From Expert Knowledge to Optimization Modeling: Prototype-Based Data Synthesis and Logical Reinforcement Learning

Bo Hu[1] **Qingcan Kang**[2*] **Qian Liu**[1] **Mingxuan Yuan**[2]
[1]Department of Industrial Engineering and Decision Analytics,
Hong Kong University of Science and Technology
[2]Huawei Noah's Ark Lab, Hong Kong
bhuan@connect.ust.hk, qianliu@ust.hk
{kangqingcan, Yuan.Mingxuan}@huawei.com

Abstract

Existing approaches to optimization modeling using large language models (LLMs) treat tasks through from-scratch construction, generating variables, objectives, and constraints without leveraging reusable structural knowledge. However, in real-world scenarios, expert modelers typically begin by identifying the canonical type or core structure of a problem, followed by iterative refinements based on specific task requirements. To incorporate this expert incremental logic, we propose a novel two-stage framework. In the first stage, we conduct Logic-Anchored Data Synthesis starting from 17 canonical prototypes, preserving critical bottleneck constraints and generating synthetic problem–formulation pairs. These pairs, together with the OR-Instruct dataset, are used for supervised fine-tuning (SFT) to initialize a policy. Separately, for each generated problem, we produce multiple modeling trajectories that follow a prototype-grounded order with variables preceding expressions. These trajectories are ranked according to criteria of accuracy and efficiency, and subsequently used to train a Logical Reward Model (LRM). In the second stage, we introduce Logic-Test-Time Group Relative Policy Optimization (Logic-TGRPO), which leverages the LRM during test-time reinforcement learning to reward accurate prototype identification and disciplined structural adherence while penalizing illogical patterns. Evaluated across seven benchmarks, our 8B parameter model achieves an average accuracy of 80.6%, outperforming comparably scaled baselines and competing effectively with heavyweight multi-agent systems with far fewer parameters. Performance remains competitive across benchmarks with diverse narrative sources. Expert modeling knowledge, represented through structural prototypes and logic-aware rewards, improves automated optimization modeling. Ablations show that the synthetic data and both logic-aware rewards each contribute to the overall gains, with the full framework achieving the highest average accuracy.

1 Introduction

Optimization Modeling (OM) translates unstructured decision problems into formal mathematical formulations, serving as the analytical backbone of operations research (OR). Bridging ambiguous natural language descriptions to executable solver code remains challenging, traditionally requiring deep expertise. Research on automating OM with LLMs has evolved through two streams: the

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

first comprises prompt-based methods [Ramamonjison et al., 2023, Li et al., 2023] and multi-agent frameworks [Xiao et al., 2024, AhmadiTeshnizi et al., 2024], which guide LLMs without parameter updates but often struggle with implicit constraints or incur high communication overhead; the second adopts data-centric fine-tuning [Huang et al., 2025a, Wu et al., 2025, Ding et al., 2026], which internalizes domain knowledge and represents the state of the art.

Despite these advances, a critical gap persists: existing approaches treat modeling as *from-scratch construction* — they parse the problem description and generate variables, constraints, and objectives without leveraging pre-existing structural knowledge. This contrasts with human expert cognition. It is a common practice among optimization modeling experts to first classify a problem into a canonical family (e.g., knapsack, transportation), each with a *bottleneck constraint* or core skeleton, then inherit that skeleton and incrementally add or modify constraints based on the context. This *incremental modeling* strategy is both fast and accurate, yet it has been largely overlooked.

To bridge this gap, we propose a framework that injects human expertise from two complementary angles, both grounded in the incremental modeling paradigm. First, we design a Logic-Anchored Data Synthesis pipeline that starts from 17 canonical OR prototypes grounded in OptiMUS-0.3 [AhmadiTeshnizi et al., 2024], with small adaptations. The pipeline preserves the core bottleneck skeleton of each domain while varying only secondary elements to maintain structural integrity. To ensure logical consistency, we enforce a prototype-grounded reasoning order: variables must be declared before any objective functions or constraints. However, we allow flexibility in the subsequent sequence, permitting either a variables-objective-constraints or variables-constraints-objective flow. For each generated problem instance, we produce multiple solution trajectories and rank them by a preference criterion: correct and fast (short runtime) receive the highest rank, correct but slower receive a medium rank, and incorrect pathways receive the lowest rank. The resulting data support two training objectives: (i) synthetic problem–formulation pairs, together with the existing OR-Instruct dataset [Huang et al., 2025a], are used for SFT to initialize the policy; (ii) preference-ranked modeling trajectories are used to train the LRM to assess structural quality, efficiency, and prototype adherence. Importantly, during LRM training, we require the model to explicitly output a prototype label (e.g., `[PROTOTYPE:Packaging]`) within the reasoning block, and to learn to score the correctness of this identification.

Second, we introduce Logic-TGRPO, which extends OR-R1 [Ding et al., 2026] by starting from the SFT-initialized policy and using the pre-trained LRM as a reward proxy during test-time RL on unlabeled problems. The overall reward provides preference-based rewards that evaluate the following aspects: (i) whether the identified prototype is correct, (ii) whether the modeling trajectory follows disciplined order (variables first, then objective/constraints in either order), and (iii) whether the final solution is correct and efficient. Trajectories that correctly identify the prototype, adhere to the incremental skeleton, and achieve short runtime receive the highest rewards; illogical patterns such as missing prototype declaration or writing objective/constraints before variables incur penalties. Thus, the RL phase directly reinforces the expert's first step — recognizing the prototype — along with the subsequent incremental derivation, without requiring ground-truth answers. Experiments on seven OR benchmarks (NL4Opt, MAMO EasyLP, MAMO ComplexLP, NLP4LP, ComplexOR, IndustryOR, ReSocratic) demonstrate that our 8B model achieves 80.6% average accuracy, outperforming equivalently scaled baselines and rivaling heavyweight multi-agent systems with far fewer parameters. Ablation studies show that the synthetic data and both logic-aware rewards each improve average accuracy across the seven benchmarks, and the full framework achieves the best overall performance.

Our key contributions are summarized as follows:

- We propose a novel **Logic-Anchored Data Synthesis** method that preserves domain-specific bottleneck skeletons and enforces a prototype-grounded reasoning order. It generates synthetic problem–formulation pairs for SFT and preference-ranked modeling trajectories for training a **Logical Reward Model (LRM)** that evaluates prototype correctness, structural order, and efficiency.
- We develop **Logic-TGRPO**, a reinforcement learning framework that starts from an SFT-initialized policy and utilizes the pre-trained LRM to provide explicit rewards for prototype identification and disciplined modeling order during test-time RL on unlabeled problems, effectively reinforcing the complete incremental modeling discipline without requiring ground-truth labels.

- We demonstrate that our **8B parameter model** achieves 80.6% average accuracy across seven OR benchmarks, the highest reported average among the compared single-pass models of similar parameter scale (7B–14B).

2 Related Work

2.1 LLMs for Optimization Modeling

Research on automating OM with LLMs falls into two streams: prompt engineering and multi-agent collaboration [Ramamonjison et al., 2023, Li et al., 2023, AhmadiTeshnizi et al., 2024, Xiao et al., 2024, Mostajabdaveh et al., 2025, Thind et al., 2025, Zhang et al., 2025], and fine-tuning on curated OR datasets [Huang et al., 2025a, Jiang et al., 2024, Yang et al., 2024, Wu et al., 2025, Lu et al., 2025]. A recent survey [Xiao et al., 2025] provides a comprehensive overview. Despite their differences, these approaches share a common limitation: they treat modeling as *from-scratch construction*, generating variables, constraints, and objectives by parsing the problem description without leveraging any pre-existing structural knowledge. This contrasts with human experts, who first classify a problem into a canonical family (e.g., knapsack, transportation), identify its characteristic bottleneck constraint or core skeleton, and then incrementally adapt that skeleton to the specific context. To inject this expert *incremental modeling* logic, our framework first recognizes the problem's domain prototype and bottleneck components (informed by the 17 canonical categories from [AhmadiTeshnizi et al., 2024]), then constructs variables first, followed by objectives and constraints (in either order) accordingly, shifting from parallel extraction to top-down, causality-aware derivation.

2.2 Data Synthesis for Optimization Modeling

Prior strategies include ORLM's augmentation [Huang et al., 2025a], OptMATH's bidirectional generation [Lu et al., 2025], and MILP-Evolve's evolutionary operators [Li et al., 2024]. Benchmarks include NLP4LP [AhmadiTeshnizi et al., 2024] and IndustryOR [Huang et al., 2025a]. A common limitation is that all these methods produce flat problem–formulation pairs without preserving a canonical skeleton. Our logic-anchored synthesis instead encodes expert incremental logic. Starting from 17 canonical templates [AhmadiTeshnizi et al., 2024], we keep bottleneck elements untouched while varying only minor constraints. Applying multiple evolutionary operators cumulatively yields broad diversity and complexity. These operations yield synthetic problem–formulation pairs. Separately, for each problem, we generate multiple modeling trajectories following a prototype-grounded order with variables preceding expressions. Trajectories are ranked by preference: correct and fast (short runtime) are highest, correct but slower second, and incorrect lowest. Each trajectory includes a prototype label. The problem–formulation pairs together with OR-Instruct are used for SFT; the ranked trajectories train a LRM that assesses structural quality, efficiency, and prototype correctness.

2.3 Reinforcement Learning for LLM Reasoning and Optimization

Reinforcement learning has proven effective for aligning LLMs with human preferences. Foundational works include RLHF [Ouyang et al., 2022, Christiano et al., 2017], DPO [Rafailov et al., 2023], and PPO [Schulman et al., 2017]. DeepSeekMath [Shao et al., 2024] introduced GRPO, a value-free group-based method, and DeepSeek-R1 [Guo et al., 2025] demonstrated emergent reasoning via pure RL. For OR modeling, OR-R1 [Ding et al., 2026] first combined supervised fine-tuning with test-time GRPO, using format, code validity, and majority voting rewards. However, OR-R1 only rewards final output correctness, ignoring the structural quality and efficiency of the reasoning trajectory. Our Logic-TGRPO addresses this by starting from the SFT-initialized policy and using the pre-trained LRM as a reward proxy during test-time RL on unlabeled problems. The LRM provides dense preference-based rewards that explicitly evaluate four aspects: (i) correctness of the identified prototype; (ii) adherence to the incremental skeleton; (iii) disciplined order (variables first, then constraints/objectives in either permitted order); and (iv) computational efficiency (short runtime) and final answer correctness. Illogical patterns such as missing prototype declaration, writing constraints before variables, or failing to preserve the bottleneck skeleton incur penalties. In this way, Logic-TGRPO reinforces the complete expert incremental discipline — from prototype recognition to stepwise derivation — shifting the RL objective from output correctness alone to internalizing efficient, expert-like modeling expertise.

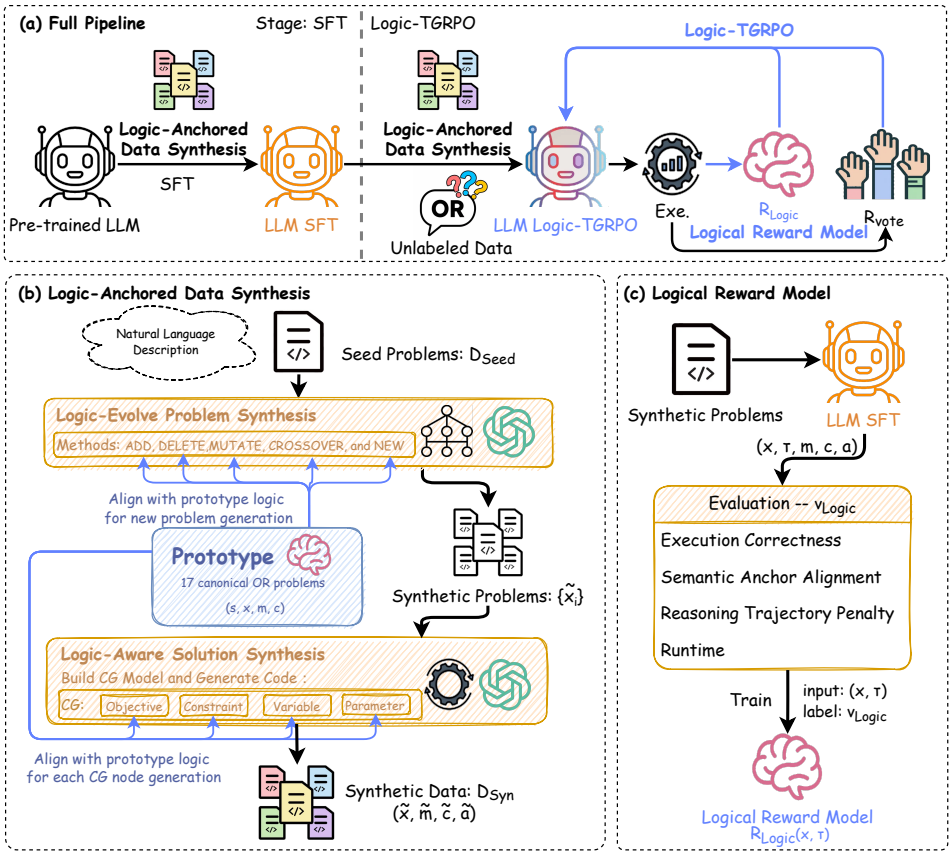

Figure 1: Overall framework of our proposed method. **(a) Full Pipeline**: The training workflow where the base LLM undergoes Supervised Fine-Tuning (SFT) followed by Logic-TGRPO optimization. **(b) Logic-Anchored Data Synthesis**: Evolutionary problem synthesis guided by expert-curated prototypes. **(c) Logical Reward Model**: A specialized evaluator that explicitly scores the intrinsic reasoning trajectory.

3 Method

3.1 Overview

To overcome data scarcity and logical inconsistencies in automated OR modeling, we propose a two-stage training framework utilizing Qwen3-8B [Yang et al., 2025] as the primary base LLM (Figure 1(a)). First, **Logic-Anchored Data Synthesis** (Figure 1(b)) leverages 17 canonical OR problem prototypes to explicitly guide both problem and solution generation, yielding a synthetic dataset for supervised fine-tuning (SFT). Second, **Logic-TGRPO** optimizes the SFT-initialized model, gated by formatting and executability checks, with additive rewards given by majority voting, continuous logical rewards, and prototype adherence rewards. Notably, the logical reward is provided by a specialized **Logical Reward Model** (Figure 1(c)), parameterized as a Qwen3-1.7B evaluator trained on execution and semantic feedback to explicitly score the intrinsic thinking trajectory. The framework incorporates expert modeling knowledge through prototype-anchored synthesis, structured supervision in SFT, and logic-aware rewards in RL.

3.2 Logic-Anchored Data Synthesis

Prototype Construction. To prevent synthetic data from degenerating into trivial or out-of-distribution variations, we construct a set $\mathcal{P}$ of 17 canonical OR prototypes grounded in OptiMUS-0.3 [AhmadiTeshnizi et al., 2024], with small adaptations, spanning diverse domains (e.g., resource allocation and network flow). Each prototype $p_k \in \mathcal{P}$ is explicitly formulated as a 4-tuple

$p_k = (s_k, x_k, m_k, c_k)$, representing the scenario category, natural language description, formal mathematical model, and executable solver code, respectively. These tuples record the canonical modeling scaffold—the structurally essential objective, variables, and constraints—serving as strict logical anchors for subsequent generation phases (see Appendix A for detailed structural analyses).

Logic-Evolve Problem Synthesis. Inspired by standard evolutionary data generation [Li et al., 2024], our problem synthesis applies stochastic operators (ADD, DELETE, MUTATE, CROSSOVER, NEW) to seed data. Crucially, each generated problem $\tilde{x}$ must strictly reference an assigned base prototype p_k. This logic-anchoring ensures that, while contextual semantics and numerical parameters evolve, the synthesized problem preserves the fundamental mathematical essence of the target OR class rather than drifting into superficially similar but structurally invalid variants.

Logic-Aware Solution Synthesis. For solution generation, we build upon the Connection Graph (CG) representation of OptiMUS [AhmadiTeshnizi et al., 2024], augmenting the CG with explicit logic conditioning from the matched prototype p_k. The constraint and variable extractors must cross-reference entities in p_k: required terms existing in the prototype are directly adopted, whereas novel terms require explicitly generated justifications. This comparative synthesis reflects the top-down cognitive process of OR experts, who validate new formulations against canonical prototypes before adapting them. We further enrich the CG by injecting *Related Constraints* and *Role in Objective* properties into corresponding nodes, yielding a fully annotated data tuple $(\tilde{x}, \tilde{m}, \tilde{c}, \tilde{a})$ comprising the synthesized problem, formal model, executable code, and generated answer.

3.3 Supervised Fine-Tuning (SFT) Phase

The SFT phase aims to instill foundational reasoning patterns and domain-specific formatting into the base LLM policy π_θ, parameterized by θ. Using a mixture of the OR-Instruct dataset $\mathcal{D}_{\text{OR-Instruct}}$ and our synthetic dataset $\mathcal{D}_{\text{syn}} = \{(\tilde{x}, \tilde{m}, \tilde{c}, \tilde{a})\}$, we optimize the standard autoregressive negative log-likelihood loss $\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(q,y)\sim\mathcal{D}_{\text{OR-Instruct}}\cup\mathcal{D}_{\text{syn}}}\left[\sum_{t=1}^{|y|}\log \pi_\theta(y_t \mid q, y_{<t})\right]$, where q corresponds to the input problem prompt (representing raw input in $\mathcal{D}_{\text{OR-Instruct}}$ and $\tilde{x}$ in $\mathcal{D}_{\text{syn}}$), $y = (y_1, \ldots, y_{|y|})$ denotes the target textual response formed by concatenating the mathematical model $\tilde{m}$ and executable code $\tilde{c}$, and $y_{<t} = (y_1, \ldots, y_{t-1})$ denotes the response prefix preceding step t.

3.4 Logic-TGRPO

To enhance logical coherence without costly human annotations, we apply a modified TGRPO [Ding et al., 2026] using a training pool $\mathcal{D}_{\text{unlabeled}}$. Standard GRPO samples G textual outputs $\{o_i\}_{i=1}^{G}$ for a query $x \sim \mathcal{D}_{\text{unlabeled}}$ and updates π_θ using normalized relative advantages. However, because our hard-filtering mechanism bounds the reward space to a stable interval, we discard advantage normalization and directly optimize the bounded raw composite rewards R_i, stabilizing training without requiring a separate value network.

Logical Reward Model. To evaluate the thinking trajectory τ without rigidly enforcing a single rule-based template, we employ an LLM-as-a-judge framework with a Logical Reward Model $f_\phi(x, \tau)$ parameterized by ϕ, via the Qwen3-1.7B. The trajectory τ is parsed into four semantic categories: $\mathcal{T} = \{$*prototype, objective, constraints, variables*$\}$. Let $s(\tau) \in \mathcal{T}$ denote the first category appearing in τ, and $C(\tau) \subseteq \mathcal{T}$ the set of covered categories. To explicitly penalize illogical top-down modeling—such as jumping to formulas before defining variables or prototypes—we define a trajectory penalty R_{traj} that penalizes structural omissions and starting with objectives or constraints:

$$R_{\text{traj}} = -\lambda_{\text{start}} \cdot \mathbb{I}(s(\tau) \in \{\textit{objective}, \textit{constraints}\}) - \lambda_{\text{omit}} \sum_{j\in\mathcal{T}} \mathbb{I}(j \notin C(\tau)), \tag{1}$$

where λ_{start} and λ_{omit} are positive penalty coefficients. To train f_ϕ, we construct a continuous evaluation target v_{logic} from offline execution feedback, combining execution correctness $R_{\text{exec}} \in \{0, 1\}$, semantic alignment R_{semantic} (which assesses the semantic consistency between the generated formal model and the natural language problem), and efficiency R_{time} (which rewards shorter solver execution times). When execution fails, the reward defaults to R_{traj}:

$$v_{\text{logic}} = \mathbb{I}(R_{\text{exec}} = 1) \cdot \left(R_{\text{exec}} + R_{\text{semantic}} + R_{\text{time}}\right) + \mathbb{I}(R_{\text{exec}} = 0) \cdot R_{\text{traj}}. \tag{2}$$

Minimizing the mean squared error against v_{logic} yields the trained model f_ϕ, which provides a smooth logical reward signal $R_{\text{logic}}(x, \tau) = f_\phi(x, \tau)$ for downstream RL.

Prototype Reward. Beyond trajectory efficiency, the final formulation must faithfully capture the core mathematical skeleton of its corresponding problem family. We introduce a prototype reward $R_{\text{proto}}(x, o_i)$ evaluated via an LLM-as-a-judge. The judge first attempts to match the problem to one of the 17 canonical prototypes. If unmatched, $R_{\text{proto}} = 0$. If matched, it assesses whether the generated model in output o_i successfully captures the prototype's core structural logic, assigning a continuous score $R_{\text{proto}} \in [0, 1]$ to prevent drift into superficially plausible but malformed formulations.

Reward Function Design. Following OR-R1 [Ding et al., 2026], we retain format and code-validity checks as gating mechanisms, while our final Logic-TGRPO reward adds three reward components: voting, logic, and prototype rewards. Let $R_{\text{format}} \in [0, 1]$ measure the proportion of required formatting blocks generated, and $R_{\text{code}} \in \{0, 1\}$ denote syntactic code executability. For voting, $R_{\text{vote}}(o_i) = r_{\text{match}}$ if the generated answer $a(o_i)$ equals the majority-voted answer $\hat{a}_{\text{vote}}$, and r_{mismatch} otherwise ($r_{\text{match}} > r_{\text{mismatch}} > 0$). To prevent π_θ from exploiting fundamentally invalid outputs, a hard-filtering mechanism governed by a pass threshold γ_{pass}, a fail threshold γ_{fail}, and a failure penalty λ_{filter} yields the total composite reward R_i for output o_i (with trajectory τ_i):

$$R_i = \begin{cases} R_{\text{vote}}(o_i) + R_{\text{logic}}(x, \tau_i) + R_{\text{proto}}(x, o_i), & \text{if } R_{\text{format}}(o_i) + R_{\text{code}}(o_i) > \gamma_{\text{pass}}, \\ -\lambda_{\text{filter}}, & \text{else if } R_{\text{format}}(o_i) < \gamma_{\text{fail}}, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

This ensures dense rewards are exclusively reserved for outputs satisfying strict structural and syntactic prerequisites (see Appendices B and C for exact hyperparameter instantiations).

4 Experiments

To thoroughly analyze the performance of our proposed method, we conduct extensive experiments comparing our framework against state-of-the-art learning-based methods across comprehensive OR benchmarks. Our experiments address three questions: (i) How does our logic-anchored framework perform against existing learning-based approaches across diverse real-world OR benchmarks? (ii) How effectively does Logic-Anchored Data Synthesis improve model alignment and reasoning accuracy relative to conventional synthetic strategies? (iii) What is the specific impact of R_{logic} and R_{proto} in Logic-TGRPO on final solving capability?

4.1 Experimental Setup

Datasets. To ensure an unbiased evaluation, we source our benchmarks based on the taxonomy provided in *A Survey of Optimization Modeling Meets LLMs* [Xiao et al., 2025]. We evaluate on **NL4Opt** [Ramamonjison et al., 2023], **MAMO** [Huang et al., 2025b], **NLP4LP** [AhmadiTeshnizi et al., 2024], **ComplexOR** [Xiao et al., 2024], **IndustryOR** [Huang et al., 2025a], and **ReSocratic** [Yang et al., 2024], covering problems from simple linear programming to complex domain-specific mixed-integer programming.

Training Datasets. Our training data is strategically partitioned to maximize data efficiency. For the **SFT phase**, we use a distilled mixture of the open-source $\mathcal{D}_{\text{OR-Instruct}}$ dataset and 100 heavily curated synthetic samples from $\mathcal{D}_{\text{syn}}$ generated by our Logic-Anchored Data Synthesis pipeline. An archived SFT scale analysis shows smaller additional gains beyond 100 synthetic examples (Appendix D). For the **Logical Reward Model**, we train the trajectory evaluator f_ϕ using preference-ranked modeling trajectories generated for the synthetic problems, with preferences determined by solution correctness and execution efficiency. For the **Logic-TGRPO phase**, following OR-R1 [Ding et al., 2026], we form $\mathcal{D}_{\text{unlabeled}}$ from 100 unlabeled problem statements sampled at the same rate from the seven evaluation benchmarks (1,510 test instances in total, approximately 6.6% sampled). Because these statements are part of the evaluated test sets, this phase performs label-free transductive adaptation. Benchmark-provided gold formulations, solver code, objective values, and answer labels are excluded from RL training targets and reward signals.

Table 1: Solution accuracy (%) with frozen SFT/native-thinking results separated from label-free transductive adaptation results. Best results are **bold**; second-best are underlined.

Model & Paradigm	NL4Opt	MAMO EasyLP	MAMO ComplexLP	NLP4LP	ComplexOR	IndustryOR	ReSocratic	AVG
Zero-shot / Base Models								
Qwen3-8B	76.5	87.3	25.2	74.7	16.7	31.0	77.9	55.6
Single-pass Fine-Tuned & RL Baselines								
LLMOPT (Qwen2.5-14B)† [Jiang et al., 2024]	80.3	89.5	44.1	73.4	35.3	29.0	–	58.6
ORLM (Llama3-8B)† [Huang et al., 2025a]	86.9	81.6	39.3	82.0	50.0	32.0	–	62.0
Step-Opt-Instruct (LLaMA3-8B) [Wu et al., 2025]	84.5	85.3	61.6	–	–	36.4	–	67.0
OptMATH (Qwen2.5-7B) [Lu et al., 2025]	**94.7**	86.5	51.2	–	–	–	–	77.5
OR-R1 SFT(3K)-TGRPO (Qwen3-8B) [Ding et al., 2026]	88.3	86.1	49.9	84.6	46.3	35.3	–	65.1
Frozen SFT Variants (no test-time updates)								
Qwen3-8B SFT (w/o Synthetic Data)	85.9	93.9	45.0	87.6	44.4	45.2	78.9	68.7
Qwen3-8B SFT (Synthetic Data)	80.3	90.1	55.0	79.2	44.4	47.6	76.4	67.6
Qwen3-8B SFT (Combined)	90.1	93.0	51.4	91.0	33.3	45.2	82.1	69.4
Frozen Native-Thinking / SFT+Thinking References								
Qwen3-8B Native Thinking (no SFT/RL)	79.3	84.2	38.7	93.3	33.3	42.9	82.6	64.9
Qwen3-8B SFT + Thinking (Combined)	94.4	93.0	53.2	96.6	44.4	52.4	89.6	74.8
Label-free Transductive Test-Time Adaptation								
Logic-TGRPO (w/o Synthetic Data)	91.5	**94.3**	57.7	**98.9**	55.6	59.5	90.8	78.3
Logic-TGRPO (Full)	92.5	93.8	**66.7**	97.2	**61.1**	**61.9**	**91.3**	**80.6**

† Results are taken from OR-R1 [Ding et al., 2026]. "–" indicates results not reported in the original source.

Base Models and Variants. We initialize our framework using the open-weights Qwen3-8B model. During the Supervised Fine-Tuning (SFT) phase, the model operates in a standard generation mode. In contrast, during the Logic-TGRPO phase, we explicitly activate the "thinking mode" to elicit extended chain-of-thought reasoning before final answer generation. For reward computation, both the semantic alignment reward (R_{semantic}) and the prototype adherence reward (R_{proto}) are evaluated using GPT-5-mini as a judge. To rigorously ablate our contributions, our base model variants focus on Qwen3-8B configurations: comparing SFT on OR-Instruct only, SFT on our synthetic data only, and their combination, as well as evaluating the impact of the thinking mode.

Baselines. We categorize our baselines into two distinct paradigms: (1) *Single-pass Fine-Tuned & RL Models*: This includes LLMOPT [Jiang et al., 2024], ORLM [Huang et al., 2025a], Step-Opt-Instruct [Wu et al., 2025], OptMATH [Lu et al., 2025], and OR-R1 [Ding et al., 2026] (which utilizes TGRPO). These models, like ours, generate answers in a single inference pass without external solver loops. (2) *Multi-Agent & Tree-Search Frameworks*: This includes Chain-of-Experts (CoE) [Xiao et al., 2024], OptiMUS [AhmadiTeshnizi et al., 2024], and OptiTree [Liu et al., 2025]. These frameworks rely on iterative solver feedback, multi-round debugging, or Monte Carlo Tree Search (MCTS) powered by proprietary models (e.g., GPT-4o).

Implementation Details. We employ Low-Rank Adaptation (LoRA) [Hu et al., 2022] in both the SFT and Logic-TGRPO stages to improve computational efficiency. Training is conducted on a single modified NVIDIA RTX 4090 (48GB VRAM) GPU. Detailed reward hyperparameters and threshold values are deferred to Appendix B.

4.2 Main Results

Following standard practice [Huang et al., 2025a, Ding et al., 2026], we adopt **Solution Accuracy** as our primary metric, defined as $\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(a_i = a_i^*)$. This strictly requires the generated code to output the exact ground-truth answer a_i^*. We decouple our evaluation into two paradigms: **Table 1** compares Logic-TGRPO against equivalently scaled, single-pass foundation models, where "Combined" denotes the mixture of $\mathcal{D}_{\text{OR-Instruct}}$ and our synthetic data; subsequently, **Table 2** contrasts our lightweight framework with compute-intensive, search-based multi-agent systems.

Evaluation regimes. Frozen SFT and native-thinking models undergo no test-time parameter updates; native thinking uses the base checkpoint without SFT or RL (64.9% macro). Logic-TGRPO results are evaluated after label-free adaptation to the sampled test inputs, with no further updates during final prediction. Table 1 separates these regimes; published baselines remain external references.

Table 2: Cross-paradigm comparison against computationally intensive frameworks.

Framework	Base LLM	Inference Paradigm	NL4Opt	MAMO EasyLP	MAMO ComplexLP	ComplexOR	IndustryOR
Published GPT-4o results							
Chain-of-Experts† [Xiao et al., 2024]	GPT-4o	Multi-Agent + Code Eval	76.4	85.7	46.4	68.4	34.0
OptiMUS† [AhmadiTeshnizi et al., 2024]	GPT-4o	Modular Pipeline + Debugging	82.0	85.1	47.3	79.0	34.0
OptiTree† [Liu et al., 2025]	GPT-4o	Taxonomy Tree Search	**96.2**	**95.6**	**81.0**	**84.2**	**48.0**
Qwen3-8B under matched single-GPU wall-clock budget							
OptiTree-style Search	Qwen3-8B	Taxonomy Tree Search (budget-matched)	58.9	39.6	43.2	27.8	21.4
Logic-TGRPO (Ours)	**Qwen3-8B**	**Single-pass**	**92.5**	**93.8**	**66.7**	**61.1**	**61.9**

† GPT-4o results for Chain-of-Experts, OptiMUS, and OptiTree are taken from [Liu et al., 2025].

Table 3: Comparison of computational efficiency (average inference latency per problem in seconds).

Framework	NL4Opt	MAMO EasyLP	MAMO ComplexLP	ComplexOR	IndustryOR	Average
OptiTree (Inference) [Liu et al., 2025]	13.9	9.3	13.3	31.0	19.9	17.5
Logic-TGRPO (Ours)	**4.7**	**6.0**	**7.5**	**11.2**	**9.2**	**7.7**

4.2.1 Overcoming the SFT Bottleneck via Logic-Anchored RL

Compared to existing single-pass fine-tuned and RL baselines (Table 1), our base *SFT (Combined)* model demonstrates highly competitive performance without activating an extended thinking mode. It achieves an average accuracy of 69.4%, outperforming recent open-source baselines including Step-Opt-Instruct (67.0%) and the identically backboned OR-R1 (65.1%). Delving into the dataset specifics, comparing the *Combined* configuration with the *w/o Synthetic Data* variant confirms that our logic-anchored synthetic data substantially boosts structural generalization, yielding absolute gains of $+4.2\%$ on `NL4Opt` and $+6.4\%$ on `MAMO ComplexLP`. However, standard SFT inherently acts as behavioral cloning: while it effectively mimics frequent modeling patterns, it struggles to enforce strict logical consistency under coupled real-world constraints, creating a performance bottleneck.

To overcome this limitation, we activate the reasoning mode and apply **Logic-TGRPO** to inject continuous logical verification during reinforcement learning. The full Logic-TGRPO model reaches an outstanding average accuracy of 80.6%, under label-free transductive adaptation and yielding a $+5.8\%$ absolute improvement over the *SFT + Thinking* baseline (74.8%). By systematically penalizing logically inconsistent formulations across the reasoning trajectory, the most substantial gains precisely emerge on structurally difficult benchmarks. Logic-TGRPO attains new state-of-the-art single-pass accuracy on `MAMO ComplexLP` (66.7% vs. 53.2%), `ComplexOR` (61.1% vs. 44.4%), and `IndustryOR` (61.9% vs. 52.4%), showing improved end-task solution accuracy in this evaluation regime.

4.2.2 Bridging the Gap with Heavyweight Search-Based Frameworks

To provide a holistic view of the performance-efficiency trade-off, Table 2 contrasts our framework against computationally intensive multi-agent and tree-search systems. Frameworks such as OptiTree and OptiMUS achieve high peak accuracy (e.g., 84.2% on `ComplexOR`) by leveraging proprietary backbones like GPT-4o, taxonomy tree expansion, and iterative solver debugging. Logic-TGRPO is competitive with these heavyweight systems despite its smaller backbone and single-pass inference: GPT-4o-based OptiTree achieves higher accuracy on four of the five shared benchmarks, while Logic-TGRPO achieves higher accuracy on IndustryOR.

We establish a budget-matched OptiTree-style search baseline constrained to the identical **Qwen3-8B** backbone and a single-GPU wall-clock limit. Under these conditions, search-based inference degrades. Logic-TGRPO achieves 61.9% on `IndustryOR`, exceeding all published results compared in Table 2. OptiTree achieves 48.0% with GPT-4o, while the budget-matched Qwen3-8B search baseline achieves 21.4%. Concurrently, Logic-TGRPO proves highly efficient (Table 3): it reduces average inference latency by 56% (17.5s vs. 7.7s) and reduces latency on intricate `ComplexOR` problems by 64% (31.0s vs. 11.2s) compared to standard OptiTree. These results indicate that expert modeling knowledge, represented through structural prototypes and logic-aware rewards, improves automated optimization modeling.

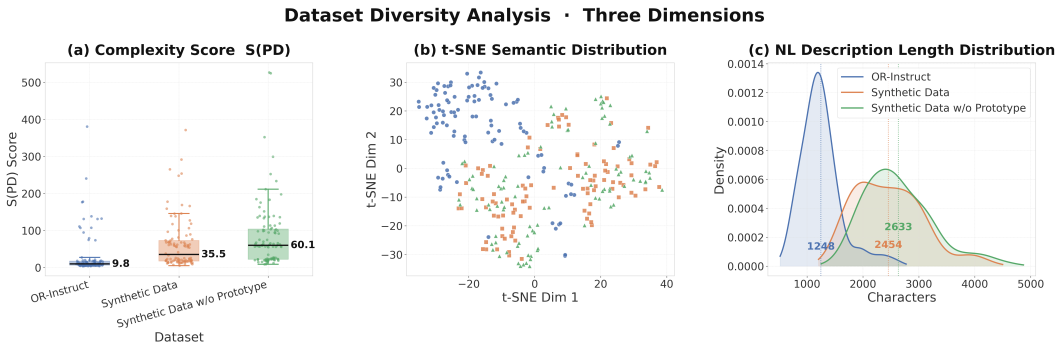

Figure 2: Intrinsic comparison of OR-Instruct, our full dual-anchor synthetic corpus, and the corpus generated without the problem-synthesis prototype while retaining the solution-synthesis prototype. **(a)** Parser-derived complexity under our project-specific instantiation of the OptMATH $S(\mathrm{PD})$ feature family [Lu et al., 2025]. **(b)** Descriptive t-SNE projection. **(c)** Kernel-density estimates of natural-language description length. Panels (b)–(c) are illustrations rather than evidence of out-of-distribution (OOD) generalization [Liu et al., 2021].

Table 4: Ablation study on the components of Logic-Anchored Data Synthesis and Logic-TGRPO. Within each block, **bold** denotes the best and underlined the second-best result.

Model Configuration	NL4Opt	MAMO EasyLP	MAMO ComplexLP	NLP4LP	ComplexOR	IndustryOR	ReSocratic	AVG
SFT Phase Ablations								
Qwen3-8B SFT (Combined)	**90.1**	93.0	**51.4**	91.0	33.3	**45.2**	**82.1**	69.4
– w/o Problem Synthesis Prototype	89.2	**94.7**	48.6	90.4	38.9	42.9	81.6	69.5
– w/o Solution Synthesis Prototype	87.8	93.0	**51.4**	88.2	**44.4**	**45.2**	81.6	**70.2**
– w/o Both Synthesis Prototypes	78.4	94.3	49.5	**91.6**	38.9	40.5	80.9	67.7
Logic-TGRPO Phase Ablations								
Logic-TGRPO (Full)	**92.5**	93.8	**66.7**	**97.2**	**61.1**	61.9	**91.3**	**80.6**
– w/o R_{proto}	91.5	92.7	64.0	96.1	**61.1**	**64.3**	89.6	79.9
– w/o R_{logic}	**92.5**	**95.2**	59.5	95.5	55.6	61.9	89.3	78.5
– w/o R_{logic} and R_{proto}	91.1	93.4	65.8	94.9	55.6	61.9	89.3	78.9

4.3 Analysis of Synthetic Data Quality

Figure 2 compares OR-Instruct, our full dual-anchor synthetic corpus, and the corpus generated without the problem-synthesis prototype while retaining the solution-synthesis prototype, with $N = 100$ in each group. Panel (a) uses our project-specific non-unit weighting of the OptMATH $S(\mathrm{PD})$ feature family [Lu et al., 2025]. The plotted median complexity scores are 9.8, 35.5, and 60.1, respectively. Under this score, the full corpus has a lower median complexity than the problem-prototype ablation. Panel (c) shows mean description lengths of $1,248$, $2,454$, and $2,633$ characters for OR-Instruct, the full synthetic corpus, and the problem-prototype ablation, respectively. The t-SNE and length-density panels provide descriptive views of these three corpora.

Quantitative distribution diagnostics. We use squared maximum mean discrepancy (MMD^2) [Gretton et al., 2012] and classifier two-sample testing (C2ST) [Lopez-Paz and Oquab, 2017] with ROC-AUC [Fawcett, 2006]. Comparing the full corpus with a separate ablation that removes both synthesis prototypes yields parser-derived structural $\mathrm{MMD}^2 = 0.018$ and C2ST ROC-AUC $= 0.612$. Across the seven benchmarks, text MMD^2 ranges from 0.053 to 0.138, and grouped C2ST yields ROC-AUC values from 0.999 to 1.000. These diagnostics quantify distribution differences in the chosen representations; they do not establish structural OOD generalization [Liu et al., 2021] or beyond-library transfer. Additional corpus statistics and complexity sensitivity analyses are reported in Appendix E.

4.4 Ablation Studies

SFT Phase. The full `Combined` configuration achieves the highest scores across four datasets (`NL4Opt`, `MAMO ComplexLP`, `IndustryOR`, and `ReSocratic`), confirming that dual-prototype anchoring establishes a robust structural foundation. However, its overall AVG (69.4%) is lower than the single-prototype ablations (69.5% and 70.2%), with `ComplexOR` at 33.3%. The prototype ablations

show task-dependent contributions: partial removals yield mixed, domain-specific fluctuations, while removing *both* prototypes lowers `NL4Opt` to 78.4% and gives the lowest AVG in the phase (67.7%). These results indicate partial redundancy between the two anchors.

Logic-TGRPO Phase. The full Logic-TGRPO framework achieves the best results on `NL4Opt`, `MAMO ComplexLP`, `NLP4LP`, and `ReSocratic`, and has the highest overall AVG (80.6%). Ablations reveal that R_{logic} and R_{proto} provide complementary inductive biases. Without R_{logic}, accuracy declines on hard benchmarks (e.g., `MAMO ComplexLP` drops by 7.2%), and the AVG is 78.5%. Removing R_{proto} matches Full on `ComplexOR` (61.1%), improves `IndustryOR`, and lowers the AVG to 79.9%; removing both rewards yields 78.9%. The joint configuration improves `ComplexOR` from 33.3% for SFT Combined to 61.1%.

5 Conclusion

We propose a two-stage framework that injects expert incremental modeling logic into LLMs via Logic-Anchored Data Synthesis and Logic-TGRPO. By coupling prototype-guided SFT initialization with a pre-trained Logical Reward Model that provides dense process supervision—rewarding correct prototype identification, disciplined reasoning order, and execution efficiency—our 8B model achieves 80.6% average accuracy across seven OR benchmarks, surpassing equivalently scaled baselines and reducing average inference latency by 56% over search-based systems. The results support expert-knowledge-guided optimization modeling under an explicitly transductive adaptation protocol.

Limitations and Future Work. The 17 manually curated prototypes may be insufficient for highly unstructured domains. Future work will explore automated latent prototype discovery, introduce process-level reinforcement learning to score specific intermediate modeling steps, and extend our continuous logical reward to other formal reasoning domains like theorem proving and code synthesis.

Acknowledgments and Disclosure of Funding

Bo Hu and Qian Liu received financial support from the Hong Kong Research Grants Council, Grant T32-615/24-R.

Bo Hu thanks Dr. Qingcan Kang and Professor Qian Liu for their guidance and support throughout this work. He also thanks his family and his girlfriend for their encouragement and support.

References

Ali AhmadiTeshnizi, Wenzhi Gao, Herman Brunborg, Shayan Talaei, Connor Lawless, and Madeleine Udell. OptiMUS-0.3: Using large language models to model and solve optimization problems at scale. *arXiv preprint arXiv:2407.19633*, 2024.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Zezhen Ding, Zhen Tan, Jiheng Zhang, and Tianlong Chen. OR-R1: Automating modeling and solving of operations research optimization problem via test-time reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 228–236, 2026.

Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006. doi: 10.1016/j.patrec.2005.10.010. URL `https://doi.org/10.1016/j.patrec.2005.10.010`.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. URL `https://jmlr.org/papers/v13/gretton12a.html`.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Chenyu Huang, Zhengyang Tang, Shixi Hu, Ruoqing Jiang, Xin Zheng, Dongdong Ge, Benyou Wang, and Zizhuo Wang. ORLM: A customizable framework in training large models for automated optimization modeling. *Operations Research*, 73(6):2986–3009, 2025a.

Xuhan Huang, Qingning Shen, Yan Hu, Anningzhe Gao, and Benyou Wang. LLMs for mathematical modeling: Towards bridging the gap between natural and mathematical languages. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2678–2710, 2025b.

Caigao Jiang, Xiang Shu, Hong Qian, Xingyu Lu, Jun Zhou, Aimin Zhou, and Yang Yu. LLMOPT: Learning to define and solve general optimization problems from scratch. *arXiv preprint arXiv:2410.13213*, 2024.

Beibin Li, Konstantina Mellou, Bo Zhang, Jeevan Pathuri, and Ishai Menache. Large Language Models for supply chain optimization. *arXiv preprint arXiv:2307.03875*, 2023.

Sirui Li, Janardhan Kulkarni, Ishai Menache, Cathy Wu, and Beibin Li. Towards foundation models for mixed integer linear programming. *arXiv preprint arXiv:2410.08288*, 2024.

Haoyang Liu, Jie Wang, Yuyang Cai, Xiongwei Han, Yufei Kuang, and Jianye Hao. OptiTree: Hierarchical thoughts generation with tree search for LLM optimization modeling. *arXiv preprint arXiv:2510.22192*, 2025.

Jiashuo Liu, Zheyan Shen, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624*, 2021. URL `https://arxiv.org/abs/2108.13624`.

David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. In *International Conference on Learning Representations*, 2017. URL `https://arxiv.org/abs/1610.06545`.

Hongliang Lu, Zhonglin Xie, Yaoyu Wu, Can Ren, Yuxuan Chen, and Zaiwen Wen. OptMATH: A scalable bidirectional data synthesis framework for optimization modeling. *arXiv preprint arXiv:2502.11102*, 2025.

Mahdi Mostajabdaveh, Timothy Tin Long Yu, Samarendra Chandan Bindu Dash, Rindra Ramamonjison, Jabo Serge Byusa, Giuseppe Carenini, Zirui Zhou, and Yong Zhang. Evaluating LLM reasoning in the operations research domain with ORQA. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24902–24910, 2025.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Rindranirina Ramamonjison, Timothy Yu, Raymond Li, Haley Li, Giuseppe Carenini, Bissan Ghaddar, Shiqi He, Mahdi Mostajabdaveh, Amin Banitalebi-Dehkordi, Zirui Zhou, et al. NL4Opt competition: Formulating optimization problems based on their natural language descriptions. In *NeurIPS 2022 competition track*, pages 189–203. PMLR, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Raghav Thind, Youran Sun, Ling Liang, and Haizhao Yang. OptimAI: Optimization from natural language using LLM-powered AI agents. *arXiv preprint arXiv:2504.16918*, 2025.

Yang Wu, Yifan Zhang, Yurong Wu, Yuran Wang, Junkai Zhang, and Jian Cheng. Training LLMs for optimization modeling via iterative data synthesis and structured validation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 12880–12896, 2025.

Ziyang Xiao, Dongxiang Zhang, Yangjun Wu, Lilin Xu, Yuan Jessica Wang, Xiongwei Han, Xiaojin Fu, Tao Zhong, Jia Zeng, Mingli Song, et al. Chain-of-Experts: When LLMs meet complex operations research problems. In *The Twelfth International Conference on Learning Representations*, 2024.

Ziyang Xiao, Jingrong Xie, Lilin Xu, Shisi Guan, Jingyan Zhu, Xiongwei Han, Xiaojin Fu, WingYin Yu, Han Wu, Wei Shi, et al. A survey of optimization modeling meets LLMs: Progress and future directions. *arXiv preprint arXiv:2508.10047*, 2025.

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

Zhicheng Yang, Yiwei Wang, Yinya Huang, Zhijiang Guo, Wei Shi, Xiongwei Han, Liang Feng, Linqi Song, Xiaodan Liang, and Jing Tang. OptiBench meets ReSocratic: Measure and improve LLMs for optimization modeling. *arXiv preprint arXiv:2407.09887*, 2024.

Bowen Zhang, Pengcheng Luo, Genke Yang, Boon-Hee Soong, and Chau Yuen. OR-LLM-Agent: Automating modeling and solving of operations research optimization problems with reasoning LLM. *arXiv preprint arXiv:2503.10009*, 2025.

A Expert-Curated Prototype Library: Structural Analyses

We organize optimization problems into 17 canonical prototypes. Each prototype is treated as a reusable modeling scaffold rather than a surface-level text pattern: it specifies the typical decision variables, objective form, and bottleneck constraints that define the problem family. This design follows the expert modeling intuition that one should first identify the structural class of a problem and then instantiate the corresponding formulation with instance-specific coefficients and side conditions.

Table 5 summarizes the full prototype library used in our logic-anchored synthesis pipeline. The emphasis is on structural invariants that should remain stable under contextual rewriting.

A.1 Representative Case I: Resource Allocation

Prototype intuition. Resource Allocation covers problems in which a finite scalar resource must be distributed across competing options to optimize total return. Its modeling core is a small set of continuous allocation variables, one global budget constraint, and a few policy constraints controlling how the allocation is composed.

Canonical formulation. Let $x_i \geq 0$ denote the amount allocated to option $i \in I$, let r_i be the return coefficient, let B be the total budget, let $\alpha \in [0, 1]$ be the minimum share required for a subset $S \subseteq I$, and let ℓ_i be the minimum allocation for option i. A canonical model is

$$\max \quad \sum_{i \in I} r_i x_i, \tag{4}$$

$$\text{s.t.} \quad \sum_{i \in I} x_i \leq B, \tag{5}$$

$$\sum_{i \in S} x_i \geq \alpha \sum_{i \in I} x_i, \tag{6}$$

$$x_i \geq \ell_i, \quad i \in I, \tag{7}$$

$$x_i \geq 0, \quad i \in I. \tag{8}$$

Prototype	Core language pattern	Variables / objective	Key structural constraints
Resource Allocation	Allocate a limited resource across competing uses	Continuous x_i; maximize return	Global budget $\sum_i x_i \leq B$, lower bounds, and share/policy constraints.
Production Planning	Decide production quantities under resource limits	Production x_i; maximize profit	Capacity constraints $Ax \leq b$, optional demand bounds, and integrality.
Scheduling	Assign starts or activities over time	Time-indexed x_t; minimize cost	Coverage over time, overlap logic, continuity, and cyclic constraints.
Blending	Mix ingredients into products with quality requirements	Blend quantities x_j; maximize profit	Material-balance constraints, composition bounds, and raw-material limits.
Workforce Management	Assign worker groups to satisfy period demand	Start-day variables x_t; minimize workforce	Demand coverage induced by work/off patterns, plus cyclic schedule structure.
Selection	Choose a subset of candidates, items, or projects	Binary z_i; minimize cost or maximize value	Cardinality, budget, qualification, and incompatibility constraints.
Sizing	Determine scale or capacity of production activities	Capacity/production x_i; maximize throughput	Machine-time or processing-capacity bottlenecks, often with coupled limits.
Routing	Move goods from origins to destinations	Arc or route variables x_{ij}; minimize cost	Supply, demand, path-feasibility, and nonnegativity/integrality constraints.
Inventory Management	Coordinate production and inventory over time	Production x_t, inventory I_t; minimize cost	Inventory balance $I_t = I_{t-1} + x_t - d_t$, capacity, and boundary conditions.
Packaging / Packing	Pack valuable items into limited capacity	Packing variables z_i; maximize packed value	Knapsack-type capacity $\sum_i w_i z_i \leq C$, exclusivity, and integrality.
Pricing	Set prices under demand-response logic	Price p_i; maximize profit or revenue	Price bounds, demand-linking constraints, and discrete pricing rules.
Balancing	Balance workload or throughput across units	Allocation x_{ij}, scalar P; maximize output	Bottleneck linking constraints, capacities, and cross-unit consistency.
Portfolio Optimization	Allocate capital across investment assets	Allocation x_i; maximize return or utility	Budget equality, diversification rules, and lower/upper share bounds.
Network Flow	Send flow through a graph from supply to demand	Arc flow f_{ij}; minimize flow cost	Flow conservation at nodes and capacity bounds on arcs.
Covering	Select facilities so all demand points are covered	Binary open variables y_j; minimize cost	Coverage constraints $\sum_{j \in N(i)} y_j \geq 1$ for every demand point.
Layout	Allocate limited space across products or activities	Area variables x_i; maximize revenue	Total-space, budget, labor, or adjacency constraints.
Warehouse Selection	Open warehouses and optionally assign regions	Opening y_j, assignment x_{ij}; minimize cost	Opening logic, assignment feasibility, capacity, and demand satisfaction.

Table 5: Compact summary of the 17 expert-curated optimization prototypes. We emphasize the canonical decision structure of each class rather than its narrative surface form.

Structural bottleneck. The defining bottleneck is the composition-share constraint in Equation (6). In many natural-language instances, this condition appears as a ratio requirement, but an expert formulation rewrites it as a linear inequality. For example, if C and D denote the investments allocated to condos and to all other options, respectively, then a statement such as "at least 20% of total investment must be allocated to condos" becomes $C \geq 0.2(C + D)$, equivalently $4C \geq D$. This transformation is precisely the kind of prototype-grounded prior that our framework aims to internalize.

Why it is representative. This prototype illustrates a fundamental expert modeling habit: identify the allocation skeleton first, then instantiate coefficients and policy rules. Once the problem is recognized as Resource Allocation, the admissible formulation space becomes much narrower, which reduces spurious variables and incorrect nonlinear expressions.

A.2 Representative Case II: Network Flow

Prototype intuition. Network Flow captures problems in which commodities or services move through a graph from supply nodes to demand nodes. Its defining feature is not the transportation narrative itself, but the conservation law imposed at intermediate and terminal nodes.

Canonical formulation. Let $G = (V, E)$ be a directed graph, let $f_{ij} \geq 0$ denote the flow on arc $(i, j) \in E$, let c_{ij} be the unit cost, let u_{ij} be the arc capacity, and let b_v denote the net supply at node v. A canonical minimum-cost formulation is

$$\min_{f} \quad \sum_{(i,j)\in E} c_{ij} f_{ij} \tag{9}$$

$$\text{s.t.} \quad \sum_{j:(v,j)\in E} f_{vj} - \sum_{i:(i,v)\in E} f_{iv} = b_v, \quad \forall v \in V, \tag{10}$$

$$0 \leq f_{ij} \leq u_{ij}, \quad \forall (i, j) \in E. \tag{11}$$

Structural bottleneck. The essence of this prototype is the flow-conservation constraint in Equation 10. Expert modelers do not treat source constraints, transshipment constraints, and demand constraints as unrelated clauses; instead, they recognize them as one unified node-balance template. This abstraction is exactly what makes the Network Flow prototype reusable across logistics, transportation, and infrastructure settings.

Why it is representative. This prototype provides a concrete scaffold for formulating network-flow problems; benchmark accuracy alone does not distinguish structural reasoning from lexical or template association. Once the graph structure is recognized, the canonical variable design, objective, and conservation constraints follow almost deterministically from the prototype.

B RL Reward Details

Detailed construction of v_{logic}. In the main text, we present v_{logic} in a compact form for readability. Its exact instantiation is

$$v_{\text{logic}} = \begin{cases} R_{\text{exec}} + R_{\text{semantic}} + R_{\text{time}}, & \text{if } R_{\text{exec}} = 1, \\ R_{\text{traj}}, & \text{if } R_{\text{exec}} = 0. \end{cases} \tag{12}$$

Here, $R_{\text{semantic}} \in [0, 0.5]$ is the semantic alignment score assigned by an external judge model, and

$$R_{\text{time}} = \beta_{\text{time}} \exp(-\kappa_{\text{time}} t_{\text{exec}}), \tag{13}$$

where t_{exec} is the execution time, $\beta_{\text{time}} = 2.0$, and $\kappa_{\text{time}} = 0.5$.

Training and generation configuration. SFT uses batch size 1, gradient accumulation over 16 steps, 80 epochs, a learning rate of 2×10^{-5}, and a maximum sequence length of 3,600 tokens. Logic-TGRPO rollouts use temperature 0.9, top-p 1.0 (no nucleus truncation), four generations per prompt, and limits of 5,500 thinking tokens and 1,500 completion tokens. Final pass@1 evaluation uses greedy decoding (temperature 0, top-p 1.0). These stochastic rollout settings are distinct from the final inference settings.

Detailed trajectory and filtering hyperparameters. For the trajectory penalty, we set

$$\lambda_{\text{start}} = 0.1, \qquad \lambda_{\text{omit}} = 0.05. \tag{14}$$

For the hard-filtering mechanism in Logic-TGRPO, we set

$$\gamma_{\text{pass}} = 1.6, \qquad \gamma_{\text{fail}} = 0.7, \qquad \lambda_{\text{filter}} = 0.5. \tag{15}$$

Unlabeled pool used in RL. During the Logic-TGRPO stage, we optimize the policy entirely on $\mathcal{D}_{\text{unlabeled}}$ without using human annotations. Thus, the RL signal is exclusively derived from structural validity, executability, logical reward modeling, and within-group consensus, cleanly decoupling policy improvement from the need for supervised answer matching.

C Prompt Templates and Generation Formatting

To ensure full reproducibility and to clarify the exact contexts provided to the models, we detail the prompt templates utilized during both the Supervised Fine-Tuning (SFT) and Logic-TGRPO training phases.

C.1 Supervised Fine-Tuning (SFT) Prompt.

During the SFT phase, the objective is to align the model to output a structured Connection Graph (CG) representation alongside executable solver code. We format the input using a direct instruction-following template to strictly govern the output schema:

> Instruction: Below is an operations research question. Build a structured CG model in JSON with `variables`, `objective`, and `constraints`, each containing mathematical formulations only, together with the complete Python solution using `gurobipy` that appropriately addresses the question.
> Input: [QUESTION]

C.2 Logic-TGRPO Phase Prompts

In the reinforcement learning phase, we explicitly elicit extended chain-of-thought reasoning before generating the final structured answer. We utilize the chat template format specific to the Qwen3 tokenizer, distinguishing between the system role and the user role.

System Prompt. We assign an expert persona and explicitly enforce the usage of reasoning boundary tags:

> You are an expert in operations research. Be concise in your reasoning and focus on key steps only. When reasoning, always close your thinking block with `</think>` before answering.

User Prompt and Output Template. The user prompt concatenates the raw problem description with a token-length constraint and a mandatory output format. This strict formatting ensures that the generated output can be deterministically parsed for execution and RL reward computation.

Logic-TGRPO User Prompt and Output Template

```
Below is an operations research question. Build a mathematical model and
corresponding Python code using `gurobipy` that appropriately addresses the
question.

Question: [QUESTION]

Please use fewer than 5,000 tokens for reasoning.

Response: Fill in the following format

Mathematical Model: [YOUR ANSWER HERE]

Decision Variables: [YOUR ANSWER HERE]

Objective Function: [YOUR ANSWER HERE]

Constraints: [YOUR ANSWER HERE]

Python Code Solution Using `gurobipy`: [YOUR ANSWER HERE]
```

Forcing the Reasoning State. Qwen3's native thinking format begins the assistant response with `<think>`. We append `<think>\n` at that same response boundary to activate the native format rather than introduce a different reasoning template. This formatting choice does not itself establish faithful internal reasoning.

Format Reward R_{format} Calculation. As established in Section 3.4, the format reward $R_{\text{format}} \in [0, 1]$ evaluates the policy's adherence to the requested template. We implement this deterministically by verifying the presence of six required structural substrings within the generated output: `Mathematical Model`, `Decision Variables`, `Objective Function`, `Constraints`, `Python Code Solution Using`, and `gurobipy`. The ratio of successfully matched substrings directly constitutes the continuous format score.

D Synthetic Data Scale Sensitivity

For the archived scale-sensitivity sweep, we vary the number of synthetic problem–formulation pairs used alongside OR-Instruct during SFT from 50 to 1,000. Table 6 reports the macro-average solution accuracy across the seven benchmarks, with the original reference answers used for all four settings. Increasing the synthetic set from 50 to 100 examples improves average accuracy by 7.20 percentage points; increasing it from 100 to 500 and 1,000 examples yields further gains of 0.35 and 1.17 points, respectively. These differences are computed before rounding. In this sweep, most of the improvement occurs between 50 and 100 synthetic examples.

Table 6: Archived synthetic data scale sensitivity for SFT with OR-Instruct. All four settings use the original benchmark reference answers, including the original ComplexOR reference.

Synthetic examples	Average accuracy (%)
50	61.45
100	68.66
500	69.01
1,000	69.83

The 100-example setting in this archived sweep uses a different synthetic training subset from SFT Combined in Table 1, as well as the original benchmark reference answers. The affected in-house results in Tables 1, 2, and 4 use the corrected ComplexOR LP reference (55.67 rather than 66), with the same evaluator and all instances retained. The 68.66% and 69.4% results therefore differ in both training subset and evaluation reference; we compare data scales within the archived sweep.

E Additional Distribution Analysis

The joint ablation analyzed in Section 4.3 removes both problem- and solution-synthesis prototypes and is separate from the problem-prototype ablation shown in Figure 2. It contains 121 examples with a mean description length of 2,538 characters. The median complexity scores under our non-unit weighting are 9.77 for OR-Instruct, 35.48 for the full corpus, and 58.00 for the joint ablation. Under the all-ones sensitivity instantiation, the corresponding scores are 23.62, 35.82, and 41.33. Comparing the full and joint-ablation corpora, the custom scalar difference is not significant (KS $p = 0.2005$), whereas the all-ones sensitivity is (KS $p = 0.0033$). These results indicate a modest, weighting-sensitive shift in parser-observed structure and complexity.

The benchmark-level text diagnostics measure source distinguishability in the chosen representation. We avoid high-dimensional KL estimation for the sparse 4,096-dimensional representation and small target sets because density estimation is unreliable. Distribution statements distinguish source/narrative, mathematical structure, complexity, and adaptation-exposure axes; high source distinguishability is not evidence of test independence. Stable performance comparisons for genuinely unmatched prototype strata require additional cases and are not established by these aggregate diagnostics.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have made main claims that accurately reflect the paper's contributions and scope in abstract and introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the limitations of this work.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: This paper does not include the content of the theoretical analysis results that require assumptions or proof.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides all the information needed to reproduce the main experimental results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We have provided the experimental code and test data to facilitate reproduction.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies all experimental details, including hyper-parameters, the rationale for their selection and underlying model.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: In our experiments, all results are averaged over multiple independent runs to ensure statistical reliability and reduce evaluation variance.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We explicitly specify in the Implementation Details paragraph that all training and experimental runs were conducted on a single modified NVIDIA RTX 4090 GPU equipped with 48GB of VRAM.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms with the NeurIPS Code of Ethics in every respect.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The work performed has no direct societal impact, as it relies solely on calling LLMs and their behavior is fully controlled. The approach does not involve any external factors or unintended consequences that could potentially affect society.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The creators of the data and LLMs employed in this research have been appropriately acknowledged and adhered to the respective licenses governing their use for research purposes.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new assets, including the source code and datasets used in this work, are well documented and the documentation is provided alongside the assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs constitute the core methodology of this research. We explicitly describe the use of LLMs for logic-anchored data synthesis, as an evaluator for the Logical Reward Model, and detail the target policy model (Qwen3-8B) optimized via Logic-TGRPO.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.