Draft-and-Audit Reinforcement Learning for Optimization Modeling

Zeping Min, Weihang Xu, Zhengzhong You, Wotao Yin, Xinshang Wang

Alibaba Group

Motivation: Reliable NL2Opt

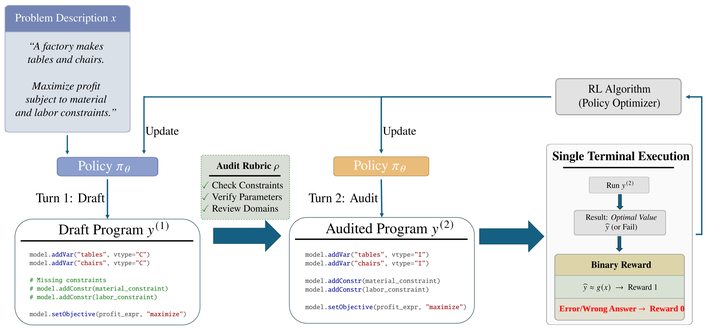
Natural language to optimization (NL2Opt) asks a model to convert a word problem into executable solver code. Correctness requires more than valid Python: the generated program must preserve variables, constraints, objectives, domains, and numeric values.

- We target **silent modeling failures**: programs that execute successfully but return the wrong objective value.
- We train revision as a learned capability rather than relying on an inference-time repair scaffold.
- We evaluate by sandboxed execution and compare the returned objective with the ground truth.

$$r(x, y) = \mathbb{1}[\mathcal{E}(y) \neq \bot \ \wedge \ |\mathcal{E}(y) - g(x)| \leq \tau] \, .$$

DA-RL Protocol

We optimize a strict two-turn policy with shared parameters. The model first drafts a complete PySCIPOpt program and then audits that draft using a fixed rubric.



- **Draft:** generate a complete solver program from the problem text and few-shot modeling examples.
- **Audit:** inspect constraints, parameters, and variable domains; edit only when clear rubric violations are found.
- **Terminal-only verification:** execute only the audited program $y^{(2)}$; no solver traces are shown between turns.
- **Shared parameters:** reward from the audited output updates both the draft and audit behaviors, encouraging first-pass code that is easier to verify and repair.

Learning Objective

For each problem x, DA-RL samples a draft and an audited output:

$$y^{(1)} \sim \pi_\theta(\cdot \mid x), \qquad y^{(2)} \sim \pi_\theta(\cdot \mid x, y^{(1)}, \rho).$$

We maximize terminal correctness of the audited program:

$$\mathcal{J}(\theta) = \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y^{(1)}, y^{(2)}} \left[r(x, y^{(2)}) \right].$$

- We initialize from Qwen2.5-7B-Instruct and train with PPO using binary execution rewards.
- A successful terminal reward reinforces both turns, coupling drafting with auditing.
- This coupling encourages **audit-friendly drafts**: first-pass programs that are internally consistent and easier to repair.

Main Results

Under the public-split protocol, we train on 80% of each benchmark and evaluate on the held-out 20%, averaged over three random splits.

Model	NL4	ME	MC	IOR	OM	Avg
Claude Sonnet 4.5	92.0	92.4	50.4	**45.0**	55.9	67.1
Claude Opus 4.5	90.6	91.1	56.6	40.0	**56.9**	67.0
SIRL 7B†	96.3	91.7	51.7	33.0	30.5	60.6
DA-RL 7B	**97.1**	91.8	**84.5**	36.7	30.4	**68.1**

NL4: NL4Opt; ME: MAMO-Easy; MC: MAMO-Complex; IOR: IndustryOR; OM: OptMATH-Bench. † Reference-only due to different training or evaluation access.

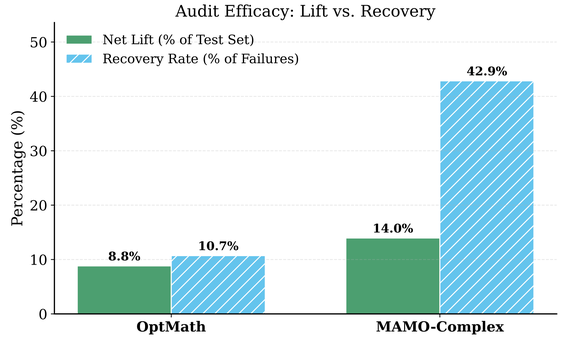
- DA-RL achieves the best macro average among comparable public-split rows.
- The largest gain appears on MAMO-Complex, where constraint and domain errors are common.

Ablation	MAMO-C	OptMATH
With few-shot	84.5	30.4
Zero-shot	76.7	11.8

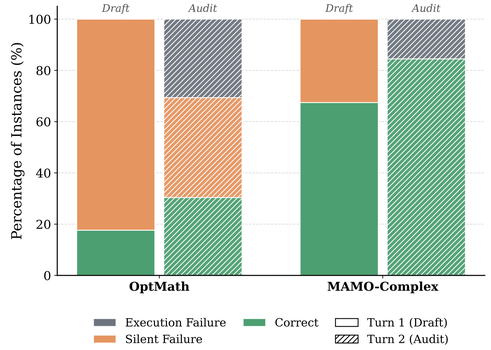
Few-shot exemplars help the policy ground unstructured text into valid mathematical formulations, especially on harder benchmarks.

Audit Behavior

We execute both the draft and audited program on held-out data to measure how the audit changes outcomes.



- MAMO-Complex: **+14.0%** net lift and **42.9%** error recovery.
- OptMATH-Bench: **+8.8%** net lift and **10.7%** error recovery.
- We observe **0% regression** on analyzed splits, indicating conservative repairs.



Takeaways

- We formulate NL2Opt as a learned two-turn Draft→Audit policy instead of a single-pass translation.
- Terminal-only verification forces the model to internalize rubric-guided revision without intermediate solver feedback.
- Shared parameters create cross-turn synergy: better auditing also shapes drafts toward easier verification and repair.
- Auditing reduces silent failures, including a drop from 32.6% to 0.0% on MAMO-Complex.

Core result: DA-RL 7B reaches 68.1% macro pass@1

Key mechanism: learned conservative repair