OR-R1: Automating Modeling and Solving of Operations Research Optimization Problem via Test-Time Reinforcement Learning

Zezhen Ding[1], Zhen Tan[2], Jiheng Zhang[1*], Tianlong Chen[3*]

[1]The Hong Kong University of Science and Technology
[2]Arizona State University
[3]University of North Carolina at Chapel Hill
zdingah@connect.ust.hk, ztan36@asu.edu, jiheng@ust.hk, tianlong@cs.unc.edu

Abstract

Optimization modeling and solving are fundamental to the application of Operations Research (OR) in real-world decision making, yet the process of translating natural language problem descriptions into formal models and solver code remains highly expertise intensive. While recent advances in large language models (LLMs) have opened new opportunities for automation, the generalization ability and data efficiency of existing LLM-based methods are still limited, as most require vast amounts of annotated or synthetic data, resulting in high costs and scalability barriers. In this work, we present OR-R1, a data-efficient training framework for automated optimization modeling and solving. OR-R1 first employs supervised fine-tuning (SFT) to help the model acquire the essential reasoning patterns for problem formulation and code generation from limited labeled data. In addition, it improves the capability and consistency through Test-Time Group Relative Policy Optimization (TGRPO). This two-stage design enables OR-R1 to leverage both scarce labeled and abundant unlabeled data for effective learning. Experiments show that OR-R1 achieves state-of-the-art performance with an average solving accuracy of 67.7%, using only $1/10$ the synthetic data required by prior methods such as ORLM, exceeding ORLM's solving accuracy by up to 4.2%. Remarkably, OR-R1 outperforms ORLM by over 2.4% with just 100 synthetic samples. Furthermore, TGRPO contributes an additional $3.1\% - 6.4\%$ improvement in accuracy, significantly narrowing the gap between single-attempt (Pass@1) and multi-attempt (Pass@8) performance from 13% to 7%. Extensive evaluations across diverse real-world benchmarks demonstrate that OR-R1 provides a robust, scalable, and cost-effective solution for automated OR optimization problem modeling and solving, lowering the expertise and data barriers for industrial OR applications.

Code — https://github.com/SCUTE-ZZ/OR-R1

Introduction

The field of artificial intelligence has witnessed remarkable advancements, with Large Language Models (LLMs) emerging as powerful tools across diverse domains (Zhao et al. 2025; He et al. 2024; Jiang et al. 2024b). A particularly promising, yet challenging, application lies in automating the modeling and solving of optimization problems (Ramamonjison et al. 2023). These problems are central to numerous scientific and industrial applications, including logistics (Lee, Lee, and Zhang 2015; Harrison et al. 2019), resource allocation (Bretthauer and Shetty 1995), and scheduling (Brucker et al. 1999; Long et al. 2020), where even minor improvements can yield significant real-world benefits. Traditionally, formulating these problems into precise mathematical models and subsequently generating executable solver code has demanded specialized human expertise, often a time-consuming and error-prone process (Huang et al. 2025; Jiang et al. 2024a). LLMs have emerged as a promising tool to automate this process, reducing the expertise barrier for solving optimization problems.

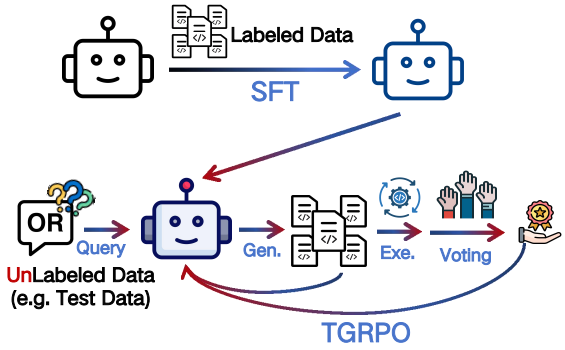

Figure 1: Overview of OR-R1.

Recent research has explored the integration of LLMs into the optimization pipeline, broadly falling into two main categories: prompt-based approaches and learning-based approaches. Prompt-based approaches typically leverage LLMs' in-context learning capabilities through carefully designed prompts, few-shot examples, or chain-of-thought methods to generate optimization models or code directly. Notable works in this category include the Chain-of-Experts (Xiao et al. 2023), Optimus (AhmadiTeshnizi, Gao, and Udell 2024), MAMO (Huang et al. 2024b), ORQA (Mostajabdaveh et al. 2025), OR-LLM-Agent (Zhang and Luo 2025), and OptimAI (Thind et al. 2025). These methods of-

*Corresponding author.

ten utilize general-purpose LLMs without extensive domain-specific fine-tuning, relying on their pre-trained knowledge and sophisticated prompting strategies. In contrast, learning-based approaches involve fine-tuning or training LLMs on domain-specific datasets to enhance their understanding and generation capabilities for optimization problems. Key contributions in this area include ORLM (Huang et al. 2025), LLMOPT (Jiang et al. 2024a), and OptiBench (Yang et al. 2024b). These works aim to build more specialized and performant models through data-driven learning.

Learning-based approaches hold great promise for automated optimization; for instance, ORLM, using a Llama3-8B model, has even surpassed GPT-4o (Hurst et al. 2024) in some tests. Despite this, a major hurdle remains: the heavy reliance on vast amounts of domain-specific data. While synthetic data can be mass-produced, it often lacks the real-world rigor and diversity of human-made examples. Conversely, manual annotation of high-quality data is extremely costly, leading to a shortage of large, useful datasets and limiting adoption. Our main goal is to significantly cut down on the amount of labeled data needed. Moreover, research, including insights from ORLM, shows another key challenge: LLMs can find optimal solutions, but their single-attempt outputs often lack consistency. This means they're more likely to get the right answer if they generate multiple times. This inconsistency highlights a clear need to improve the reliability of individual outputs.

To address these critical issues, the large data demands and inconsistent outputs, we introduce OR-R1. This innovative learning-based framework is specifically designed to boost LLM performance in Operations Research by requiring far less data and delivering greater consistency. First, OR-R1 uses Supervised Fine-Tuning (SFT) to teach the core reasoning of the model for optimization modeling and code generation with limited labeled data. Then, it integrates Test-Time Group Relative Policy Optimization (TGRPO). TGRPO works by having the LLM predict labels for unlabeled data, using a voting system to create high-quality pseudo-labels. These pseudo-labels then serve as a reward function for reinforcement learning. This two-part strategy not only drastically cuts the need for expensive labeled training data but also substantially improves the consistency of the model's predictions, making single-attempt generations much more reliable.

Our main contributions and findings are:

- We introduce **OR-R1**, a novel framework that, for the first time, integrates SFT and **TGRPO** for automated operations research problems modeling and solving.
- We design a multi-faceted reward system specifically tailored for optimization problem scenarios, comprising **Format Reward** for structural correctness, **Valid-Code Reward** for executability, **Majority Voting Reward** for numerical accuracy, enabling robust model learning.
- OR-R1 surpasses previous state-of-the-art methods using only 1/10 synthetic data of ORLM, and attains an average solving accuracy of **67.7%** across diverse public benchmarks, while significantly narrowing the gap between single-attempt and multi-attempt accuracy.

Related Work

Automated OR Problem Modeling and Solving. Large Language Models (LLMs) have demonstrated impressive capabilities in formal reasoning and code generation, advancing from solving math word problems (Cobbe et al. 2021; Hendrycks et al. 2021; Ahn et al. 2024) to generating competition-level code (Li et al. 2022; Chaudhary 2023; Bairi et al. 2024). Meanwhile, methods such as few-shot learning (Brown et al. 2020), Chain-of-Thought prompting (Wei et al. 2022), Tree-of-Thoughts (Yao et al. 2023), and Graph-of-Thoughts (Besta et al. 2024) have significantly enhanced the reasoning abilities of Large Language Models (LLMs). Building upon these foundational advancements in general math and coding, a specialized and highly impactful application of LLMs has emerged in the domain of Operations Research (OR). Specifically, LLMs are now being used for automated optimization modeling. This involves translating natural language descriptions of real-world problems into precise mathematical optimization formulations (e.g., MILP, LP) or directly into executable solver code. Research in this area can be broadly categorized into prompt-based and learning-based approaches. Prompt-based methods leverage large pretrained LLMs with sophisticated prompting strategies to generate models. Notable examples include the NL4Opt competition (Ramamonjison et al. 2023), which showcased LLMs' potential in understanding problem descriptions, and frameworks such as Optimus (AhmadiTeshnizi, Gao, and Udell 2024), MAMO (Huang et al. 2024b), ORQA (Mostajabdaveh et al. 2025), OR-LLM-Agent (Zhang and Luo 2025), OptMATH (Lu et al. 2025), and OptimAI (Thind et al. 2025), which explore various prompting strategies, agent-based systems, and benchmarks for OR problem solving. In contrast, learning-based approaches involve fine-tuning LLMs on domain-specific datasets to achieve deeper understanding and generation capabilities. Key contributions include ORLM (Huang et al. 2025), LLMOPT (Jiang et al. 2024a), and OptiBench (Yang et al. 2024b), which focuses on creating specialized models or benchmarks for this task. However, these methods typically rely on large amounts of synthetic data or costly human-annotated data for training. Synthetic data often suffers from accuracy issues, as highlighted by ORLM, where their synthetic data had an accuracy of only around 70%. In contrast, acquiring high-quality human-annotated data is expensive and time consuming.

RL for LLM Alignment and Reasoning. Reinforcement Learning (RL) has become a cornerstone for aligning LLMs with human preferences and improving their reasoning capabilities. Initial works like RLHF (Ouyang et al. 2022; Christiano et al. 2017) and Direct Preference Optimization (DPO) (Rafailov et al. 2023) have shown the effectiveness of fine-tuning LLMs with human feedback. More advanced RL techniques, often based on Proximal Policy Optimization (PPO) (Schulman et al. 2017), are used to refine model behaviors for complex tasks. DeepSeekMath (Shao et al. 2024), which introduced Group Relative Policy Optimization (GRPO), demonstrated how RL can push the limits of mathematical reasoning by learning from comparisons between multiple generated solutions. Subsequent

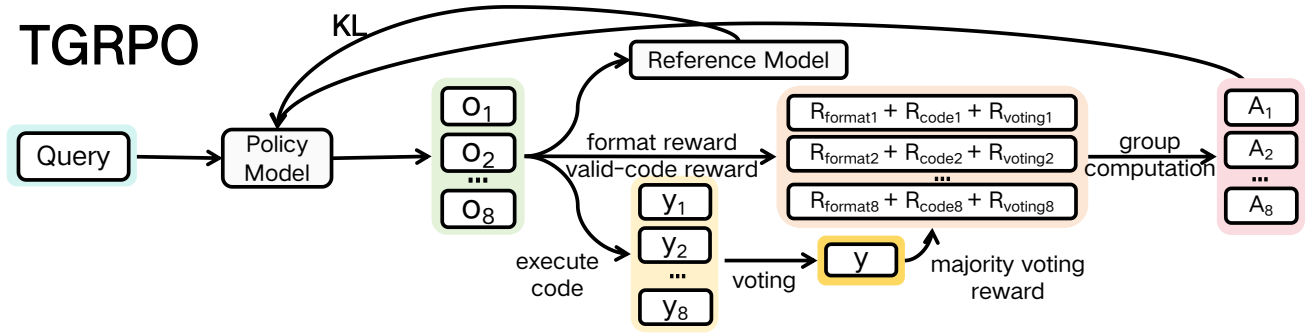

Figure 2: The figure illustrates the training process of TGRPO.

work, DeepSeek-R1 (Guo et al. 2025), further explores incentivizing reasoning capabilities via RL. Other research has focused on scaling RL systems for LLMs (Yu et al. 2025; Sheng et al. 2024; Zhang et al. 2024), self-correction mechanisms (Kumar et al. 2024; Qu et al. 2024) or formal provers (Xin et al. 2024) to guide RL training.

Test-Time Adaptation and Confidence in LLMs. Ensuring the reliability and accuracy of LLM outputs, particularly at inference time, is crucial. In complex problem-solving tasks, such as mathematical reasoning and code generation, it's commonly observed that approaches employing multiple votings over generated candidate solutions, followed by selection of the most consistent or accurate, achieve significantly better results than single-attempt methods (Pass@1) (Chen et al. 2023; Huang et al. 2024a). This phenomenon highlights that improving the consistency of generated outputs can lead to a substantial enhancement in Pass@1 performance. To address this, Test-Time Adaptation (TTA) methods aim to adapt models to new data distributions or improve performance. Examples include TENT (Wang et al. 2020), which minimizes entropy during inference. Building on this, Test-Time Reinforcement Learning (Zuo et al. 2025; Prabhudesai et al. 2025; Yu et al. 2025; Sheng et al. 2024; Zhang et al. 2024; Kumar et al. 2024; Qu et al. 2024; Xin et al. 2024) extends TTA by applying reinforcement learning principles to adapt models at test time. By employing strategies like majority voting among candidate outputs and learning from consistency signals, these methods enable models to adapt on-the-fly to distribution shifts or challenging instances without extensive retraining. Despite these advances, such RL and adaptation techniques have not yet been systematically explored for solving operations research (OR) problems. In this work, we conduct the first application of these methods to automated OR problem solving.

Method

Overview

The overall workflow of the OR-R1 training procedure is illustrated in Figure 1. Our method utilizes the Qwen3-8B model as the base. First, we perform supervised fine-tuning (SFT) using a small, randomly selected subset of data from the 3000 IndustryOR dataset, an open-source resource from ORLM. Second, the model undergoes further training with TGRPO on unlabeled test set data. TGRPO training is guided by a composite reward function specifically designed for OR problem modeling and solving. This reward integrates several components derived from the generated outputs: (1) Format Reward, which measures adherence to the required structural format; (2) Valid-Code Reward, which checks the syntactic correctness and executability of the generated code; and (3) Majority Voting Reward, which reflects the consensus among multiple candidate solutions through majority voting. By combining these criteria, the reward function incentivizes the model to generate well-structured, functional, and consistent solutions for OR tasks.

Supervised Fine-tuning (SFT) Phase

The objective of the SFT phase is to maximize the likelihood of generating the correct output given the input. This is achieved by minimizing the negative logarithmic likelihood loss, a standard approach in supervised learning. The objective function for SFT is formally defined as:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(x,o)\sim\mathcal{D}_{\text{SFT}}}\left[\sum_{t=1}^{|o|}\log P(o_t|x,o_{<t};\theta)\right] \quad (1)$$

where

- $\mathcal{L}_{\text{SFT}}(\theta)$ represents the loss function for the model parameters θ.
- $\mathbb{E}_{(x,o)\sim\mathcal{D}_{\text{SFT}}}$ denotes the expectation over the input-output pairs (x, o) sampled from the supervised data set $\mathcal{D}_{\text{SFT}}$, where x is the input prompt and $o = (o_1, o_2, \ldots, o_{|o|})$ is the target output sequence.
- $P(o_t|x, o_{<t};\theta)$ is the probability of generating the t-th token o_t given the input prompt x, all preceding tokens $y_{<t}$, and the model parameters θ.

Test-Time Group Relative Policy Optimization (TGRPO)

The theoretical foundation of TGRPO is consistent with that of GRPO. The training process is illustrated in Figure 2. A key characteristic of TGRPO is its ability to forego a separate critic model, instead estimating the baseline from group scores, significantly reducing computational training

resources. The objective of TGRPO is to maximize the objective of the policy, driving the model to produce higher-quality outputs based on reward signals. The objective function for TGRPO is formally defined as:

$$\mathcal{J}_{TGRPO}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{old}}(O|q)}\Big[\frac{1}{G}\sum_{i=1}^{G}\Big(\min\Big(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}A_i, \text{clip}\Big(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1-\epsilon, 1+\epsilon\Big)A_i\Big) - \beta\mathbb{D}_{KL}(\pi_\theta||\pi_{ref})\Big)\Big] \quad (2)$$

$$\mathbb{D}_{KL}(\pi_\theta||\pi_{ref}) = \frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - \log\frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - 1 \quad (3)$$

$$A_i = \frac{R_i - \text{mean}(\{R_1, R_2, \cdots, R_G\})}{\text{std}(\{R_1, R_2, \cdots, R_G\})} \quad (4)$$

where

- $\mathcal{J}_{\text{TGRPO}}(\theta)$: Represents the objective function for the policy model with parameters θ.
- $\mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)}$: Denotes the expectation over questions q sampled from the distribution $P(Q)$ and groups of outputs $\{o_i\}_{i=1}^{G}$ sampled from the old policy $\pi_{\theta_{\text{old}}}$ given q.
- G: The number of generated outputs in a group.
- $\pi_\theta(o_i|q)$: The probability of generating output o_i given question q under the current policy π_θ.
- $\pi_{\theta_{\text{old}}}(o_i|q)$: The probability of generating output o_i given question q under the old policy $\pi_{\theta_{\text{old}}}$.
- π_{ref}: A fixed reference policy, often the SFT model or an initial pre-trained model, used for the KL divergence regularization.
- R_i: The reward for the output o_i.
- A_i: The advantage for output o_i.
- ϵ: A hyperparameter for the PPO-style clipping.
- β: A hyperparameter controlling the strength of the KL divergence penalty.
- $\mathbb{D}_{\text{KL}}(\pi_\theta||\pi_{\text{ref}})$: The KL divergence between the current policy π_θ and a reference policy π_{ref}.

Reward Function

Our training framework incorporates a composite reward function, combining several distinct reward components to guide the model's learning process:

Format Reward. The Format Reward encourages the model to generate outputs that adhere to a predefined structural or syntactical format. Specifically, we check for the presence of six key fields: '## Mathematical Model:', '## Decision Variables:', '## Objective Function:', '## Constraints:', '## Python Code Solution Using `coptpy`:', and '```python'. The reward is calculated as the proportion of these fields successfully identified in the output:

$$R_{\text{format}}(o_i) = \frac{\text{Number of required fields found}}{6} \quad (5)$$

Valid-Code Reward. The Valid-Code Reward incentivizes the generation of executable or syntactically correct code, crucial for tasks involving code generation or problem-solving through programmatic means. This is a binary reward:

$$R_{\text{code}}(o_i) = \begin{cases} 1, & \text{if code can correctly call `coptpy`} \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

Majority Voting Reward. The Majority Voting Reward is derived from the Test-Time Reinforcement Learning (TTRL) framework (Zuo et al. 2025), where a consensus output is established through majority voting among multiple candidate generations. This estimated consensus output then serves as a proxy label to compute a rule-based reward. The reward function is defined as:

$$R_{\text{voting}}(y_i, y) = \begin{cases} 1, & \text{if } y_i = y \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

where:

- y_i denotes the optimal solution by executing the code.
- y denotes the majority-voted prediction or the consensus output. For majority voting, only results that produce normal values from code execution are considered, ignoring cases with 'No Best Solution' or 'None' results.

The final composite reward R is the sum of these individual reward components:

$$R_i = R_{\text{format}}(o_i) + R_{\text{code}}(o_i) + R_{\text{voting}}(y_i, y) \quad (8)$$

Experiments

To analyze the performance of OR-R1, we conduct experiments based on the open-source LLM Qwen3-8B and compare it with various learning-based methods on extensive datasets. The experiments aim to answer three questions:

- **RQ1:** How does OR-R1 perform compared to existing learning-based methods across diverse real-world operations research benchmarks?
- **RQ2:** What is the impact of the TGRPO algorithm on model performance?
- **RQ3:** How do different reward formulations affect the performance of OR-R1?

Experimental Setup

Datasets. To thoroughly evaluate the capabilities of OR-R1, we utilize a diverse set of optimization-related benchmarks that are collected from previously published sources. These datasets are chosen because they present unique challenges and comprehensively cover various aspects of optimization problem solving. The specific characteristics and sizes of each key test set are based on the operations and implementations referenced from the LLMOPT. These benchmarks collectively provide a comprehensive evaluation of LLMs in optimization modeling and solving. **NL4Opt** (Ramamonjison et al. 2023) offers 230 linear programming word problems, including an 'objective' domain for generalization assessment. **Mamo** (Huang et al. 2024b) challenges

Model	NL4OPT	MAMO EasyLP	MAMO ComplexLP	IndustryOR	NLP4LP	ComplexOR	OptiBench	ICML Competition	AVG
Base Model Variants									
Qwen3-8B SFT(3K)	86.0±2.0	87.0±1.0	39.9±3.0	33.0±1.0	82.9±0.5	40.7±6.4	61.4±1.2	85.8±2.0	64.6±1.2
Qwen2.5-7B SFT(3K)	83.0±1.9	85.6±0.7	37.3±1.2	32.7±0.6	80.0±0.7	40.7±3.2	57.0±1.6	79.2±1.8	61.9±1.2
Llama3-8B SFT(3K)	80.3±3.6	81.7±2.1	32.2±2.6	24.7±3.1	78.5±1.8	37.0±8.5	54.4±2.2	77.1±1.3	58.2±1.7
Qwen3-8B SFT(100)	81.8±1.7	84.4±3.3	31.9±3.1	29.3±4.2	78.2±0.9	35.2±3.2	56.2±2.0	79.4±2.8	59.5±1.7
Learning Base Model									
LLMOPT(Qwen2.5-14B)	80.3	**89.5**	44.1	29.0	73.4	35.3	53.8	75.3	60.1
ORLM(Llama3-8B)	86.9	81.6	39.3	32.0	82.0	**50.0**	56.5	79.3	63.5
Proposed Method									
OR-R1 SFT(100)-TGRPO	88.0±0.7	87.4±2.5	45.7±8.5	30.3±3.1	84.0±1.0	46.3±8.5	61.2±0.7	84.1±1.5	65.9±2.2
OR-R1 SFT(3K)-TGRPO	**88.3±0.9**	86.1±1.0	**49.9±15.0**	**35.3±2.9**	**84.6±0.8**	46.3±3.2	**62.9±1.0**	**88.3±1.8**	**67.7±2.7**

Table 1: Main evaluation results on eight operations research benchmarks. Solution accuracy (%) is reported for each method, with the overall average (AVG) in the last column. Bold indicates the best result. Values with '±' represent the mean and standard deviation over three independent training.

LLMs with 652 **Easy LP** and 211 **Complex LP** instances requiring deeper mathematical reasoning. **NLP4LP** (AhmadiTeshnizi, Gao, and Udell 2024) features 242 richly annotated linear and mixed-integer linear programming problems, mitigating data leakage. **ComplexOR** (Xiao et al. 2023) presents 18 complex real-world operation research problems with implicit constraints and domain-specific knowledge requirements. **IndustryOR** (Huang et al. 2025) assesses performance on 100 real-world operation research problems. **OptiBench** (Yang et al. 2024b) includes 605 optimization modeling word problems across linear, non-linear, and tabular data types, evaluating iterative optimization. Finally, the **ICML Competition** (Yang et al. 2024a) track focuses on automated optimization problem solving, using 410 evaluable data points from its public leaderboard.

Training DataSets. The training of OR-R1 primarily involves two critical phases: Supervised Fine-Tuning (SFT) and a TGRPO-based reinforcement learning stage.

- **SFT stage:** We utilize **ORInstruct**, a public dataset from ORLM. It contains 3,000 synthetic samples, which corresponds to 1/10 of the full dataset used in ORLM.
- **TGRPO stage:** We used unlabeled test data for training.

Base Model. Our base model is Qwen3-8B (Yang et al. 2025a). It's chosen for its robust general-purpose language understanding and generation capabilities, demonstrating strong performance in both Math and Coding Tasks. Furthermore, its model size is comparable to that of previous related work, making it a suitable choice for our scenario.

Baselines. We evaluate our method against several strong baselines, including both general LLMs and specialized optimization-focused models. Our primary baseline is the Qwen3-8B model fine-tuned on the ORInstruct(3K) dataset. Also, we compare with the following methods:

- **ORLM (Huang et al. 2025):** This framework focuses on training open-source Large Language Models (LLMs) for optimization modeling and solver code development. It uses a semi-automated data synthesis framework called ORInstruct to generate high-quality training data from seed industry cases. The authors demonstrate results using Llama3-8B (Dubey et al. 2024) as their base model.
- **LLMOPT (Jiang et al. 2024a):** This framework employs multi-instruction tuning to improve problem formalization and solver code generation accuracy. It uses a 'five-element formulation' to define optimization problems and leverages data augmentation with expert / GPT4-based data labeling (Achiam et al. 2023). The framework incorporates supervised fine-tuning, model alignment, and a self-correction mechanism, with their primary results based on Qwen2.5-14B (Yang et al. 2025b).
- **Base Model Variants:** To ensure comprehensive comparison across different model architectures and sizes, we also include SFT-tuned versions of Qwen3-8B, Qwen2.5-7B (Yang et al. 2025b) and Llama3-8B (Dubey et al. 2024) in our baseline evaluation. This helps isolate the impact of our methodology from the inherent capabilities of different base models.

Evaluation Metrics. Aligned with ORLM, we adopt **Solution Accuracy** as our primary evaluation metric. For each optimization problem, the evaluation follows an end-to-end process: the LLM generates a response in natural language, which contains code blocks marked by ```python ... ```. A script automatically extracts and executes the generated Python code to obtain the predicted optimal objective value y_i for problem i. Let y_i^* denote the ground truth optimal value. We define **Solution Accuracy** as:

$$\text{Solution Accuracy} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(y_i = y_i^*)$$

where N is the total number of problems, and $\mathbb{I}(\cdot)$ is the indicator function, which equals 1 if the condition holds and

0 otherwise. In other words, a problem is considered correctly solved if and only if the predicted optimal value exactly matches the ground truth optimal value. This metric directly reflects the model's end-to-end capability for generating and solving optimization problems.

Implementation Details. The training process consists of two stages: SFT and TGRPO. In the SFT stage, Qwen3-8B is fine-tuned using the AdamW optimizer, a warmup-decay scheduler, and standard settings. In the TGRPO stage, the SFT output is further optimized using AdamW with a cosine scheduler and PEFT (LoRA). Both stages are trained on 4×A100 (40G) GPUs with BF16 precision. For all hyperparameters and detailed settings, please refer to the Appendix.

Main Results

Based on the experimental results (Table 1), our method demonstrates significant advantages across different test sets. Specifically, OR-R1 SFT(3K)-TGRPO achieves an average accuracy of 67.7%, substantially outperforming all baseline models and achieving optimal performance on multiple test sets including NL4OPT, MAMO ComplexLP, IndustryOR, NLP4LP, OptiBench, and ICML Competition. Notably, even with only 100 samples for SFT, OR-R1 SFT(100)-TGRPO still achieves an average accuracy of 65.9%, surpassing other baseline methods including ORLM and LLMOPT, which highlights the effectiveness of the TGRPO method. By comparing different base models, we find that Qwen3-8B, after SFT with 3K data, shows superior performance compared to Qwen2.5-7B and Llama3-8B. While this validates the importance of advanced base models, our TGRPO method can further improve performance by 3.1%-6.4%, demonstrating its effectiveness in optimizing operations research modeling and solving tasks.

Training Dynamics of TGRPO

Figure 3 illustrates the training dynamics of the three core reward components (format, valid-code, and majority voting) for SFT(3K)-TGRPO. **Format Reward:** The format reward remains consistently high (above 0.98) throughout training, indicating that the model quickly learns to generate outputs in the correct format and maintains this ability stably. **Valid-Code Reward:** The valid-code reward shows a clear upward trend in the early stages and gradually stabilizes around 0.9. This suggests that the model becomes increasingly capable of producing syntactically valid code as training progresses. **Majority Voting Reward:** The majority voting reward starts lower (around 0.7) but steadily improves, stabilizing near 0.8. This reflects the model's enhanced ability to generate solutions that are favored by majority voting, i.e., more frequently correct or consensus answers. Across all three plots, the shaded regions indicate significant variance in reward values during training, but the overall trajectories of the moving averages are positive and stable. This demonstrates that SFT(3K)-TGRPO effectively optimizes all three reward components, leading to better structured, more valid, and more reliable model outputs.

Our motivation for developing TGRPO stems from the observed significant gap (13%) between multiple sampling

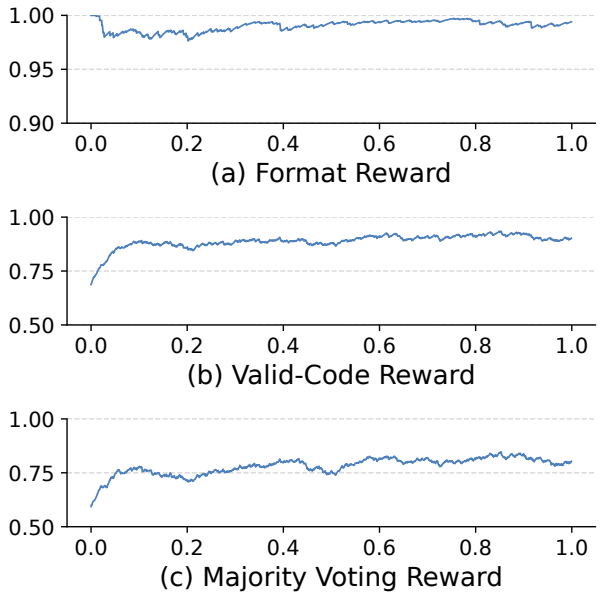

Figure 3: Overview of training dynamics for OR-R1 core reward components of SFT(3K)-TGRPO.

attempt(Pass@8) and single attempt(Pass@1) performance. While Pass@8 achieves impressive accuracy, Pass@1 initially performs substantially lower. This discrepancy indicates that the model possesses the underlying capability but lacks consistency in single-attempt scenarios. TGRPO was specifically designed to address this challenge by improving the model's deterministic performance and enhancing output consistency. As demonstrated in Fig. 4, TGRPO shows promising results in achieving this goal. After training, we successfully reduced this gap to 7%. Pass@1 shows consistent improvement throughout the training process, indicating the effectiveness of our approach in enhancing single-

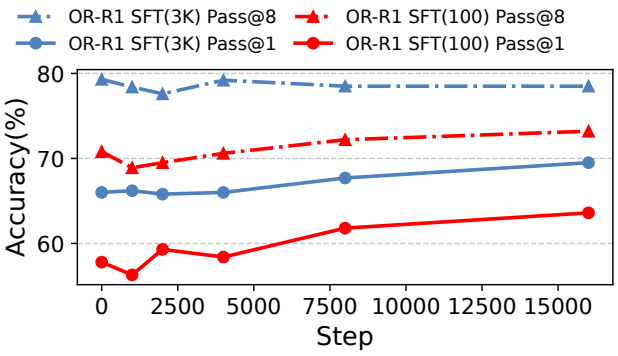

Figure 4: Performance of Pass@1 and Pass@8 during TGRPO Training. Pass@1 measures the accuracy when only the model's top prediction is considered, while Pass@8 reflects the probability that at least one out of the top 8 generated solutions is correct.

Model	NL4OPT	MAMO EasyLP	MAMO ComplexLP	IndustryOR	NLP4LP	ComplexOR	OptiBench	ICML Competition	AVG
Base Model Variants									
Qwen3-8B SFT(3K)	86.9	87.2	42.3	34.0	83.4	44.4	62.1	87.6	66.0
Ablation Studies (Base Model Qwen3-8B SFT(3K))									
$+RL(R_{format})$	$86.5_{\downarrow 0.4}$	$87.1_{\downarrow 0.1}$	$46.9_{\uparrow 4.6}$	$34.0_{\downarrow 0.0}$	$83.5_{\uparrow 0.1}$	$44.4_{\downarrow 0.0}$	$62.6_{\uparrow 0.5}$	$87.3_{\downarrow 0.3}$	$66.5_{\uparrow 0.5}$
$+RL(R_{code})$	$86.5_{\downarrow 0.4}$	**87.4**$_{\uparrow 0.2}$	$52.6_{\uparrow 10.3}$	$\underline{38.0}_{\uparrow 4.0}$	$84.3_{\uparrow 0.8}$	$38.9_{\downarrow 5.5}$	$\underline{63.8}_{\uparrow 1.7}$	$86.3_{\downarrow 1.3}$	$67.2_{\uparrow 1.2}$
$+RL(R_{voting})$	**89.0**$_{\uparrow 2.1}$	$87.0_{\downarrow 0.2}$	$58.3_{\uparrow 16.0}$	$33.0_{\downarrow 1.0}$	$84.7_{\uparrow 1.3}$	$38.9_{\downarrow 5.5}$	$62.8_{\uparrow 0.7}$	**90.5**$_{\uparrow 2.9}$	$68.0_{\uparrow 2.0}$
$+RL(R_{format}+R_{code})$	$86.5_{\downarrow 0.4}$	$86.8_{\downarrow 0.4}$	$55.0_{\uparrow 12.7}$	$35.0_{\uparrow 1.0}$	$84.7_{\uparrow 1.3}$	$44.4_{\downarrow 0.0}$	$63.1_{\uparrow 1.0}$	$86.6_{\downarrow 1.0}$	$67.8_{\uparrow 1.8}$
$+RL(R_{format}+R_{voting})$	$87.8_{\uparrow 0.9}$	$\underline{87.3}_{\uparrow 0.1}$	$51.7_{\uparrow 9.4}$	$33.0_{\downarrow 1.0}$	**86.0**$_{\uparrow 2.6}$	**50.0**$_{\uparrow 5.6}$	**64.0**$_{\uparrow 1.9}$	$88.5_{\uparrow 0.9}$	$68.5_{\uparrow 2.5}$
$+RL(R_{code}+R_{voting})$	$87.3_{\uparrow 0.4}$	$87.1_{\downarrow 0.1}$	$\underline{67.8}_{\uparrow 25.5}$	**41.0**$_{\uparrow 7.0}$	$\underline{85.5}_{\uparrow 2.1}$	$38.9_{\downarrow 5.5}$	$63.3_{\uparrow 1.2}$	$87.6_{\downarrow 0.0}$	$\underline{69.7}_{\uparrow 3.7}$
$+RL(R_{format}+R_{code}+R_{voting})$	$\underline{88.6}_{\uparrow 1.7}$	$87.0_{\downarrow 0.2}$	**66.8**$_{\uparrow 24.5}$	$37.0_{\uparrow 3.0}$	$84.3_{\uparrow 0.9}$	**50.0**$_{\uparrow 5.6}$	$63.5_{\uparrow 1.4}$	$\underline{88.8}_{\uparrow 1.2}$	**70.8**$_{\uparrow 4.8}$

Table 2: Ablation study on the reward formulations of OR-R1, using Qwen3-8B SFT(3K) as the base model. We report solution accuracy (%) on eight operations research benchmarks and the overall average (AVG). Each row shows the effect of adding different reward components either individually or in combination. Performance gains or drops relative to the SFT-only baseline are marked in red (better) and blue (worse).

generation reliability. It should be noted that the performance curve maintains an upward trajectory even at the end of our training iterations. Due to computational resource constraints, we had to limit the training duration. However, the steady positive trend suggests potential for improvements with extended training, as the model has not yet reached a performance plateau.

Data Scale Effect on TGRPO

Figure 5 illustrates how model accuracy changes with different data scales for TGRPO training. As the number of TGRPO training samples (N) increases from 10 to 50, accuracy steadily improves from 66.0% to 69.1%. However, further increasing the data scale to include all available samples does not lead to higher accuracy. This suggests that TGRPO achieves significant performance gains with a relatively small amount of data, and increasing the data size further brings diminishing returns. In summary, TGRPO not only eliminates the need for additional labeled data, but also achieves strong training performance with only a small amount of in-domain data, showing its high data efficiency.

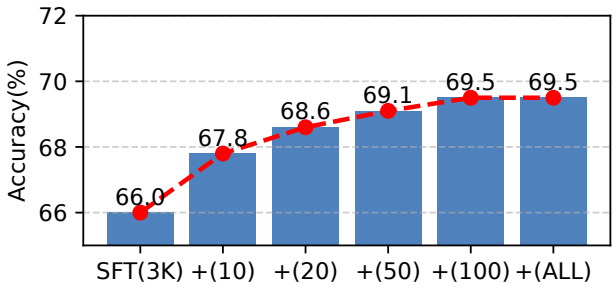

Figure 5: The impact of different data scales on TGRPO performance. In "+(N)", N denotes the number of samples randomly selected from each test set for TGRPO training. Notably, all models here were trained for 160 steps.

Ablation Studies

Table 2 summarizes the impact of different reward components in OR-R1 using Qwen3-8B SFT(3K) as the base model. Adding individual rewards, such as format, code, or voting, each brings some improvement over the baseline of SFT only, with the voting reward showing the largest gain of a single component. Combining rewards further boosts performance: the best results are achieved when all three components are used together, yielding an average accuracy of 70.8% (+4.8% over baseline). These findings show that the rewards are complementary and that a comprehensive reward design is key to maximizing model performance.

Conclusion

This study introduces OR-R1, a data-efficient training framework for solving Operations Research (OR) optimization problems. By integrating Test-Time Group Relative Policy Optimization (TGRPO), OR-R1 achieves state-of-the-art performance, with an average accuracy of 67.7% across multiple benchmarks, surpassing established methods like ORLM and LLMOPT. Remarkably, OR-R1 reaches competitive performance using only 100 labeled samples in the SFT stage, demonstrating its ability to drastically reduce data requirements while maintaining high accuracy. By leveraging tailored rewards, OR-R1 improves single-attempt reliability (Pass@1) and narrows the gap to multi-attempt performance (Pass@8) from 13% to 7%, ensuring consistent and robust outputs. This framework offers a scalable and cost-effective solution for training domain-specific large language models, which could benefit automated optimization applications in real-world industrial scenarios.

Acknowledgments

We thank the reviewers for their helpful comments and feedback that improved the final version of this article. Z. Ding and J. Zhang was supported by the Hong Kong RGC Theme-based Research Scheme T32-615-24-R/ER.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

AhmadiTeshnizi, A.; Gao, W.; and Udell, M. 2024. Optimus: Scalable optimization modeling with (mi) lp solvers and large language models. *arXiv preprint arXiv:2402.10172*.

Ahn, J.; Verma, R.; Lou, R.; Liu, D.; Zhang, R.; and Yin, W. 2024. Large Language Models for Mathematical Reasoning: Progresses and Challenges. arXiv:2402.00157.

Bairi, R.; Sonwane, A.; Kanade, A.; C, V. D.; Iyer, A.; Parthasarathy, S.; Rajamani, S.; Ashok, B.; and Shet, S. 2024. Codeplan: Repository-level coding using llms and planning. *Proceedings of the ACM on Software Engineering*, 1(FSE): 675–698.

Besta, M.; Blach, N.; Kubicek, A.; Gerstenberger, R.; Podstawski, M.; Gianinazzi, L.; Gajda, J.; Lehmann, T.; Niewiadomski, H.; Nyczyk, P.; et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 17682–17690.

Bretthauer, K. M.; and Shetty, B. 1995. The nonlinear resource allocation problem. *Operations research*, 43(4): 670–683.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Brucker, P.; Drexl, A.; Möhring, R.; Neumann, K.; and Pesch, E. 1999. Resource-constrained project scheduling: Notation, classification, models, and methods. *European journal of operational research*, 112(1): 3–41.

Chaudhary, S. 2023. Code Alpaca: An Instruction-following LLaMA model for code generation. https://github.com/sahil280114/codealpaca.

Chen, X.; Aksitov, R.; Alon, U.; Ren, J.; Xiao, K.; Yin, P.; Prakash, S.; Sutton, C.; Wang, X.; and Zhou, D. 2023. Universal Self-Consistency for Large Language Model Generation. arXiv:2311.17311.

Christiano, P. F.; Leike, J.; Brown, T.; Martic, M.; Legg, S.; and Amodei, D. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The llama 3 herd of models. *arXiv e-prints*, arXiv–2407.

Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; Bi, X.; et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Harrison, A.; Van Hoek, R.; Skipworth, H.; and Aitken, J. 2019. *Logistics management and strategy*. Pearson UK.

He, C.; Luo, R.; Bai, Y.; Hu, S.; Thai, Z. L.; Shen, J.; Hu, J.; Han, X.; Huang, Y.; Zhang, Y.; et al. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*.

Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Huang, B.; Lu, S.; Chen, W.; Wan, X.; and Duan, N. 2024a. Enhancing Large Language Models in Coding Through Multi-Perspective Self-Consistency. arXiv:2309.17272.

Huang, C.; Tang, Z.; Hu, S.; Jiang, R.; Zheng, X.; Ge, D.; Wang, B.; and Wang, Z. 2025. Orlm: A customizable framework in training large models for automated optimization modeling. *Operations Research*.

Huang, X.; Shen, Q.; Hu, Y.; Gao, A.; and Wang, B. 2024b. Mamo: a mathematical modeling benchmark with solvers. *arXiv preprint arXiv:2405.13144*.

Hurst, A.; Lerer, A.; Goucher, A. P.; Perelman, A.; Ramesh, A.; Clark, A.; Ostrow, A.; Welihinda, A.; Hayes, A.; Radford, A.; et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

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

Jiang, J.; Wang, F.; Shen, J.; Kim, S.; and Kim, S. 2024b. A Survey on Large Language Models for Code Generation. arXiv:2406.00515.

Kumar, A.; Zhuang, V.; Agarwal, R.; Su, Y.; Co-Reyes, J. D.; Singh, A.; Baumli, K.; Iqbal, S.; Bishop, C.; Roelofs, R.; et al. 2024. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*.

Lee, C.-Y.; Lee, H. L.; and Zhang, J. 2015. The impact of slow ocean steaming on delivery reliability and fuel consumption. *Transportation Research Part E: Logistics and Transportation Review*, 76: 176–190.

Li, Y.; Choi, D.; Chung, J.; et al. 2022. Competition-Level Code Generation with AlphaCode. *arXiv preprint arXiv:2203.07814*.

Long, Z.; Shimkin, N.; Zhang, H.; and Zhang, J. 2020. Dynamic scheduling of multiclass many-server queues with abandonment: The generalized cμ/h rule. *Operations Research*, 68(4): 1218–1230.

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 preprint arXiv:2502.11102*.

Mostajabdaveh, M.; Yu, T. T. L.; Dash, S. C. B.; Ramamonjison, R.; Byusa, J. S.; Carenini, G.; Zhou, Z.; and Zhang,

Y. 2025. Evaluating LLM Reasoning in the Operations Research Domain with ORQA. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 24902–24910.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744.

Prabhudesai, M.; Chen, L.; Ippoliti, A.; Fragkiadaki, K.; Liu, H.; and Pathak, D. 2025. Maximizing Confidence Alone Improves Reasoning. *arXiv preprint arXiv:2505.22660*.

Qu, Y.; Zhang, T.; Garg, N.; and Kumar, A. 2024. Recursive introspection: Teaching language model agents how to self-improve. *Advances in Neural Information Processing Systems*, 37: 55249–55285.

Rafailov, R.; Sharma, A.; Mitchell, E.; Manning, C. D.; Ermon, S.; and Finn, C. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36: 53728–53741.

Ramamonjison, R.; Yu, T.; Li, R.; Li, H.; Carenini, G.; Ghaddar, B.; He, S.; Mostajabdaveh, M.; Banitalebi-Dehkordi, A.; Zhou, Z.; et al. 2023. Nl4opt competition: Formulating optimization problems based on their natural language descriptions. In *NeurIPS 2022 Competition Track*, 189–203. PMLR.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Bi, X.; Zhang, H.; Zhang, M.; Li, Y.; Wu, Y.; et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Sheng, G.; Zhang, C.; Ye, Z.; Wu, X.; Zhang, W.; Zhang, R.; Peng, Y.; Lin, H.; and Wu, C. 2024. HybridFlow: A Flexible and Efficient RLHF Framework. *arXiv preprint arXiv: 2409.19256*.

Thind, R.; Sun, Y.; Liang, L.; and Yang, H. 2025. OptimAI: Optimization from Natural Language Using LLM-Powered AI Agents. *arXiv preprint arXiv:2504.16918*.

Wang, D.; Shelhamer, E.; Liu, S.; Olshausen, B.; and Darrell, T. 2020. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Xiao, Z.; Zhang, D.; Wu, Y.; Xu, L.; Wang, Y. J.; Han, X.; Fu, X.; Zhong, T.; Zeng, J.; Song, M.; et al. 2023. Chain-of-experts: When llms meet complex operations research problems. In *The twelfth international conference on learning representations*.

Xin, H.; Ren, Z.; Song, J.; Shao, Z.; Zhao, W.; Wang, H.; Liu, B.; Zhang, L.; Lu, X.; Du, Q.; et al. 2024. Deepseek-prover-v1. 5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. *arXiv preprint arXiv:2408.08152*.

Yang, A.; Li, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Gao, C.; Huang, C.; Lv, C.; et al. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yang, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Li, C.; Liu, D.; Huang, F.; Wei, H.; Lin, H.; Yang, J.; Tu, J.; Zhang, J.; Yang, J.; Yang, J.; Zhou, J.; Lin, J.; Dang, K.; Lu, K.; Bao, K.; Yang, K.; Yu, L.; Li, M.; Xue, M.; Zhang, P.; Zhu, Q.; Men, R.; Lin, R.; Li, T.; Tang, T.; Xia, T.; Ren, X.; Ren, X.; Fan, Y.; Su, Y.; Zhang, Y.; Wan, Y.; Liu, Y.; Cui, Z.; Zhang, Z.; and Qiu, Z. 2025b. Qwen2.5 Technical Report. arXiv:2412.15115.

Yang, Z.; Huang, Y.; Shi, W.; Feng, L.; Song, L.; Wang, Y.; Liang, X.; and Tang, J. 2024a. Benchmarking LLMs for Optimization Modeling and Enhancing Reasoning via Reverse Socratic Synthesis. *arXiv preprint arXiv:2407.09887*.

Yang, Z.; Wang, Y.; Huang, Y.; Guo, Z.; Shi, W.; Han, X.; Feng, L.; Song, L.; Liang, X.; and Tang, J. 2024b. OptiBench meets ReSocratic: Measure and improve LLMs for optimization modeling. *arXiv preprint arXiv:2407.09887*.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36: 11809–11822.

Yu, Q.; Zhang, Z.; Zhu, R.; Yuan, Y.; Zuo, X.; Yue, Y.; Fan, T.; Liu, G.; Liu, L.; Liu, X.; et al. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Zhang, B.; and Luo, P. 2025. Or-llm-agent: Automating modeling and solving of operations research optimization problem with reasoning large language model. *arXiv preprint arXiv:2503.10009*.

Zhang, C.; Sheng, G.; Liu, S.; Li, J.; Feng, Z.; Liu, Z.; Liu, X.; Jia, X.; Peng, Y.; Lin, H.; et al. 2024. A Framework for Training Large Language Models for Code Generation via Proximal Policy Optimization. In *NL2Code Workshop of ACM KDD*.

Zhao, W. X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; Du, Y.; Yang, C.; Chen, Y.; Chen, Z.; Jiang, J.; Ren, R.; Li, Y.; Tang, X.; Liu, Z.; Liu, P.; Nie, J.-Y.; and Wen, J.-R. 2025. A Survey of Large Language Models. arXiv:2303.18223.

Zuo, Y.; Zhang, K.; Sheng, L.; Qu, S.; Cui, G.; Zhu, X.; Li, H.; Zhang, Y.; Long, X.; Hua, E.; et al. 2025. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*.