Benchmarking Artificial Intelligence: OpenAI's o3 Is Not AGI. arXiv:2501.07458.

Qin, Y.; Li, X.; Zou, H.; Liu, Y.; Xia, S.; Huang, Z.; Ye, Y.; Yuan, W.; Liu, H.; Li, Y.; et al. 2024. O1 Replication Journey: A Strategic Progress Report–Part 1. *arXiv preprint arXiv:2410.18982*.

Ramamonjison, R.; Yu, T. T. L.; Li, R.; Li, H.; Carenini, G.; Ghaddar, B.; He, S.; Mostajabdaveh, M.; Banitalebi-Dehkordi, A.; Zhou, Z.; and Zhang, Y. 2023. NL4Opt Competition: Formulating Optimization Problems Based on Their Natural Language Descriptions. *CoRR*, abs/2303.08233.

Shinn, N.; Labash, B.; and Gopinath, A. 2023. Reflexion: an autonomous agent with dynamic memory and self-reflection. *CoRR*, abs/2303.11366.

Tang, Z.; Huang, C.; Zheng, X.; Hu, S.; Wang, Z.; Ge, D.; and Wang, B. 2024. ORLM: Training Large Language Models for Optimization Modeling. *CoRR*, abs/2405.17743.

Team, K.; Du, A.; Gao, B.; Xing, B.; Jiang, C.; and Cheng Chen, e. a. 2025. Kimi k1.5: Scaling Reinforcement Learning with LLMs. arXiv:2501.12599.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E. H.; Le, Q. V.; and Zhou, D. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.

Xiao, Z.; Zhang, D.; Wu, Y.; Xu, L.; Wang, Y. J.; Han, X.; Fu, X.; Zhong, T.; Zeng, J.; Song, M.; and Chen, G. 2024. Chain-of-Experts: When LLMs Meet Complex Operations Research Problems. OpenReview.net.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K. R.; and Cao, Y. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Zhang, Y.; Kang, Q.; YU, W. Y.; HaileiGong; Fu, X.; Han, X.; Zhong, T.; and Ma, C. 2025. Decision Information Meets Large Language Models: The Future of Explainable Operations Research. In *The Thirteenth International Conference on Learning Representations*.