PTPO: PROGRESSIVE THEOREM-GUIDED POLICY OPTIMIZATION FOR LLM-BASED MATHEMATICAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite recent great advances in mathematical reasoning, large language models still struggle with competition-level problems that require selecting appropriate mathematical theorems and applying them under the correct conditions. Existing reinforcement learning methods mainly optimize final-answer correctness or generic reasoning trajectories, often providing limited guidance on the theorem-level principles. To address this limitation, we propose Progressive Theorem-guided Policy Optimization (PTPO), a theorem-guided reinforcement learning framework with input-adaptive progressive prompt construction. We first construct a structured theorem library and generates theorem-grounded chain-of-thought prompts that explicitly connect problems with relevant mathematical principles. Based on the theorem library, we then adaptively inject theorem-grounded prompts according to the question difficulty and model's current ability, offering rich hints for hard questions and progressively reducing hints as the model improves. This progressive schedule encourages the model to transition from external theorem guidance to autonomous theorem application. Extensive experiments on nine benchmarks show that our proposed PTPO achieves the best overall performance, and it enables faster convergence and stronger reasoning. Our source code and theorem library are available at `https://anonymous.4open.science/r/PTPO-FED5`.

1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable capabilities across a wide range of reasoning-intensive tasks. Among them, mathematical reasoning has become a central testbed for evaluating whether LLMs can perform symbolic manipulation, multi-step deduction, and knowledge-grounded problem solving (Cobbe et al., 2021; Hendrycks et al., 2021). Unlike general text generation, solving complex mathematical problems often requires the model to identify the underlying mathematical principle, select the appropriate theorem, and apply it under the correct conditions. Therefore, improving mathematical reasoning is not only about producing fluent chain-of-thought (CoT) explanations (Wei et al., 2022), but also about grounding the reasoning process in valid mathematical knowledge (Yang et al., 2024; Ahn et al., 2024; Meadows & Freitas, 2023; Wu et al., 2022; Trinh et al., 2024).

Recent advances have significantly improved the mathematical reasoning ability of LLMs. Chain-of-thought prompting encourages models to decompose problems into intermediate reasoning steps (Wei et al., 2022; Wang et al., 2023), instruction tuning further adapts models to mathematical solution formats (Yue et al., 2024; Luo et al., 2023), and reinforcement learning with verifiable feedback has emerged as an effective paradigm for optimizing reasoning models using automatically checkable correctness signals (Guo et al., 2025; Yu et al., 2025). These methods have shown that LLMs can be substantially improved by reinforcing answers and high-quality reasoning trajectories.

However, existing methods primarily optimize final-answer correctness or solution trajectories, providing limited explicit guidance on which theorems to select and how to apply them. For example, Group Relative Policy Optimization (GRPO) (Shao et al., 2024) samples multiple responses for each problem and updates the policy using group-relative advantages derived from their rewards. As il-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

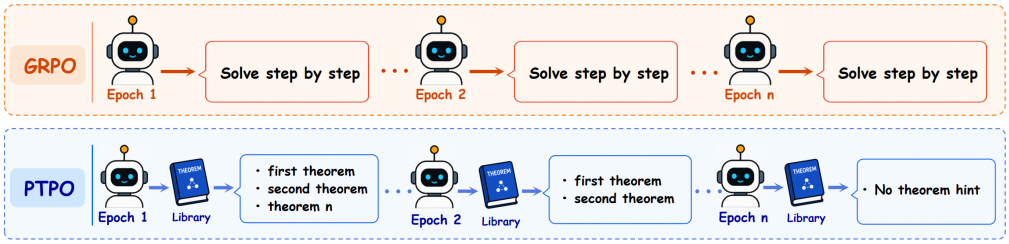


Figure 1: Motivation of our progressive theorem-guided reinforcement learning framework. Standard RL, such as GRPO, uses fixed problem prompts during training, while our method adaptively injects theorem-aware prompts and gradually reduces them as the model improves.

lustrated in Figure 1, standard RL typically uses the same generic instruction across training epochs, without adapting theorem guidance to the model's evolving ability. This differs from human mathematical learning, where humans often begin with explicit theorem guidance when facing difficult problems, learn to apply the required theorem to each reasoning step, and gradually become less dependent on such guidance through repeated practice (Zhou et al., 2023; Zelikman et al., 2022). In this sense, each theorem often corresponds to a key reasoning step: once the correct theorem is selected and properly applied, the resulting derivation becomes more constrained and reliable. By contrast, outcome-oriented rewards can encourage the model to reach correct answers, but they do not explicitly teach which mathematical principle should govern the intermediate reasoning process. As a result, models may generate fluent and plausible multi-step solutions while missing the key theorem, applying an inappropriate principle, or overlooking necessary conditions. Existing theorem-aware methods provide useful theorem information (Chen et al., 2023; Sheng et al., 2025; Welleck et al., 2021), but usually treat it as a static supervision signal, without studying how theorem usage can be progressively internalized during reinforcement learning.

To fill this gap, we propose a theorem-guided reinforcement learning framework with input-adaptive progressive prompt construction. Instead of using fixed problem prompts throughout training, our method adaptively injects theorem-grounded CoT prompts according to the question difficulty and model's current ability. Problems that the model struggles with receive richer theorem-aware guidance, while easier problems receive less external information. Through repeated training with gradually reduced hints, the model is encouraged to shift from relying on external theorem guidance to autonomously identifying and applying relevant theorems, similar to how students internalize mathematical knowledge through guided practice.

Our framework consists of two main stages. First, we construct a structured theorem library from large-scale mathematical corpora and retrieve relevant theorems to build theorem-grounded CoT prompts. This provides a scalable way to align mathematical problems with their underlying principles without relying on manually curated theorem annotations. Second, we introduce a progressive theorem-guided reinforcement learning strategy, where the amount of theorem-grounded CoT prompt is adaptively adjusted across training. Our contributions are summarized as follows:

- We construct a structured library of 7,875 theorems and use it to build theorem-grounded CoT prompts that align mathematical problems with relevant theorems.
- We propose Progressive Theorem-guided Policy Optimization (PTPO), which adaptively adjusts theorem-guided prompts according to problem difficulty and the model's current capability.
- Our proposed PTPO improves both training efficiency and mathematical reasoning performance, achieving faster convergence and the best overall results on nine mathematical benchmarks with fewer training resources.

2 Related Work

2.1 Chain-of-Thought Reasoning

Chain-of-thought (CoT) prompting has become a key technique for improving the reasoning ability of LLMs by encouraging them to generate intermediate reasoning steps before final an-

swers (Wei et al., 2022). Follow-up studies further show that explicit reasoning traces, self-consistency, and step-by-step supervision can substantially improve performance across diverse reasoning tasks (Wang et al., 2023; Yue et al., 2024; Shao et al., 2024). In mathematical reasoning, CoT is particularly important because solving problems requires multi-step symbolic derivation and logical justification. However, existing CoT-based methods mainly supervise the reasoning trajectory itself, while often overlooking the mathematical principles that justify each step. Our work addresses this limitation by constructing theorem-grounded CoT prompts, enabling models to learn not only step-by-step reasoning patterns but also the underlying theorem knowledge that should guide them.

2.2 RL for LLM Reasoning

Reinforcement learning with verifiable rewards has become a powerful paradigm for improving LLM reasoning, especially in mathematics where final answers can often be automatically checked. Representative methods such as DeepSeekMath, DeepSeek-R1, and DAPO use rule-based correctness signals and group-based policy optimization to improve reasoning performance without dense human supervision (Shao et al., 2024; Guo et al., 2025; Yu et al., 2025). Other works explore process-level rewards or more informative supervision signals for mathematical reasoning (Wang et al., 2024). Despite their success, these methods mainly focus on optimization and reward design, while the training prompts usually remain fixed and provide limited explicit guidance on which mathematical principles should govern the reasoning process. Prompt augmentation methods such as QuestA and PieceHint improve sample efficiency by injecting partial solutions or critical steps during training (Li et al., 2026; Fang et al., 2026), but their hints are not explicitly grounded in theorem knowledge. In contrast, our PTPO framework adaptively injects theorem-guided CoT hints according to the question difficulty and model's current ability, progressively guiding the model from external theorem assistance to autonomous theorem usage.

2.3 Theorem-guided Reasoning

Explicit theorem knowledge has been increasingly studied for mathematical reasoning. NaturalProofs and NaturalProver investigate theorem retrieval and proof generation with natural language mathematical statements (Welleck et al., 2021; 2022). TheoremQA introduces a theorem-driven benchmark covering expert-curated questions across mathematics, physics, engineering, and finance (Chen et al., 2023). Theorem Rationale further constructs question–theorem–solution triples and designs theorem-oriented instructions to improve theorem selection and application (Sheng et al., 2025). These works demonstrate the importance of theorem knowledge, but they use theorem information as static supervision or evaluation resources. Our method instead builds a reusable theorem repository and incorporates theorem-grounded CoT into reinforcement learning as adaptive guidance, allowing theorem usage to be progressively internalized during policy optimization.

3 Theorem-guided Prompt Repository

To support theorem-aware mathematical reasoning, we first construct a theorem-guided prompt repository that consists of two components: a structured mathematical theorem library and a set of theorem-grounded explicit CoT prompts. As illustrated in Figure 2, the theorem library stores reusable mathematical principles in a structured format, while the prompt repository connects each problem with relevant theorems and converts them into explicit reasoning trajectories. This repository is constructed offline and later used as adaptive reasoning guidance during training.

3.1 Structured Theorem Formulation

Following prior work (Chen et al., 2023; Yang et al., 2023), we represent each theorem $T_i \in \mathcal{T}$ in a unified structured format:

$$T_i = (n_i, s_i, f_i, k_i), \tag{1}$$

where n_i is the theorem name, s_i denotes its natural-language statement, f_i is its symbolic expression, and k_i indicates a set of keywords. Figure 2(a) shows an example. This formulation captures both semantic and formal aspects of mathematical knowledge: the statement supports alignment with free-form problem descriptions, the symbolic expression records the mathematical relation

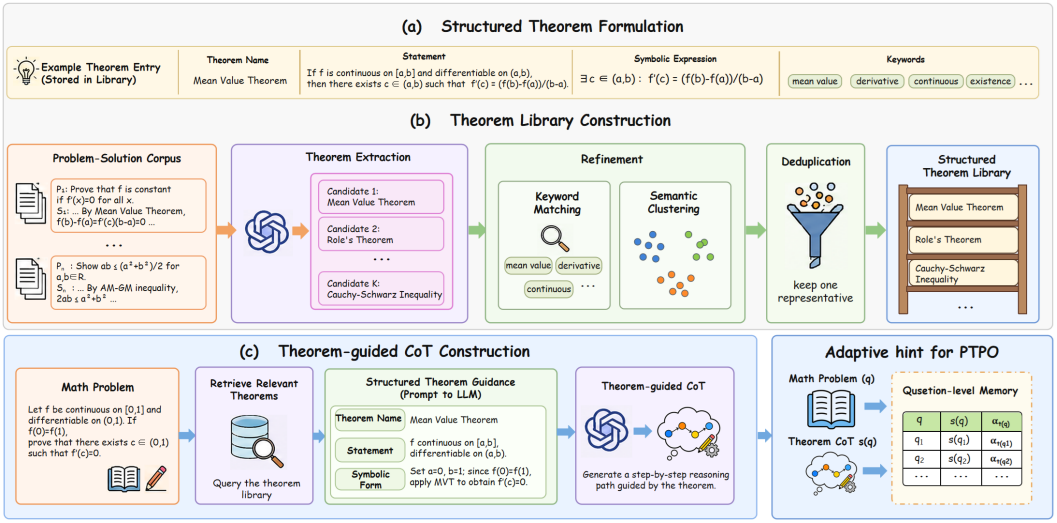


Figure 2: Illustration of the theorem-guided prompt repository. Given a mathematical problem, relevant theorems are retrieved from the structured theorem library and injected into the reasoning process. The generated theorem-grounded CoT explicitly connects problem-solving steps with the underlying mathematical principles, providing adaptive reasoning guidance for later training.

compactly, and the keywords support retrieval and lexical matching. This structured formulation provides a common schema for later theorem construction, retrieval, and theorem-grounded CoT generation.

3.2 Automated Theorem Construction

Theorem Extraction. To instantiate the above structured schema, we extract theorem candidates from solved problem-solution pairs. Given a solved instance $x = (q, r)$, where q denotes the problem and r denotes the solution process, we employ a large language model to identify the mathematical principles that support the reasoning steps:

$$\mathcal{T}_x = \mathrm{LLM}(q, r),$$

where $\mathcal{T}_x = \{T_1, T_2, \ldots, T_m\}$ denotes the set of theorem candidates extracted from the solution process. Following recent studies on theorem-level mathematical reasoning (Chen et al., 2023; Sheng et al., 2025), we instruct the extraction model to identify all relevant theorems used in each solution and convert them into the structured format defined above. Since a single solution may involve multiple mathematical principles, this step may produce multiple theorem entries for each problem. The corresponding extraction prompt is provided in Appendix B.1. This extraction step converts problem-specific reasoning trajectories into candidate theorem knowledge, which is further filtered to build the final theorem library.

Theorem Consolidation and Filtering. The above theorem extraction step may produce many candidate theorems that are semantically equivalent or highly similar, since the same mathematical principle can appear in different problems or be expressed in different forms. Keeping these redundant candidates would make the theorem library unnecessarily large and may reduce the precision of subsequent theorem retrieval. Therefore, we enhance the extracted candidates through semantic deduplication and filtering. Specifically, we encode each extracted theorem into a dense representation via bge-m3 (Chen et al., 2024) and cluster semantically similar theorems in the embedding space. For each cluster, we merge redundant or near-duplicate candidates into a canonical theorem entry. We further filter the consolidated results by checking the consistency between natural language statements and symbolic formulations, removing incomplete or ambiguous entries, and discarding low-confidence or infrequent patterns. Through this process, we obtain a compact and reliable theorem library $\mathcal{T}$ containing approximately 7,875 structured theorems, which is used for subsequent theorem-grounded CoT construction.

3.3 THEOREM-GROUNDED COT

After constructing the theorem library, we next use it to generate theorem-grounded CoT data. A straightforward approach is to prompt an LLM with only a problem-solution pair $x = (q, r)$. However, this often leads the model to paraphrase the original solution without explicitly linking each reasoning step to the underlying mathematical principles. To address this issue, we first retrieve relevant theorems from the structured theorem library and use them as mathematical anchors for subsequent CoT generation.

Theorem Retrieval. We adopt a hybrid retrieval strategy to identify theorems that are both semantically relevant and lexically aligned with the problem and solution. Semantic retrieval captures theorems that may be expressed differently from the solution text, while keyword matching preserves exact mathematical terms and symbolic cues. Specifically, for semantic retrieval, each theorem is represented by concatenating its name, statement, keywords, and formal expression, and the problem-solution pair is encoded in the same embedding space. We then select the top-5 theorems according to embedding similarity. In parallel, for keyword matching, we extract mathematical terms from the problem and solution and match them with the keyword fields of the theorem library. The candidates retrieved from both paths are merged and deduplicated to form the theorem set used for CoT generation.

Theorem-grounded CoT Generation. Given a problem-solution pair $x = (q, r)$ and its retrieved theorem set $\mathcal{T}_x$, we prompt a large language model to rewrite the original solution as a theorem-grounded CoT:

$$\mathrm{CoT}_x = \mathrm{LLM}(q, r, \mathcal{T}_x).$$

The instruction asks the model to decompose the original solution into reasoning steps and explicitly associate each key step with the relevant theorem, without imposing a fixed reasoning template. The resulting CoT preserves the original solution procedure while making theorem use explicit. Both theorem retrieval and CoT generation are performed once before reinforcement learning, and the resulting theorem sets and CoTs are cached for progressive prompt construction during training.

4 PROGRESSIVE THEOREM-GUIDED POLICY OPTIMIZATION

Conventional group-based policy optimization samples multiple responses for each problem and updates the policy based on their relative correctness. However, such methods usually rely on a fixed problem prompt throughout training, without adapting the amount of reasoning guidance to the model's current ability on each question.To address this limitation, we propose Progressive Theorem-guided Policy Optimization (PTPO), a theorem-guided reinforcement learning framework with input-adaptive progressive prompt construction. As illustrated in Figure 3, for each question, PTPO estimates the current model's solving ability from rollout correctness and uses this estimate to construct the next-round prompt. Specifically, when the model struggles with a question, PTPO injects a larger portion of the corresponding theorem-grounded CoT as guidance; when the model can solve the question more reliably, PTPO reduces the injected guidance. Through this closed loop of rollout feedback, adaptive prompt construction, and policy optimization, PTPO gradually reduces external theorem guidance as the model improves.

4.1 ROLLOUT-BASED ABILITY ESTIMATION

To adapt the amount of theorem guidance for each question, PTPO first needs to estimate how well the current policy can solve that question. Since this ability changes during training, we use rollout performance as an online signal to measure the current model's solving ability for a specific question. Specifically, given a problem q, we sample n candidate responses from the current policy:

$$\{o_1, o_2, \ldots, o_n\} \sim \pi_\theta(\cdot \mid q).$$

Each sampled response is evaluated by an automatic verifier, yielding a binary correctness signal:

$$r_i \in \{0, 1\}, \quad i = 1, \ldots, n.$$

We then compute the empirical success rate of the current policy on problem q as $\alpha(q) = \frac{1}{n} \sum_{i=1}^{n} r_i$. The value of $\alpha(q)$ reflects the model's current solving ability for this question. A high value indicates

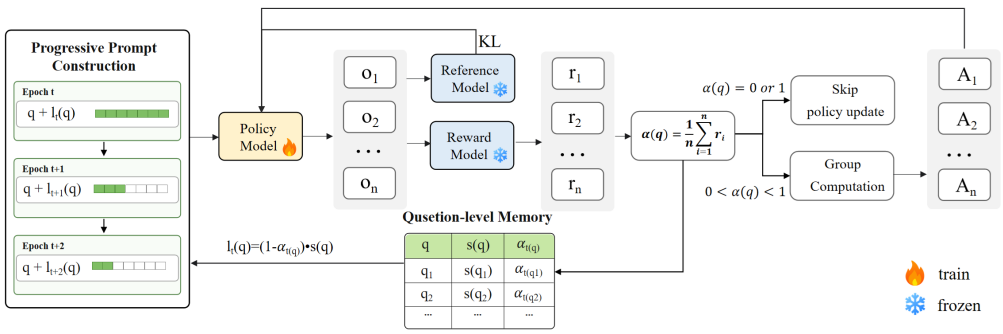


Figure 3: Overview of our proposed Progressive Theorem-guided Policy Optimization (PTPO). For each question, PTPO first constructs a training prompt by combining the original question with a portion of its theorem-grounded CoT. From the rollout outcomes, PTPO computes the question-level success rate $\alpha(q)$, i.e., the proportion of correct responses in the rollout group, as an estimate of the model's current ability on question q, and stores it in memory. In the next epoch, PTPO uses this ability estimate to provide more theorem-grounded CoT for questions with lower success rates and less guidance for questions with higher success rates.

that the model can already solve the problem reliably, while a low value suggests that the model still requires more guidance. This ability estimate is then used in the next subsection to determine how much theorem-grounded CoT should be injected into the next-round prompt.

4.2 Progressive Prompt Construction

Based on the estimated rollout success rate $\alpha(q)$, PTPO stores the result in a question-level memory and uses it to construct the prompt for the same problem in the next epoch. Specifically, the prompt ratio is $1 - \alpha(q)$. A lower success rate therefore provides a larger portion of the corresponding theorem-grounded CoT, while a higher success rate provides a shorter hint. We truncate the theorem-grounded CoT s_q according to this ratio and concatenate the selected prefix with the original problem to form its next-round training prompt. For a newly encountered problem, $\alpha(q)$ is initialized to 1, so the first rollout is performed without theorem-grounded guidance. After each rollout, the stored success rate is updated and used when the same problem appears in the following epoch.

4.3 Reward Design

Given the adaptive prompt, PTPO samples a rollout group and performs group-relative policy optimization. We use a simple rule-based reward consisting of a correctness reward and a format reward:

$$R(q, y) = R_{\text{correct}}(q, y) + \beta R_{\text{format}}(y),$$

where $R_{\text{correct}}(q, y) = 1$ if the parsed final answer matches the ground-truth answer and 0 otherwise, while $R_{\text{format}}(y) = 1$ if the response follows the required output format and 0 otherwise. The coefficient β controls the contribution of the format reward. The progressive prompt ratio is computed separately from rollout accuracy and is not included in the reward.

5 Experiments

5.1 Experimental Setup

Training Data. For training, we sample 10K mathematical problems from OpenR1-Math and refine their reasoning traces using our explicit theorem library. For each problem, we retrieve relevant theorems and rewrite the original chain-of-thought into a theorem-grounded reasoning trajectory, which is then used to construct adaptive theorem-CoT prompts for reinforcement learning.

Evaluation Benchmarks. For evaluation, we use a broad set of held-out mathematical reasoning benchmarks, including AIME 2024 (Li et al., 2024), AIME 2025 (Balunović et al., 2025), AMC 2023 (Li et al., 2024), MATH-500 (Hendrycks et al., 2021; Lightman et al., 2023), Minerva

Model	AIME24	AIME25	AMC23	MATH	Minerva	Olympiad	HMMT	BRUMO	CMIMC	Avg
Backbone: DeepSeek-R1-Distill-Qwen-1.5B										
DeepSeek-R1-1.5B	29.90	22.40	63.82	84.90	34.65	45.95	13.44	30.94	12.89	37.65
DeepScaleR-1.5B	40.21	28.65	73.83	89.30	39.34	52.79	18.96	40.00	21.00	44.90
TP-GRPO	33.12	25.63	64.01	83.81	–	43.91	–	–	–	–
ProRL-V2	51.87	35.73	88.75	92.00	49.03	67.04	19.38	47.29	25.86	53.08
BroRL	**57.50**	36.88	–	**92.14**	49.08	61.54	–	–	–	–
PTPO	53.95	**39.58**	**90.85**	91.60	**49.93**	**67.24**	**21.97**	**52.18**	**25.91**	**54.76**
Backbone: OpenMath-Nemotron-1.5B										
Nemotron-1.5B	58.75	48.44	90.55	92.40	26.93	71.70	30.10	61.67	30.08	56.74
QuestA	70.56	62.08	93.44	92.95	**32.08**	72.28	37.94	64.56	41.48	63.04
PieceHint	54.50	43.70	89.10	91.30	28.50	68.30	–	–	–	–
PTPO	**71.08**	**62.10**	**96.01**	**93.90**	29.59	**76.48**	**38.00**	**65.10**	**42.21**	**63.82**

Table 1: Performance comparison on nine mathematical reasoning benchmarks using two backbones. Our proposed PTPO consistently achieves competitive or superior performance compared with existing RL-based methods, obtaining the highest average accuracy for both backbones.

Math (Lewkowycz et al., 2022), OlympiadBench (He et al., 2024), HMMT Feb 2025 (Balunović et al., 2025), CMIMC 2025 (Balunović et al., 2025), and BRUMO 2025 (Balunović et al., 2025). These benchmarks cover diverse mathematical domains and difficulty levels. All evaluation benchmarks are excluded from training.

Evaluation Protocol. Following prior work, we report accuracy as the primary evaluation metric. For all experiments, we use the same decoding configuration across methods. Specifically, we set the sampling temperature to 1.0, top-p to 1.0, top-k to -1, and the maximum response length to 12,288 tokens. The maximum prompt length is set to 8,192 tokens. During rollout generation, we sample 16 responses for each prompt and evaluate the generated answers with an automatic verifier.

5.2 Performance Comparison with the State-of-the-Art

In this experiment, we compare PTPO with representative RL-based reasoning methods, including DeepScaleR (Luo et al., 2025), TP-GRPO (He et al., 2025), ProRL-V2 (Hu et al., 2025), and BroRL (Hu et al., 2026), as well as prompt-augmentation methods such as QuestA (Li et al., 2026) and PieceHint (Fang et al., 2026). Table 1 presents the performance comparison under two backbone settings. With the same backbone, PTPO achieves the highest average accuracy on nine datasets. On *DeepSeek-R1-Distill-Qwen-1.5B*, it increases average accuracy from 37.65 to 54.76. On the stronger *OpenMath-Nemotron-1.5B*, it improves average accuracy from 56.74 to 63.82, leading on eight of nine benchmarks. These results demonstrate its effectiveness across models with different initial capabilities. Among the compared methods, QuestA and PieceHint also incorporate COT during training. Unlike these methods using the fixed COT, PTPO utilzed theorem-grounded CoT and progressively adjusts the amount of guidance to the model's ability. Its superior performance further demonstrate the effectiveness of this progressive theorem-guided approach.

5.3 Scalability Beyond 1.5B Models

In this experiment, we examine whether the benefit of PTPO persists as model size increases, rather than being specific to the 1.5B setting. Table 2 shows that our PTPO consistently outperforms GRPO on all four benchmarks when using the same backbone. On Qwen3.5-2B, PTPO raises average accuracy from 38.03 to 49.54, a gain of 11.51 percentage points over GRPO. On the stronger Qwen3-8B, it obtains a performance gain of 2.52. The larger absolute improvement

Model	AIME24	AIME25	AMC23	MATH	Avg.
Qwen3.5-2B	8.89	12.22	36.80	72.20	32.53
Qwen3.5-2B + GRPO	19.60	16.86	43.44	72.20	38.03
Qwen3.5-2B + PTPO	**27.50**	**24.17**	**65.38**	**81.10**	**49.54**
Qwen3-8B	76.67	65.00	95.63	94.90	83.05
Qwen3-8B + GRPO	75.83	68.33	95.63	95.05	83.71
Qwen3-8B + PTPO	**79.50**	**72.66**	**96.62**	**96.15**	**86.23**

Table 2: Our proposed PTPO still obtain consistently performance gain using larger backbone of Qwen3.5-2B and Qwen3-8B.

on Qwen3.5-2B suggests that PTPO offers greater benefits to backbones with lower initial performance, while the consistent gains on Qwen3-8B demonstrate its value for larger and more capable models. These results demonstrate the scalability of proposed method beyond 1.5B models.

5.4 Ablation Studies

In this section, we conduct ablation studies on PTPO based on OpenMath-Nemotron-1.5B.

As shown in Table 3, removing either theorem-guided CoT or progressive RL degrades performance on all four benchmarks. Removing theorem-guided CoT reduces average accuracy from 80.77 to 77.59, while removing progressive RL lowers it to 76.76, demonstrating the importance of both components. Removing both components reduces PTPO to standard GRPO, which only achieves an average accuracy of 74.97. The full PTPO framework outperforms GRPO by 5.80 percentage points and achieves the best results, highlighting the benefit of combining theorem-guided CoT with progressive RL.

Theorem CoT	Progressive RL	AIME24	AIME25	AMC23	MATH	Avg.
✗	✗	64.09	52.39	90.55	92.86	74.97
✗	✓	67.01	56.83	93.50	93.01	77.59
✓	✗	66.58	55.10	92.13	93.22	76.76
✓	✓	**71.08**	**62.10**	**96.01**	**93.90**	**80.77**

Table 3: Ablation studies. Using both theorem-guided CoT and progressive RL achieves the best.

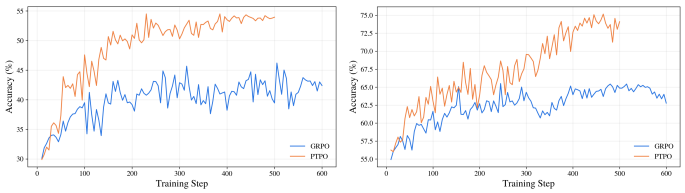
Recall that PTPO uses relevant mathematical theorems to construct a theorem-grounded CoT for each problem. To evaluate this design, we additionally compare it with two alternatives: generic CoT without explicit theorem and theorem-only hints that directly provide relevant theorems without a CoT. As shown in Table 4, our theorem-grounded CoT outperforms both alternatives on all four benchmarks, achieving an average accuracy of 80.77, compared with 79.02 for generic CoT and 75.66 for theorem-only hints. These results highlight the benefit of integrating theorem knowledge into explicit reasoning trajectories.

Method	AIME24	AIME25	AMC23	MATH	Avg.
Generic CoT	68.08	60.54	94.05	93.41	79.02
Theorem-only	65.15	53.17	91.47	92.83	75.66
Theorem CoT (ours)	**71.08**	**62.10**	**96.01**	**93.90**	**80.77**

Table 4: Performance comparison between our theorem-grounded CoT with other alternatives.

5.5 Faster Convergence and Higher Accuracy

Figure 4 illustrates the training accuracy curves of our proposed PTPO and the baseline GRPO on AIME 2024 using two different backbones. On both backbones, PTPO improves more rapidly during the early stages and converges to a higher accuracy plateau than GRPO. A possible explanation is that our progressive theorem-guided prompting provides explicit theorem knowledge when the model needs guidance and gradually reduces this support as its ability improves, facilitating the transition from guided reasoning to independent problem solving.



(a) DeepSeek-R1-1.5B (b) OpenMath-Nemotron-1.5B

Figure 4: Training accuracy curves on AIME 2024. PTPO converges faster and reaches higher final accuracy than GRPO.

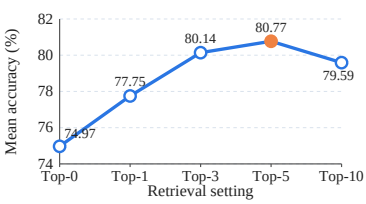


Figure 5: Effect of retrieval hyperparameter k.

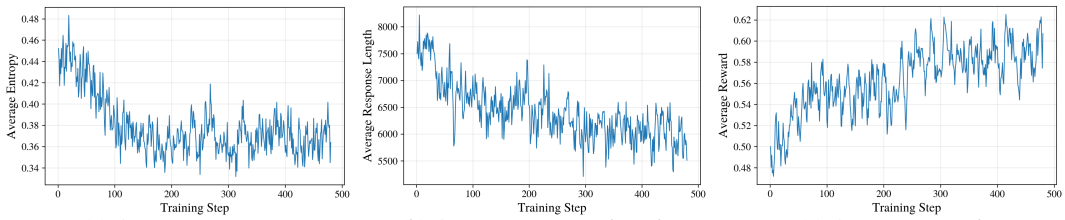
5.6 Theorem Retrieval Analysis

We investigate how theorem retrieval affects PTPO by varying the number of retrieved theorems and evaluating robustness to retrieval noise. Performance is measured by the average accuracy on four benchmarks: AIME24, AIME25, AMC23, and MATH.

Study on the number of retrieved theorems. Figure 5 illustrates the performance on the varying number of retrieved theorems with $k \in \{1, 3, 5, 10\}$. Performance improves as k increases from 1 to 5, but declines at $k = 10$. This trend suggests a trade-off between theorem coverage and retrieval noise: a small k may omit useful theorems, whereas an overly large k may introduce weakly relevant or irrelevant information despite improving coverage. Among the evaluated settings, $k = 5$ achieves the best balance and the highest accuracy.

Robustness to retrieval noise. To assess the impact of retrieval noise on PTPO, we replace the Top-5 retrieved theorems with five randomly selected theorems when constructing theorem-grounded CoTs. This replacement reduces average accuracy from 80.77 to 78.13, highlighting the importance of retrieving relevant theorems. Nevertheless, the random-theorem variant still outperforms GRPO by 3.16, suggesting that PTPO retains some robustness to imperfect retrieval. The detailed perfromance are reported in Appendix Table 6.

5.7 Training Dynamics Analysis



(a) Average entropy (b) Average response length (c) Average reward

Figure 6: As training progresses, the model shows decreasing entropy and response length while achieving increasing rewards, indicating more stable, concise, and effective reasoning behavior.

Figure 6 presents the evolution of policy entropy, response length, and average reward during reinforcement learning. The policy entropy gradually decreases in the early stage and then stabilizes within a relatively narrow range, indicating that the model shifts from broad exploration to more confident reasoning while still maintaining sufficient sampling diversity. Meanwhile, the average response length consistently decreases throughout training. This suggests that the model gradually internalizes theorem-aware reasoning patterns and becomes less dependent on externally injected theorem-grounded hints, producing more concise solutions without sacrificing performance. At the same time, the average reward steadily increases, showing stable optimization without severe reward collapse. The decreasing entropy and response length, together with the increasing reward, indicate that our method achieves a favorable balance between exploration, reasoning efficiency, and optimization stability, and effectively facilitates the transition from externally guided reasoning to autonomous theorem-aware problem solving.

6 Conclusion

We presented PTPO, a theorem-guided reinforcement-learning framework that adjusts cached reasoning guidance according to problem-level rollout performance. PTPO improves two 1.5B backbones across nine benchmarks, and controlled experiments quantify the effects of theorem grounding, progressive prompting, and theorem retrieval. Results on 2B and 8B backbones further demonstrate the effectiveness of PTPO across model scales. These findings support adaptive theorem-grounded guidance as an effective training signal for mathematical reasoning.

REPRODUCIBILITY STATEMENT

Section 3 describes construction of the theorem library and cached theorem-grounded CoTs, and Section 4 specifies the problem-level progressive prompting rule and reward. Section 5 reports the data, decoding settings, scalability results, controlled comparisons, training dynamics, and theorem-retrieval analysis. The appendices provide computational-cost details, the retrieval-evaluation protocol, prompt-construction details, and representative construction examples. Anonymous code and the theorem library are available at `https://anonymous.4open.science/r/PTPO-FED5`.

AI USE STATEMENT

Generative AI tools were used to assist with language editing and LaTeX formatting. The authors reviewed the resulting text and remain responsible for all technical content, experiments, citations, and claims. Language models used as components of the proposed data-construction and evaluation pipelines are described in the corresponding methodological and experimental sections.

REFERENCES

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. Large language models for mathematical reasoning: Progresses and challenges. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pp. 225–237. Association for Computational Linguistics, 2024.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *arXiv preprint arXiv:2505.23281*, 2025.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2318–2335, Bangkok, Thailand, 2024. Association for Computational Linguistics.

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. Theoremqa: A theorem-driven question answering dataset. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7889–7901, 2023.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Yangyi Fang, Jiaye Lin, Xiaoliang Fu, Cong Qin, and Haolin Shi. Placing puzzle pieces where they matter: A question augmentation framework for reinforcement learning. *arXiv preprint arXiv:2604.15830*, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Tarjei Paule Hage and Markus J. Buehler. BeamPERL: Parameter-efficient RL with verifiable rewards specializes compact LLMs for structured beam mechanics reasoning. *arXiv preprint arXiv:2603.04124*, 2026. URL `https://arxiv.org/abs/2603.04124`.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211.

Tao He, Rongchuan Mu, Lizi Liao, Yixin Cao, Ming Liu, and Bing Qin. Good learners think their thinking: Generative prm makes large reasoning model more efficient math learner. *arXiv preprint arXiv:2507.23317*, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

Jian Hu, Mingjie Liu, Shizhe Diao, Ximing Lu, Xin Dong, Pavlo Molchanov, Yejin Choi, Jan Kautz, and Yi Dong. ProRL V2: Prolonged training validates rl scaling laws. Notion Technical Report, August 2025. First published in August 2025.

Jian Hu, Mingjie Liu, Ximing Lu, Fang Wu, Zaid Harchaoui, Shizhe Diao, Yejin Choi, Pavlo Molchanov, Jun Yang, Jan Kautz, and Yi Dong. BroRL: Scaling reinforcement learning via broadened exploration. In *International Conference on Machine Learning*, 2026.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. *arXiv preprint arXiv:2206.14858*, 2022.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q. Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. Hugging Face repository, 2024. 13:9.

Jiazheng Li, Hong Lu, Kaiyue Wen, Zaiwen Yang, Jiaxuan Gao, Hongzhou Lin, Yi Wu, and Jingzhao Zhang. QuestA: Expanding reasoning capacity in llms via question augmentation. In *International Conference on Learning Representations*, 2026.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, Yansong Tang, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Erran Li, Raluca Ada Popa, and Ion Stoica. DeepScaleR: Surpassing o1-preview with a 1.5b model by scaling rl. Notion Blog, February 2025. DeepScaleR technical report.

Jordan Meadows and André Freitas. Introduction to mathematical language processing: Informal proofs, word problems, and supporting tasks. *Transactions of the Association for Computational Linguistics*, 11:1162–1184, 2023. doi: 10.1162/tacl_a_00594.

Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, Z. F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang, Yuxuan Liu, Wenjun Gao, Daya Guo, and Chong Ruan. DeepSeek-Prover-V2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*, 2025. URL https://arxiv.org/abs/2504.21801.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yu Sheng, Jing Li, and Dajun D. Zeng. Learning theorem rationale for improving the mathematical reasoning capability of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 15151–15159, 2025. doi: 10.1609/aaai.v39i14.33662.

Thang H. Trinh, Yuhuai Wu, Quoc V. Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625:476–482, 2024. doi: 10.1038/s41586-023-06747-5.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024. doi: 10.18653/v1/2024.acl-long.510.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Sean Welleck, Jiacheng Liu, Ronan Le Bras, Hannaneh Hajishirzi, Yejin Choi, and Kyunghyun Cho. Naturalproofs: Mathematical theorem proving in natural language. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

Sean Welleck, Jiacheng Liu, Ximing Lu, Hannaneh Hajishirzi, and Yejin Choi. Naturalprover: Grounded mathematical proof generation with language models. In *Advances in Neural Information Processing Systems*, 2022.

Zhen Hao Wong, Jingwen Deng, Runming He, Zirong Chen, Qijie You, Hejun Dong, Hao Liang, Chengyu Shen, Bin Cui, and Wentao Zhang. LogicPuzzleRL: Cultivating robust mathematical reasoning in LLMs via reinforcement learning. *arXiv preprint arXiv:2506.04821*, 2025. URL `https://arxiv.org/abs/2506.04821`.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus N. Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. In *Advances in Neural Information Processing Systems*, 2022.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, Zhenru Zhang, and Qwen Team. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. LeanDojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems*, 2023.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Li, Weinan Dai, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, and Yonghui Wu. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. Mammoth: Building math generalist models through hybrid instruction tuning. In *International Conference on Learning Representations*, 2024.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, 2022.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations*, 2023.

A Additional Experimental Details

A.1 Computational Cost

The problem-level success rate is computed from rollouts already generated during policy optimization and requires only counting the correct responses in each rollout group. Theorem retrieval and theorem-grounded CoT generation are performed once before reinforcement learning, and their outputs are cached for training. In our implementation, retrieving theorems for one query takes approximately 0.1 seconds. PTPO therefore introduces no online retrieval or additional CoT-generation calls during policy optimization.

A.2 Retrieval Evaluation Protocol

For each of the 100 sampled problems, DeepSeek-V4 is given the problem, its reference answer, and the Top-10 theorems returned by the hybrid retriever, and judges whether each theorem is directly relevant to solving the problem. Hit@k denotes the percentage of problems for which at least one relevant theorem appears among the Top-k results. Table 5 reports the complete retrieval-depth results under the same backbone, training data, progressive prompting strategy, and training budget.

Setting	Hit@k (%)	AIME24	AIME25	AMC23	MATH	Avg.
Top-0	–	64.09	52.39	90.55	92.86	74.97
Top-1	62	66.48	58.72	92.74	93.07	77.75
Top-3	82	70.45	61.53	94.81	93.76	80.14
Top-5	94	**71.08**	**62.10**	**96.01**	**93.90**	**80.77**
Top-10	97	70.22	60.73	94.17	93.25	79.59

Table 5: Complete retrieval-depth results. Top-0 uses no retrieved theorem; all settings use the same backbone, data, progressive prompt schedule, and training budget.

To evaluate robustness to retrieval noise, we replace the Top-5 retrieved theorems with five randomly sampled theorems. This comparison separates the benefit of accurate theorem retrieval from the more general effect of providing theorem-grounded reasoning context.

Method	AIME24	AIME25	AMC23	MATH-500	Avg.
GRPO	64.09	52.39	90.55	92.86	74.97
PTPO w/ Random-5	66.15	60.37	92.91	93.07	78.13
PTPO w/ Retrieved-5	**71.08**	**62.10**	**96.01**	**93.90**	**80.77**

Table 6: Robustness to theorem-retrieval noise. Random-5 replaces the retrieved theorem set with five randomly sampled theorems; all settings use the same backbone and training budget.

A.3 Theorem Use in Model Responses

To examine whether the performance gains of PTPO are accompanied by greater use of mathematical knowledge, we analyze model responses on AIME 2025. For each response, DeepSeek-V4 is given the problem, the reference answer, and the generated solution, and is asked to determine answer correctness and count the distinct mathematical theorems or principles used in the reasoning. We report the average theorem count only over responses judged correct, thereby avoiding the confounding effect of incomplete or failed solutions.

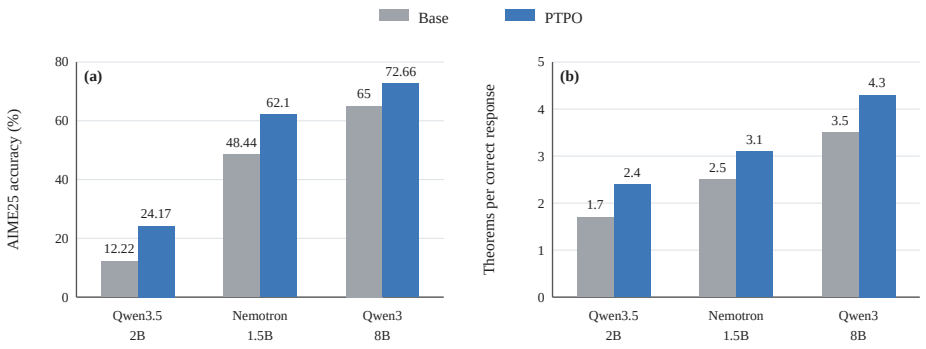


Figure 7: Accuracy and judge-estimated theorem use on AIME 2025. The theorem-use metric is the average number of distinct mathematical theorems or principles identified by DeepSeek-V4 in each correct response.

Figure 7 shows a consistent pattern across all three backbones: PTPO improves AIME 2025 accuracy while also increasing the average number of mathematical principles identified in correct responses. The theorem count rises from 1.7 to 2.4 on Qwen3.5-2B, from 2.5 to 3.1 on OpenMath-Nemotron-1.5B, and from 3.5 to 4.3 on Qwen3-8B. The persistence of this increase across substantially different model capacities suggests that PTPO does more than improve final-answer accuracy; its progressive theorem guidance is associated with broader use of relevant mathematical knowledge in successful solutions. The highest theorem count occurs on Qwen3-8B, indicating that a stronger backbone can combine its existing reasoning ability with the additional theorem-oriented training signal. Because this metric measures the breadth rather than the correctness of theorem application, we treat it as complementary evidence that PTPO encourages richer theorem-aware reasoning.

A.4 Extension to Open-Ended Proofs and Other Domains

The current implementation of PTPO assumes that the correctness of a generated answer can be automatically verified. This requirement does not necessarily restrict PTPO to conventional mathematical reasoning tasks. Open-ended proofs and cross-domain reasoning problems can also be incorporated when domain-specific tools provide reliable verification signals. For example, formal proofs can be checked exactly using Lean, as demonstrated by DeepSeek-Prover-V2 (Ren et al., 2025). LogicPuzzleRL (Wong et al., 2025) similarly employs rule-based verification for logic puzzles, while BeamPERL (Hage & Buehler, 2026) constructs exact rewards for physics problems using symbolic solvers.

Under such settings, the central training mechanism of PTPO can remain unchanged. The mathematical theorem library can be replaced with a domain-specific repository of formal rules, physical principles, logical constraints, or other reusable knowledge units. The answer verifier can likewise be replaced by the corresponding proof checker, symbolic solver, or rule-based evaluator. PTPO can then progressively reduce the injected domain knowledge according to the model's estimated ability, preserving its transition from externally guided reasoning to more autonomous problem solving.

These examples suggest that the applicability of PTPO depends primarily on the availability of structured domain knowledge and reliable verification tools, rather than on a specific mathematical task format. Extending PTPO to formal theorem proving, logical reasoning, and scientific problem solving is therefore a promising direction for future work. Nevertheless, constructing sufficiently precise knowledge repositories and robust automatic verifiers remains domain dependent and requires further investigation.

B PROMPT TEMPLATES

This section presents the complete prompts used for theorem extraction and theorem-grounded CoT construction. For readability and reproducibility, each prompt is divided into a reusable system instruction and an instance-specific user template. The placeholders are replaced with the corresponding problem, solution, and retrieved theorem descriptions during data construction.

B.1 THEOREM EXTRACTION PROMPT

We use the following prompts to convert each mathematical problem–solution pair into structured theorem entries. The system prompt specifies the extraction objective and output constraint, while the user prompt supplies the required schema and the source instance.

System Prompt for Theorem Extraction

```
You extract the mathematical theorems, laws, identities, or properties
    used explicitly or implicitly in a mathematical problem and its
    solution.

Return only a valid JSON object that follows the required schema. Do not
    add explanations outside the JSON.
```

User Prompt for Theorem Extraction

```
Read the following mathematical problem and its solution, and identify
    all mathematical principles used in the reasoning.

Output format:
{
  "theorem_id": {
    "name": "...",
    "statement": "...",
    "keywords": ["..."],
    "symbolic_form": "..."
  }
}

Problem:
{problem}

Solution:
{solution}
```

B.2 THEOREM-GROUNDED COT CONSTRUCTION PROMPT

System Prompt for Theorem-grounded CoT Construction

```
You are a mathematical and logical reasoning expert skilled at converting
     complex proofs and solution processes into structured, logically
    clear chains of thought.

Follow these steps:
1. Analyze the key information and goal of the problem.
2. Review the retrieved theorem descriptions and identify the theorems
    that are necessary or useful for solving the problem.
3. Decompose the original solution into clear, atomic reasoning steps.
4. Explicitly associate each key step with the theorem name and
    description that support it.
5. Output a structured theorem-grounded CoT solution.
```

User Prompt for Theorem-grounded CoT Construction

```
Problem:
{problem}

Relevant Theorems:
{theorem_descriptions}

Original Solution:
{original_cot}

Generate the theorem-grounded CoT solution.
```

The following prompts rewrite a reference solution as a logically structured reasoning trajectory with explicit theorem grounding. The system prompt defines the reasoning procedure, and the user prompt provides the problem, retrieved theorem descriptions, and original solution.

C Implementation Details

C.1 Theorem-Grounded CoT Construction

The following example illustrates how a reference solution is rewritten into a theorem-grounded CoT. The reference solution uses the Fundamental Theorem of Calculus and Rolle's Theorem only implicitly. The retrieved theorem entries serve as reasoning anchors, allowing the rewritten solution to explicitly identify where each theorem is applied rather than merely paraphrasing the original solution.

Example of Theorem-Grounded CoT Construction

1. Question

Let f be continuous on $[0,1]$ and differentiable on $(0,1)$. Suppose $f(0) = f(1) = 0$ and $\int_0^1 f(x)\,dx = 0$. Prove that there exist two distinct points $u, v \in (0,1)$ such that $f'(u) = f'(v) = 0$.

2. Original Solution

Define $F(x) = \int_0^x f(t)\,dt$. Then $F(0) = 0$ and $F(1) = \int_0^1 f(t)\,dt = 0$, so Rolle's theorem gives a point $c \in (0,1)$ with $F'(c) = 0$. Since $F'(x) = f(x)$, we have $f(c) = 0$. Applying Rolle's theorem to f on $[0,c]$ and $[c,1]$ gives $u \in (0,c)$ and $v \in (c,1)$ such that $f'(u) = f'(v) = 0$.

3. Retrieved Theorems

Rolle's Theorem

Name: Rolle's Theorem
Statement: If g is continuous on $[a,b]$, differentiable on (a,b), and $g(a) = g(b)$, then some $c \in (a,b)$ satisfies $g'(c) = 0$.
Symbolic form:
$g(a) = g(b) \Rightarrow \exists c \in (a,b) : g'(c) = 0$.
Keywords: continuity; differentiability; equal endpoint values; critical point; zero derivative.

Fundamental Theorem of Calculus

Name: Fundamental Theorem of Calculus
Statement: If f is continuous on $[a,b]$ and $F(x) = \int_a^x f(t)\,dt$, then F is differentiable and $F'(x) = f(x)$.
Symbolic form:
$F(x) = \int_a^x f(t)\,dt \Rightarrow F'(x) = f(x)$.
Keywords: integral function; derivative; continuity; antiderivative.

4. Theorem-grounded CoT

1. Define $F(x) = \int_0^x f(t)\,dt$.
2. Apply the Fundamental Theorem of Calculus to obtain $F'(x) = f(x)$, with $F(0) = 0$ and $F(1) = \int_0^1 f(x)\,dx = 0$.
3. Apply Rolle's Theorem to F on $[0,1]$ to obtain $c \in (0,1)$ with $F'(c) = 0$, and hence $f(c) = 0$.
4. Apply Rolle's theorem to f on $[0,c]$ to obtain $u \in (0,c)$ such that $f'(u) = 0$.
5. Apply Rolle's Theorem to f on $[c,1]$ to obtain $v \in (c,1)$ such that $f'(v) = 0$. Therefore $u \neq v$ and $f'(u) = f'(v) = 0$.

C.2 PROGRESSIVE PROMPT INJECTION

Algorithm 1 summarizes the progressive theorem-guided prompt injection procedure. PTPO maintains a question-level memory that records the rollout success rate of each problem in the preceding epoch. For a newly encountered problem, the success rate is initialized to 1, so the first rollout uses no theorem-grounded guidance.

After each rollout, the stored success rate is updated. When the same problem is revisited, a lower previous success rate results in a longer prefix of its cached theorem-grounded CoT, whereas a higher success rate provides less guidance. The prompt ratio is used only to construct the next-round prompt; the policy is optimized using the correctness and format rewards defined in Section 4.

Algorithm 1 Progressive theorem-guided prompt injection

Require: policy model π_θ; training problems $\mathcal{Q}$; theorem-grounded CoT repository $\mathcal{R}$; correctness verifier V_{correct}; format verifier V_{format}; rollout number n; format-reward coefficient β.

Initialize question-level memory $M \leftarrow \varnothing$
for each training iteration **do**
  Sample a mini-batch $\mathcal{B} \subseteq \mathcal{Q}$
  for each problem $q \in \mathcal{B}$ **do**
    Retrieve the cached theorem-grounded CoT s_q from $\mathcal{R}$
    Load the previous success rate α_q from M; if absent, set $\alpha_q \leftarrow 1$
    Compute the prompt ratio $\rho_q \leftarrow 1 - \alpha_q$
    Compute the injected length $\ell_q \leftarrow \lfloor \rho_q \, \ell(s_q) \rfloor$
    Truncate s_q to its prefix $s_q^{(\ell_q)}$
    Construct the prompted input $\widetilde{q} \leftarrow \text{Prompt}\left(q, s_q^{(\ell_q)}\right)$
    Sample n responses $\{y_1, \ldots, y_n\} \sim \pi_\theta(\cdot \mid \widetilde{q})$
    for $i \leftarrow 1, \ldots, n$ **do**
      Compute correctness $c_i \leftarrow V_{\text{correct}}(y_i, a_q)$
      Compute format validity $f_i \leftarrow V_{\text{format}}(y_i)$
      Compute reward $R_i \leftarrow c_i + \beta f_i$
    end for
    Update the rollout success rate $\alpha_q \leftarrow \frac{1}{n} \sum_{i=1}^{n} c_i$
    Update memory $M[q] \leftarrow \alpha_q$
  end for
  Compute group-relative advantages from $\{R_i\}_{i=1}^{n}$
  Update π_θ using group-based policy optimization
end for
Output: updated policy model π_θ

D CASE STUDIES

We provide representative examples from AIME 2024 and AIME 2025 to illustrate the theorem-aware reasoning behavior of PTPO. In both cases, PTPO explicitly invokes a relevant mathematical principle at a key intermediate step and reaches the correct final answer.

D.1 EXAMPLE FROM AIME 2024

Question

Find the largest possible real part of

$$(75+117i)z+\frac{96+144i}{z},$$

where z is a complex number with $|z|=4$.

Model Output by PTPO

Write $z=4e^{i\theta}$, where $\theta\in\mathbb{R}$. Substitution gives

$$(75+117i)z+\frac{96+144i}{z}=4(75+117i)e^{i\theta}+\frac{24+36i}{e^{i\theta}}.$$

Let $A=300+468i$ and $B=24+36i$. The expression is therefore

$$Ae^{i\theta}+Be^{-i\theta}.$$

Key theorem step. Using Euler's formula, $e^{i\theta}=\cos\theta+i\sin\theta$ and $e^{-i\theta}=\cos\theta-i\sin\theta$. Hence,

$$(300+468i)(\cos\theta+i\sin\theta)+(24+36i)(\cos\theta-i\sin\theta),$$

whose real part is $324\cos\theta-432\sin\theta$. The maximum of $a\cos\theta+b\sin\theta$ is $\sqrt{a^2+b^2}$, so

$$\max\mathrm{Re}(\cdot)=\sqrt{324^2+(-432)^2}=\sqrt{291600}=\boxed{540}.$$

D.2 EXAMPLE FROM AIME 2025

Question

In $\triangle ABC$, points D and E lie on $\overline{AB}$ so that $AD < AE < AB$, while points F and G lie on $\overline{AC}$ so that $AF < AG < AC$. Suppose

$$AD = 4, \quad DE = 16, \quad EB = 8,$$
$$AF = 13, \quad FG = 52, \quad GC = 26.$$

Let M be the reflection of D through F, and let N be the reflection of G through E. The area of quadrilateral $DEGF$ is 288. Find the area of heptagon $AFNBCEM$.

Model Output by PTPO

Place $A = (0, 0)$, $B = (28, 0)$, and $C = (c_x, c_y)$. Since $AC = 91$,

$$c_x^2 + c_y^2 = 91^2.$$

From the given lengths,

$$D = (4, 0), \qquad E = (20, 0),$$

and since $AF = 13$, $AG = 65$, and $AC = 91$,

$$F = \left(\frac{c_x}{7}, \frac{c_y}{7}\right), \qquad G = \left(\frac{5c_x}{7}, \frac{5c_y}{7}\right).$$

Key theorem step. Using the shoelace formula on quadrilateral $DEGF$,

$$[DEGF] = \frac{1}{2}\left|20 \cdot \frac{5c_y}{7} - 4 \cdot \frac{c_y}{7}\right| = \frac{48c_y}{7}.$$

Since $[DEGF] = 288$, we obtain $c_y = 42$. Therefore,

$$c_x^2 = 91^2 - 42^2 = 6517 = 49 \cdot 133, \qquad c_x = 7\sqrt{133}.$$

Thus,

$$F = (\sqrt{133}, 6), \qquad G = (5\sqrt{133}, 30).$$

Reflecting D through F and G through E gives

$$M = (2\sqrt{133} - 4, 12), \qquad N = (40 - 5\sqrt{133}, -30).$$

The vertices of the heptagon are

$$A(0, 0), \quad F(\sqrt{133}, 6), \quad N(40 - 5\sqrt{133}, -30),$$
$$B(28, 0), \quad C(7\sqrt{133}, 42), \quad E(20, 0),$$
$$M(2\sqrt{133} - 4, 12).$$

Applying the shoelace formula once more yields

$$[AFNBCEM] = \boxed{588}.$$