

TemplateRL: Structured Template-Guided Reinforcement Learning for LLM Reasoning

Jinyang Wu^{1*}, Chonghua Liao^{1*}, Mingkuan Feng^{1*}, Shuai Zhang¹,
Zhengqi Wen⁵, Haoran Luo², Ling Yang³, Huazhe Xu^{1,4}, Jianhua Tao^{1,5}

¹Tsinghua University, ²Nanyang Technological University,

³Princeton University, ⁴Shanghai AI Lab,

⁵Beijing National Research Center for Information Science and Technology

{wu-jy23, lch22}@mails.tsinghua.edu.cn

Abstract

Reinforcement learning (RL) has emerged as an effective paradigm for enhancing model reasoning. However, existing RL methods like GRPO typically rely on unstructured self-sampling to fit scalar rewards, often producing inefficient rollouts that fail to capture transferable problem-solving strategies. To address this limitation, we propose **TemplateRL**, a structured template-guided RL framework that augments policy optimization with explicit template guidance. Our approach first constructs a problem-solving template library via MCTS on a small seed set, then seamlessly integrates this high-level structured guidance into RL training. By guiding rollout generation to align with proven template structures, TemplateRL significantly improves high-quality trajectory hit rates while reducing ineffective exploration. This structure-guided design steers the policy toward validated strategic patterns, stabilizing training dynamics, and enhancing RL sampling efficiency. Notably, the explicit template library is interpretable, editable, and supports online updates-enabling continuous updates during both training and inference. Extensive experiments demonstrate that TemplateRL outperforms GRPO by 99% on AIME and 41% on AMC, with superior stability on weak models and remarkable cross-domain generalization, highlighting its potential for broader tasks.

1 Introduction

Reinforcement learning (RL) has demonstrated remarkable success in enhancing the reasoning capabilities of large language models (LLMs), as exemplified by OpenAI-o1 (Jaech et al., 2024), DeepSeek-R1 (Guo et al., 2025), and Kimi-1.5 (Team et al., 2025). In contrast to traditional approaches that rely on human-curated annotations (Achiam et al., 2023; Grattafiori et al., 2024), contemporary RL training paradigms (Shao et al.,

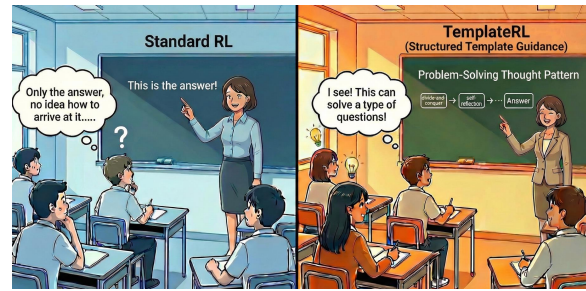


Figure 1: **Paradigm comparison: Teacher-Student Analogy for RL.** Standard RL like GRPO (Left) provides only sparse answer rewards, while TemplateRL (Right) offers structured templates encoding problem-solving thought patterns, enabling effective learning of both concrete steps and underlying strategic logic.

2024) directly optimize base models using automatically computable reward signals. This enables models to develop sophisticated capabilities, including problem decomposition and self-reflection (Gandhi et al., 2025; Yue et al., 2025).

Recent RL research primarily focuses on three directions: (1) Algorithmic Refinements that address inherent limitations such as length bias, KL divergence constraints, and advantage estimation (Yu et al., 2025; Liu et al., 2025); (2) Data-Level Optimizations that reduce annotation dependencies and improve data organization to enable minimally supervised training (Zuo et al., 2025; Wang et al., 2025b); (3) Exploration Enhancement through entropy regularization or token-level learning to encourage distributional diversity (Wang et al., 2025a; Cui et al., 2025b).

While promising, they remain fundamentally constrained by unstructured self-sampling within the model’s internal distribution, lacking mechanisms to leverage structured guidance for more effective policy learning. This leads to three critical limitations: (a) **Inefficient trajectory sampling:** without structured guidance, models struggle to consistently generate high-quality trajectories, resulting in low hit rates and training instability on

* Equal Contribution

weak models; **(b) Difficulty in learning transferable strategies:** existing RL methods tends to implicitly learn from surface-level steps rather than extract generalizable high-level thought patterns (e.g., divide-and-conquer), hindering knowledge transfer and accumulation across domains; **(c) Lack of interpretability:** existing RL optimization often produce reasoning trajectories without explicit strategic structure, hindering error diagnosis and expert intervention in critical applications.

To this end, we propose **TemplateRL**, a novel template-guided reinforcement learning framework that elegantly integrates structured thought templates into policy optimization (Figure 1). Our approach consists of three parts: **(1) template construction:** We employ Monte Carlo Tree Search on a small seed set to explore diverse solutions, then abstract successful paths into structured templates encoding high-level problem-solving patterns; **(2) template-guided training:** During RL training, we adaptively retrieve relevant templates as structured guidance for rollout generation, steering the policy toward proven strategic patterns; **(3) optional template updates:** The template library supports continuous updates by automatically abstracting new patterns from successful RL rollouts. This design delivers three key advantages: For **(a)**, structured guidance improves solution hit rates and stabilizes training on weaker models; For **(b)**, explicit templates enable learning both concrete steps and underlying strategic logic, facilitating cross-domain generalization and flexible knowledge updates; For **(c)**, template-guided trajectories form interpretable decision chains that support error diagnosis and expert refinement.

Extensive experiments validate TemplateRL’s effectiveness across four dimensions: **(1) Improved Performance:** TemplateRL significantly outperforms GRPO by 99% on AIME, and 41% on AMC on a 7B backbone, surpassing representative baselines across benchmarks, model scales and architectures; **(2) Enhanced Training Stability:** Unlike GRPO which easily collapses on weaker models (e.g., Llama-3.2-3B), TemplateRL achieves stable training dynamics; **(3) Broad Generalization:** Our method demonstrates strong cross-domain performance on out-of-distribution benchmarks, including BALROG (agent), GPQA-D (science), and MMLU-Pro (knowledge). It consistently improves across different modalities (text, vision); **(4) Dynamic Extensibility:** Template library supports continuous updates during both training and infer-

ence, enabling incremental strategy incorporation. Further analysis reveal that TemplateRL produces clear and interpretable reasoning trajectories.

2 Preliminaries

LLMs as Markov Decision Processes. LLM generation can be formulated as a token-level Markov Decision Process $\mathcal{M}(\mathcal{S}, \mathcal{A}, r, p_{\mathcal{Q}})$ (Sutton et al., 1998; Liu et al., 2025), where \mathcal{S} represents states (observation sequences) and \mathcal{A} denotes the action space (vocabulary). At each step t , state $s_t \in \mathcal{S}$ consists of the input question \mathbf{q} concatenated with all tokens generated so far $\mathbf{o}_{<t}$. This state serves as input to policy model $\pi_{\theta}(\cdot|s_t)$. Specifically, the policy processes $s_t = (\mathbf{q}, \mathbf{o}_{<t}) = (q_1, q_2, \dots, q_l, o_1, o_2, \dots, o_{t-1})$, where q_i denotes the i -th token of question \mathbf{q} and $o_{j,<t}$ represents the token generated by π_{θ} at step j . In the RL framework, the policy then samples the next token from \mathcal{A} . The entropy-regularized optimization objective (Schulman et al., 2017) is:

$$\mathcal{J}(\pi_{\theta}) = \mathbb{E}_{\mathbf{q} \sim p_{\mathcal{Q}}} \left[\mathbb{E}_{\mathbf{o} \sim \pi_{\theta}(\cdot|\mathbf{q})} [R(\mathbf{q}, \mathbf{o})] - \beta \cdot \mathbb{D}_{KL}[\pi_{\theta}(\cdot|\mathbf{q})] \|\pi_{\text{ref}}(\cdot|\mathbf{q})\right], \quad (1)$$

where $R(\mathbf{q}, \mathbf{o}) = \sum_{t=1}^{|\mathbf{o}|} r(s_t, o_t)$ is the return of trajectory $(\mathbf{q}; \mathbf{o})$, $r(\cdot)$ is the reward model, and π_{ref} is a reference policy. The KL regularization term prevents π_{θ} from deviating excessively from π_{ref} .

Template Action. A template action is a prompt a designed to elicit a specific type of reasoning from a model, given a partial solution as input. It encapsulates an abstract human-reasoning procedure, such as proposing the next sub-question or generating the subsequent inference step (Guan et al., 2025). More precisely, given a partial solution \mathbf{p} , a induces the policy model π_{θ} to generate the next partial solution \mathbf{p}' :

$$\mathbf{p}' = \pi_{\theta}(a(\mathbf{p})). \quad (2)$$

3 Template-Guided RL

As shown in Figure 2, TemplateRL consists of three parts: (1) template construction (Section 3.1); (2) template-guided training (Section 3.2); and (3) optional template updates (Section 3.3).

3.1 Template Construction

We first describe how to construct a template library, which guides RL training in Section 3.2.

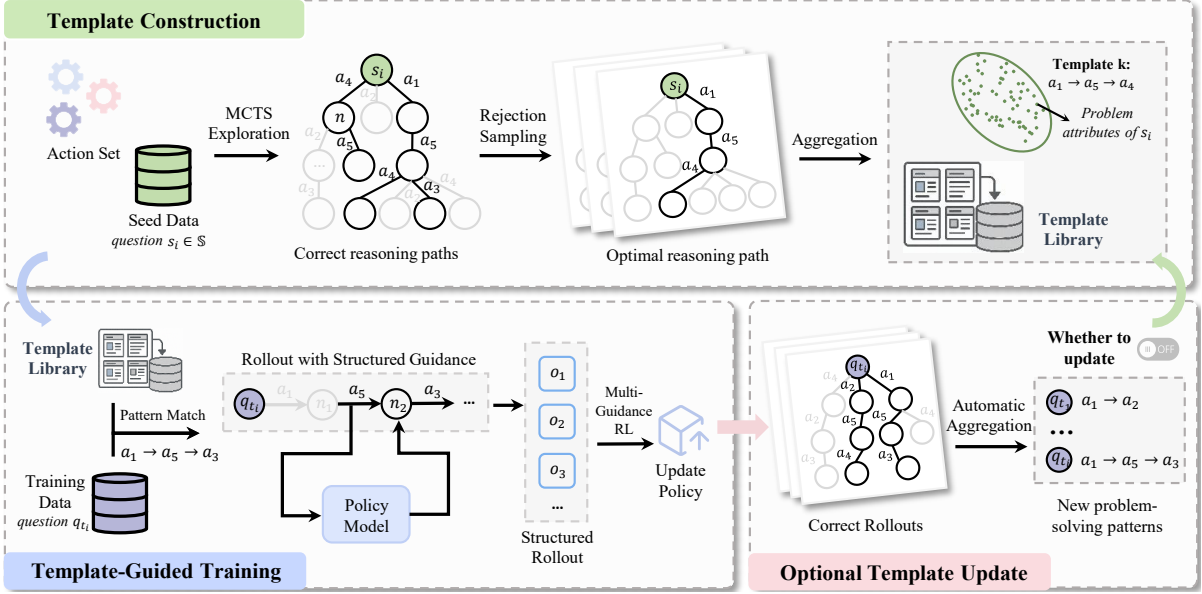


Figure 2: **Flowchart of TemplateRL.** This framework consists of three components: (1) template construction (Section 3.1); (2) template-guided training (Section 3.2); and (3) optional template updates (Section 3.3).

Previous work (Kahneman, 2011) reveals that humans solve complex reasoning tasks by applying universal guidelines (“templates”) induced from similar problems rather than starting from scratch. These high-level templates help address unfamiliar tasks by leveraging previously successful strategies. Inspired by this, we introduce “template library”, a lightweight hub of structured templates that provide high-level strategic guidance for RL.

1 MCTS Exploration. Starting with a small seed set $\mathcal{S} = \{s_1, \dots, s_s\}$, we employ Monte Carlo Tree Search (MCTS) (Kocsis and Szepesvári, 2006a) to generate solution trees. For each question $s_i \in \mathcal{S}$, given a predefined action set $\mathcal{A} = \{a_1, \dots, a_{|\mathcal{A}|}\}$ and model π_θ , MCTS build a search tree \mathcal{T}_i where: the root node represents question s_i , each edge denotes an action $a \in \mathcal{A}$, and each child node n contains partial solutions generated by π_θ under the corresponding action. A path from root s_i to leaf node $\mathbf{n}_{i,j,d_{i,j}}$ forms a solution trajectory:

$$\mathbf{t}_{i,j} = (s_i, a_{i,j,1}, \mathbf{n}_{i,j,1}, \dots, a_{i,j,d_{i,j}}, \mathbf{n}_{i,j,d_{i,j}}), \quad (3)$$

Each intermediate node $\mathbf{n}_{i,j,\ell}$ is generated based on the cumulative context of its parent nodes and current action:

$$\mathbf{n}_{i,j,\ell} = \pi_\theta([s_i, a_{i,j,1}, \mathbf{n}_{i,j,1}, \dots, a_{i,j,\ell}]). \quad (4)$$

Through this process, for each seed question s_i , we obtain multiple solution traces $\mathbb{T}_i =$

$\{\mathbf{t}_{i,1}, \dots, \mathbf{t}_{i,|\mathbb{T}_i|}\}$ (e.g., Figure 3). The MCTS algorithm assigns a final reward $R(\mathbf{t}_{i,j} | s_i)$ to each trace $\mathbf{t}_{i,j} \in \mathbb{T}_i$. More details are in Appendix C.

2 Rejection Sampling. After MCTS exploration, we obtain multiple solution traces for each seed question s_i . To identify the optimal path, we use a simple path selection metric from Wu et al. (2024):

$$\text{Score}(s_i, \mathbf{t}_{i,j}) = b \cdot R(\mathbf{t}_{i,j} | s_i) - (1 - b) \cdot C(\mathbf{t}_{i,j}), \quad (5)$$

where $C(\mathbf{t}_{i,j})$ denotes path complexity (e.g. action count, $C(\mathbf{t}_{i,j}) = d_{i,j}$), and $b \in [0, 1]$ balances solution quality and complexity. This scoring function selects paths that maximize accuracy while maintaining procedural conciseness. For each question $s_i \in \mathcal{S}$, we select the optimal trace via:

$$\mathbf{t}_{i,\text{best}} = \arg \max_{\mathbf{t}_{i,j} \in \mathbb{T}_i} \text{Score}(s_i, \mathbf{t}_{i,j}). \quad (6)$$

3 Aggregation. Since each node in $\mathbf{t}_{i,\text{best}}$ corresponds to an instantiated action $a_{i,\ell} \in \mathcal{A}$, we retain the action sequence as a high-level thought template $T_i = (a_{i,1}, \dots, a_{i,d_{i,\text{best}}})$. We then aggregate identical patterns to construct a template library $\mathcal{L} = \{\hat{T}_1, \dots, \hat{T}_{|\mathcal{L}|}\}$. Aggregation is guided by Problem Condition Complexity (PCC) (Lee and Heyworth, 2000), defined as the number of prior conditions in s_i ; we obtain $\text{PCC}(s_i) \in \mathbb{R}_{\geq 0}$ by prompting π_θ with s_i . Each template stores both a

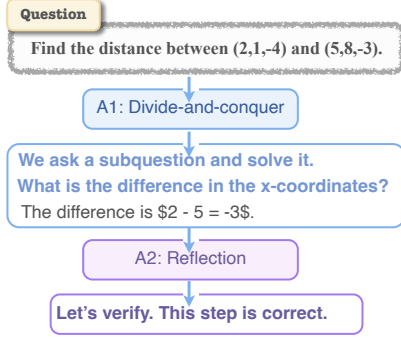


Figure 3: Structure of action-chain solution trajectories.

structured thought pattern (e.g., $a_1 \rightarrow a_2$) and the average PCC of questions sharing this pattern:

$$\hat{T}_j = (\text{PCC}_{T_j}, T_j), \text{PCC}_{T_j} = \frac{1}{|\mathcal{I}_j|} \sum_{i \in \mathcal{I}_j} \text{PCC}(\mathbf{s}_i). \quad (7)$$

where $\mathcal{I}_j = \{i : T_i = T_j\}$. These templates represent generalized problem-solving strategies and serve as structured guidance in Section 3.2. More details are in Appendix C.

Notably, while we use MCTS to build reasoning trees and abstract them into structured templates, such templates can also be obtained through other ways, such as expert-crafted general solutions for problem categories. We leave this for future work.

3.2 Template-Guided Training

Taking GRPO as an example, we now integrate high-level structured template guidance into RL to enhance policy learning. We exploit the observation that for multi-step reasoning, models are more likely to generate one correct step than to complete the entire reasoning steps in a single inference.

Rollout with Structured Guidance. Following meta-reasoning principles (Russell and Wefald, 1991), we adaptively retrieve relevant templates for each training question \mathbf{q} . We compute its $\text{PCC}(\mathbf{q})$, define the distance $d(\mathbf{q}, \hat{T}_j) = |\text{PCC}(\mathbf{q}) - \text{PCC}_{T_j}|$ to each template $\hat{T}_j \in \mathcal{L}$, and select the top- k most similar $\{\hat{T}_{i_1}, \dots, \hat{T}_{i_k}\}$.

Given policy π_θ , query \mathbf{q} , and a retrieved template $T_i = (a_1, \dots, a_{d_i})$, we roll out iteratively. Let \oplus denote sequence concatenation. Define the history $\mathbf{p}_0 = \mathbf{q}$ and for $\ell = 1, \dots, d_i$,

$$\mathbf{s}_\ell \sim \pi_\theta(\cdot \mid \mathbf{p}_{\ell-1}, a_\ell), \mathbf{p}_\ell = \mathbf{p}_{\ell-1} \oplus a_\ell \oplus \mathbf{s}_\ell. \quad (8)$$

The final template-guided rollout is $\mathbf{o} = \mathbf{p}_{d_i}$.

Multi-Guidance RL. Taking GRPO as an example, we now investigate how multiple template guidance decomposes RL’s learning objective into templated sub-objectives, and elucidate how templates enhance sampling quality and RL training.

As illustrated above, given diverse templates $\{T_1, \dots, T_{|g|}\}$, we independently sample G_i rollouts $\{\mathbf{o}_{i,1}, \dots, \mathbf{o}_{i,G_i}\}$ for each template T_i and use all rollouts to update the policy. Omitting KL term for simplicity, the optimization objective becomes

$$\tilde{\mathcal{J}}_{\text{GRPO}}(\pi_\theta) = \frac{1}{\sum_{i=1}^{|g|} G_i} \sum_{i=1}^{|g|} \sum_{j=1}^{G_i} \frac{1}{|\mathbf{o}_{i,j}|} \sum_{t=1}^{|\mathbf{o}_{i,j}|} \quad (9)$$

$$\min [\rho_{i,j,t} A_{i,t}, \hat{\rho}_{i,j,t} A_{i,t}],$$

where the probability ratio is $\rho_{i,j,t} = \frac{\pi_\theta(\mathbf{o}_{i,j,t} \mid \mathbf{q}, \mathbf{o}_{i,j,<t})}{\pi_{\theta_{\text{old}}}(\mathbf{o}_{i,j,t} \mid \mathbf{q}, \mathbf{o}_{i,j,<t})}$, and the clipped ratio is $\hat{\rho}_{i,j,t} = \text{clip}(\rho_{i,j,t}; 1 - \epsilon, 1 + \epsilon)$.

For convenience, define the template-specific objective as:

$$\mathcal{J}_i(\pi_\theta) = \frac{1}{G_i} \sum_{j=1}^{G_i} \frac{1}{|\mathbf{o}_i|} \sum_{t=1}^{|\mathbf{o}_i|} \min [\rho_{i,j,t} A_{i,t}, \hat{\rho}_{i,j,t} A_{i,t}], \quad (10)$$

Then, Equation (9) can be rewritten as

$$\tilde{\mathcal{J}}_{\text{GRPO}}(\pi_\theta) = \frac{1}{\sum_{i=1}^{|g|} G_i} \sum_{i=1}^{|g|} G_i \mathcal{J}_i(\pi_\theta). \quad (11)$$

which shows that template-guided training optimizes the model across multiple structured strategic patterns, with G_i weighting each template’s contribution. We provide theoretical analysis below, and detailed proofs in Appendix B.

Proposition 3.1 (Stability and positive-sample guarantee. Proof in Appendix B.2). *Consider guided GRPO with $|g|$ independent guidance groups, each containing $G_i = G/|g|$ trajectories. If the per-trajectory positive-advantage probability in template group i is p_i , then the probability of at least one positive trajectory is*

$$P_{\text{pos}} = 1 - P_0 = 1 - \prod_{i=1}^{|g|} (1 - p_i)^{G_i}. \quad (12)$$

Since $\log(1 - p)$ is concave, we have $\prod_i (1 - p_i)^{G_i} \leq (1 - \bar{p})^G$ for $\bar{p} = \frac{1}{|g|} \sum_i p_i$. Hence, grouping increases P_{pos} and reduces gradient variance, leading to improved training stability compared with the unguided setting.

Model	MATH500 \uparrow	AIME24 \uparrow	AMC \uparrow	Minerva \uparrow	Olympiad \uparrow	Avg. \uparrow
Qwen2.5-Math-7B-Base (Yang et al., 2024a)	50.8	13.3	42.5	12.1	17.2	27.2
Qwen2.5-Math-7B-Ins. (Yang et al., 2024a)	81.0	13.3	55.0	32.7	38.8	44.1
SimpleRL-Zero (Zeng et al., 2025)	74.6	26.7	60.0	27.6	35.8	44.9
OpenReasoner-Zero (Hu et al., 2025a)	81.0	16.7	57.5	32.7	43.2	46.2
PRIME-Zero (Cui et al., 2025a)	79.0	20.0	60.0	36.4	40.6	47.2
Oat-Zero (Liu et al., 2025)	79.6	30.0	60.0	34.2	39.9	48.7
RLOO (Ahmadian et al., 2024)	73.8	30.0	50.0	35.5	36.0	45.1
TemplateRL (Ours)	78.8	33.3	67.5	36.7	41.2	51.5*
Δ (\uparrow)	+6.8%	+11.0%	+35.0%	+3.4%	+14.4%	+14.2%
GRPO (Shao et al., 2024)	76.2	16.7	55.0	32.7	38.1	43.8
TemplateRL (Ours)	83.4	33.3	77.5	38.2	46.2	55.8*
Δ (\uparrow)	+9.4%	+99.4%	+40.9%	+16.8%	+21.2%	+27.4%

Table 1: **Performance comparison on Qwen2.5-Math-7B-Base.** The best results on each benchmark are highlighted in **bold**. * represents significantly better than baselines ($p < 0.05$).

Proposition 3.2 (Template transfer improves success probability. Proof in Appendix B.3). Let $T^*(\mathbf{q}')$ denote a template obtained from MCTS for a seed question \mathbf{q}' with positive advantage. If the advantage function $A(\tau; \mathbf{q})$ is L -Lipschitz continuous in \mathbf{q}

$$A(T^*(\mathbf{q}'); \mathbf{q}) \geq A(T^*(\mathbf{q}'); \mathbf{q}') - Ld(\mathbf{q}, \mathbf{q}'), \quad (13)$$

then for any new query \mathbf{q} satisfying

$$A(T^*(\mathbf{q}'); \mathbf{q}') > Ld(\mathbf{q}, \mathbf{q}'), \quad (14)$$

we have $A(T^*(\mathbf{q}'); \mathbf{q}) > 0$. Therefore, retrieving a high-quality template from similar questions yields a mini-group on \mathbf{q} , with positive advantage probability $r_h > p_h^{\text{policy}}$ (random rollouts).

3.3 Optional Template Update

During RL training, we optionally leverage correct rollouts to continually enrich the template library. Specifically, given a correct rollout \mathbf{o} , we apply keyword-based or lightweight model-based pattern extraction to automatically identify the underlying action sequence (template) $T' = (a'_1, \dots, a'_d)$. Such templates are then incorporated into the library for continual expansion. We empirically validate the effectiveness of dynamic template updates during both training and inference in Section 4.5.

4 Experiments

This section presents experimental settings, results, and analysis. We analyze TemplateRL from five aspects: (1) improved performance, (2) enhanced training stability, (3) broad generalization, (4) dynamic extensibility, and (5) ablation study.

4.1 Experimental Setup

Training Datasets. We use training data exclusively from MATH (Hendrycks et al., 2021) level 3-5 problems, yielding 5.5K examples. We randomly sample 500 instances to construct the template library, with the remaining 5K for training.

Evaluation. We evaluate on widely used reasoning benchmarks, including MATH500 (Hendrycks et al., 2021), AIME 2024 (Li et al., 2024), AMC (Li et al., 2024), Minerva (Lewkowycz et al., 2022), OlympiadBench (He et al., 2024), GSM8K (Cobbe et al., 2021), College Math (Tang et al., 2024), and Gaokao23 (Zhang et al., 2023). Since our training data focuses on math, we further assess out-of-domain generalization on BALROG (Paglieri et al., 2025) for agent tasks, GPQA-D (Rein et al., 2024) for graduate-level science, and MMLU-Pro (Wang et al., 2024b) for general knowledge answering.

Baseline Methods. We benchmark TemplateRL with representative baselines, such as *Standard GRPO* (Shao et al., 2024) and *Oat-Zero* (Liu et al., 2025). More details are provided in Appendix D.

Implementation Details. Following previous work (Cui et al., 2025a; Liu et al., 2025), we use Qwen2.5-Math-7B-Base (Yang et al., 2024a) as the default model. For RL training, we set $\beta = 0$ (no KL loss) and employ Dr.GRPO loss (Liu et al., 2025). Our configuration includes batch size 128, 16 samples per question, and 2 guidance templates ($|g| = 2$). The reward is binary accuracy verified by Math-Verify. We train for 500 steps on 8 A100 GPUs. More details are provided in Appendix D.

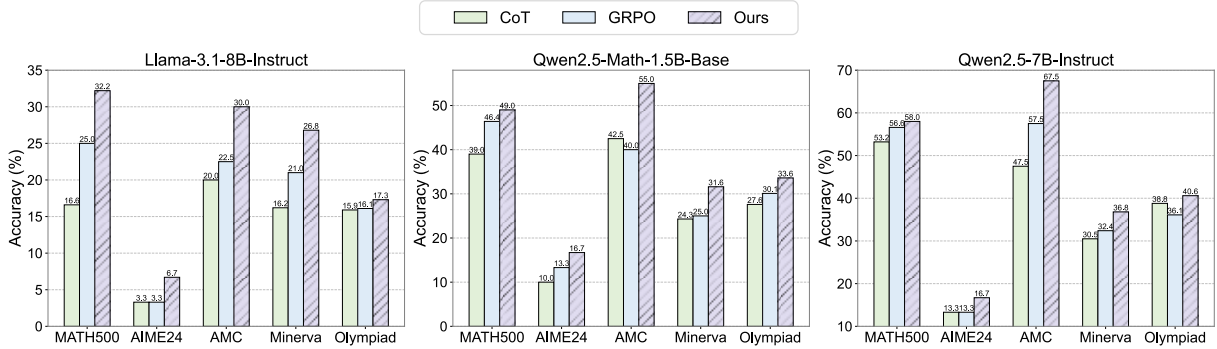


Figure 4: **Performance comparison across different model scales and architectures.** For better visualization, MATH500 results are adjusted by subtracting 20 points for all models.

4.2 Improved Performance

Main Results. Table 1 presents performance comparisons across five competition-level reasoning benchmarks. The results reveal three findings:

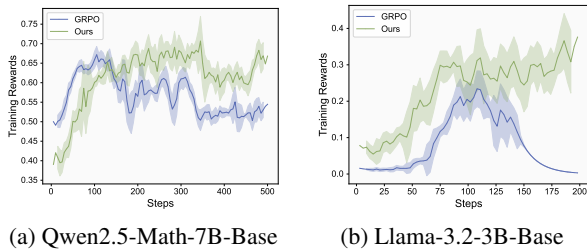


Figure 5: **Training stability verification.** We evaluate on Qwen2.5-Math-7B-Base and Llama-3.2-3B-Base.

◇ *TemplateRL Significantly Outperforms Baselines.* TemplateRL achieves an average score of 55.8, outperforming the best baseline (Oat-Zero) by 7.1 points and GRPO by 12.0 points. On AIME24, TemplateRL achieves 33.3% versus GRPO’s 16.7% (+99.4% relative gain), demonstrating the effectiveness of structure-guided policy optimization.

◇ *Greater Improvements on More Challenging Benchmarks.* Performance gains scale with task difficulty: MATH500 shows +9.4% improvement, while more challenging AMC and AIME24 exhibit +40.9% and +99.4%, respectively. This suggests explicit template provide greater benefits for problems requiring complex strategic decomposition.

◇ *Structure-Guided Training Generalizes Across RL Algorithms.* When applied to other RL algorithms (e.g., RLOO), TemplateRL maintains substantial improvements. This consistency further confirms that structure-guided training is a general approach applicable to diverse RL algorithms.

Scalability Across Different Models. As shown in Figure 4, TemplateRL maintains consistent improvements across: (1) different scales, from 1.5B to 8B parameters; (2) diverse architectures, both

Qwen and Llama families show substantial improvements; and (3) various model types, both base models and instruction-tuned models benefit from structure-guided training (details in Table 6). We further validate template guidance on larger models (32B) in Appendix E.5, demonstrating that structured guidance enhances rather than restricts exploration even for highly capable models.

4.3 Enhanced Training Stability

We analyze training stability by comparing reward curves on Qwen2.5-Math-7B-Base and Llama-3.2-3B-Base. As shown in Figure 5, TemplateRL achieves higher training rewards than GRPO on both models overall. Critically, GRPO exhibits training collapse after 100 steps on Llama-3.2-3B-Base, with rewards dropping from 0.3 to near 0.0. This aligns with prior findings (Gandhi et al., 2025; Liu et al., 2025) that GRPO frequently fails on weaker models. In contrast, TemplateRL maintains stable dynamics, with rewards above 0.25.

TemplateRL’s enhanced stability mainly stems from its structured template guidance during RL sampling. By providing explicit strategic patterns, templates equips weaker models with advanced decomposition capabilities typically exclusive to stronger models. This enables effective learning from challenging examples, which would otherwise yield sparse positive signal under standard GRPO training, as detailed in Section 3.

4.4 Broad Generalization

As shown in Table 2 and Figure 6, we examine TemplateRL’s broad generalization through out-of-domain evaluation and multimodal extensions.

Cross-Domain Generalization. To assess out-of-domain (OOD) performance, we evaluate TemplateRL on diverse reasoning tasks beyond mathematical domains: BALROG (agentic reasoning

Method	MathVision \uparrow	MathVerse \uparrow	MathVista \uparrow	MMMU \uparrow	BLINK \uparrow	Avg. \uparrow
CoT	27.7	27.3	61.1	51.2	45.1	42.5
GRPO	30.9	31.7	65.6	52.7	46.4	45.5
TemplateRL (Ours)	34.6	36.4	70.2	56.8	48.3	49.3
Δ (\uparrow)	+12.0%	+14.8%	+7.0%	+7.8%	+4.1%	+8.4%

Table 2: **Multimodal Extension.** We extend TemplateRL to multimodal domains with Qwen2.5-VL-3B-Instruct, and evaluate on multimodal reasoning benchmarks. TemplateRL consistently improves over GRPO across mathematical reasoning (MathVision, MathVerse, MathVista), general understanding (MMMU), and visual perception (BLINK).

in games), GPQA-D (graduate-level science), and MMLU-Pro (general knowledge). As shown in Figure 6, TemplateRL consistently outperforms GRPO across OOD tasks, with +6.1% performance gains on complex agentic scenarios (BALROG). This reveals that high-level template guidance effectively enhances model generalization to practical applications. Detailed results are in Appendix E.1.

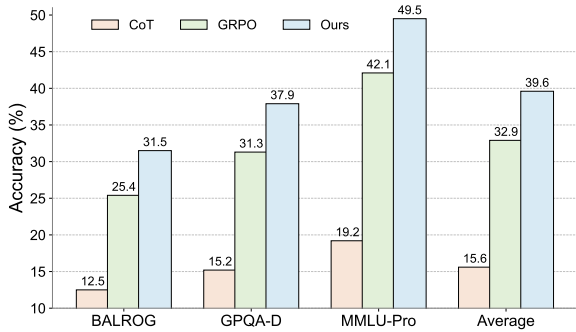


Figure 6: **Cross-domain generalization verification.** We provide OOD results on Qwen2.5-Math-7B-Base.

Multimodal Extension. Beyond textual reasoning, we extend TemplateRL to multimodal domains. We use the Geometry3K dataset (Lu et al., 2021) to construct MMTrain10K as training data and evaluate on diverse benchmarks, including mathematical reasoning (MathVision, MathVerse, MathVista), general understanding (MMMU), and visual perception (BLINK). Our settings are: batch size 128, 16 samples per question, and 2 template guidance. As shown in Table 2, TemplateRL consistently outperforms GRPO across all benchmarks, achieving +8.4% average improvement. This indicates that structured template guidance effectively generalizes beyond text modality to multimodal scenarios. More details are provided in Appendix E.3.

4.5 Dynamic Extensibility

We further explore the dynamic extensibility of the template library, enabling continuous knowledge updates during both training and inference.

Method	AIME24 \uparrow	AMC \uparrow	GPQA-D \uparrow	Avg. \uparrow
Fixed-5k	33.3	77.5	37.9	49.6
Dynamic-1k	16.7	52.5	20.2	29.8
Dynamic-2k	23.3	65.0	25.8	38.0
Dynamic-5k	36.7	80.0	38.9	51.9

Table 3: **Training-time dynamic extensibility.** We compare the fixed library baseline with dynamic expansion that continuously updates from successful rollouts.

Training-time Template Expansion. We explore dynamic library expansion during training by continuously incorporating new templates from successful rollouts. After training on 1k, 2k, 3k, and 5k samples, we extract high-level action sequences from correct samples and add them to the library. As shown in Table 3, dynamic expansion achieves substantial improvements across all benchmarks, with notable gains on AIME24 (+10.2%). This indicates that our explicit structured library enables continuous updates and reuse of successful exploration experiences, benefiting experience-intensive domains such as medical reasoning where iterative knowledge updates are necessary.

Test-time Template Expansion. Beyond training, we explore library expansion during inference. For each test sample, we retrieve five templates to generate multiple paths, apply majority voting for the answer, and immediately abstract patterns from voted outputs into the library before processing the next sample. Table 4 shows substantial improvements, demonstrating progressive library enrichment during inference using majority-voted pseudo-supervision. This continuous expansion capability helps for test-time scaling scenarios where models leverage accumulated knowledge from earlier predictions to improve subsequent ones.

4.6 Ablation Study and Analysis

Ablation Study. We investigate the effect of template guidance quantity ($|g| = 1, 2, 4$) on training dynamics with fixed total rollouts. Figure 8 shows

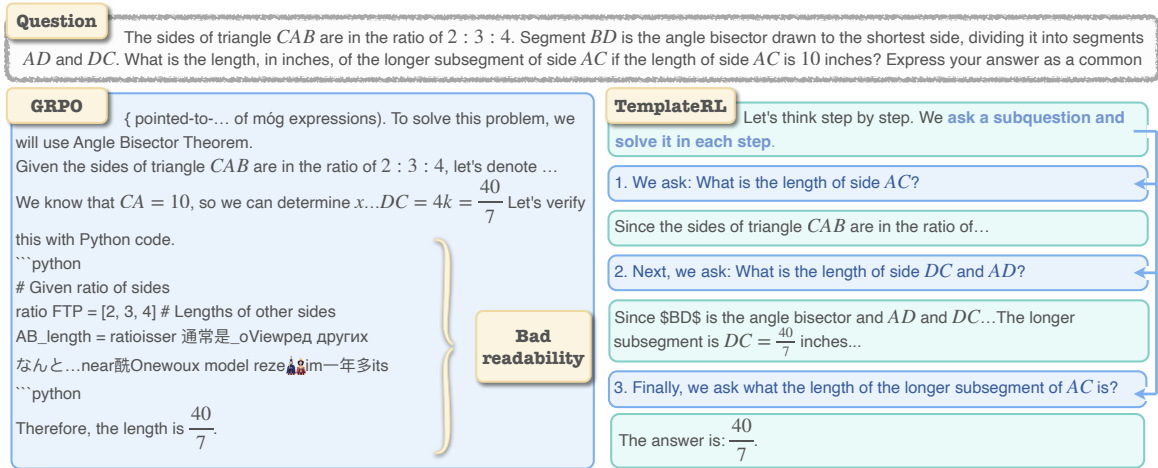


Figure 7: **Case Study.** TemplateRL produces a more structured and interpretable reasoning chain with clear steps.

Method	AIME24 \uparrow	AMC \uparrow	GPQA-D \uparrow	Avg. \uparrow
w/o update	33.3	77.5	37.9	49.6
w/ update	36.7	77.5	40.4	51.6
Δ (\uparrow)	+10.2%	+0%	+6.6%	+4.1%

Table 4: **Test-time dynamic extensibility.** We process test samples sequentially and expand the template library with majority-voting pseudo-supervision signals.

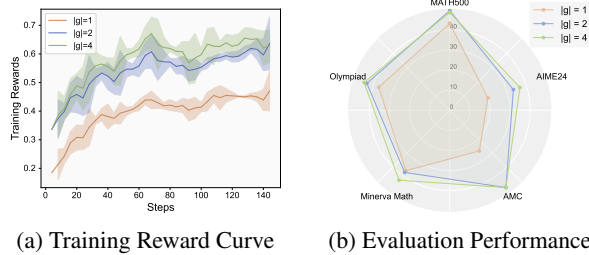


Figure 8: **Ablation study.** We provide results with different numbers of thought patterns (template guidance).

training rewards and evaluation results across varying templates. For clearer visualization, we uniformly adjust performance on AMC and MATH by subtracting a fixed value without affecting conclusions. We observe that more diverse template guidance often yields higher rewards. We choose $|g| = 2$ by default for flexible consideration.

Case Study. As shown in Figure 7 on a geometry problem, while GRPO produces less readable outputs, TemplateRL first identifies its strategy (e.g., divide-and-conquer), then addresses each subproblem with clear solutions. This demonstrates how template-guided RL training enhances system readability and interpretability.

Other Analysis. We provide additional results and analysis in Appendix, including comprehensive OOD results (E.1), discussion on template

construction strategy (E.6), ablation study (E.12), statistical analysis (E.13), case study (F).

5 Related Work

RL for LLMs. Reinforcement learning (RL) has become a popular paradigm for enhancing LLM reasoning (Guo et al., 2025; Team et al., 2025; Tang et al., 2025). Recent research mainly focuses on: (1) algorithmic refinements that address RL’s inherent limitations (Yu et al., 2025; Liu et al., 2025); (2) data-level optimizations (Zuo et al., 2025; Wang et al., 2025b); and (3) exploration enhancement (Wang et al., 2025a; Cui et al., 2025b). However, these methods remain constrained by unstructured self-sampling, lacking mechanisms to leverage structured guidance (Yue et al., 2025). TemplateRL addresses this by integrating explicit reasoning templates into policy optimization, steering training toward validated strategic patterns.

Reasoning with Template Guidance. Template-based guidance has been explored primarily at inference time, including retrieval-augmented generation (Asai et al., 2023; Zhao et al., 2024), task decomposition (Khot et al., 2022), and abstract thought patterns (Yang et al., 2024b; Wu et al., 2024). Few works systematically integrate structured templates into RL training. Our approach bridges this gap by constructing a template library and incorporating it during policy optimization. This enables effective learning of both concrete steps and underlying strategic logic, achieving superior performance, flexibility, and transferability.

6 Conclusion

We introduce TemplateRL, a template-guided RL framework for LLM reasoning. By constructing structured templates and integrating them during policy optimization, TemplateRL steers training toward validated strategic patterns. Experiments demonstrate significant improvements across models, datasets, and modalities. The framework exhibits cross-domain generalization, dynamic extensibility, and produces interpretable decision chains.

Limitations

In this paper, we primarily explore TemplateRL on text and multimodal reasoning tasks. In future work, we plan to extend the framework to more practical domains such as robotic control, scientific discovery, and embodied AI agents. Applying structured template guidance to real-world scenarios is expected to yield broader impact and demonstrate the generalizability of structured guidance across physical and interactive environments.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. U2436210, No. 62322120).

References

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. [Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms](#). *Preprint*, arXiv:2402.14740.
- Akari Asai, Sewon Min, Zexuan Zhong, and Danqi Chen. 2023. Retrieval-based language models and applications. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 6: Tutorial Abstracts)*, pages 41–46.
- Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. 2008. Monte-carlo tree search: A new framework for game ai. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 4, pages 216–217.
- Huiyao Chen, Yi Yang, Yinghui Li, Meishan Zhang, Baotian Hu, and Min Zhang. 2025. Beyond chunking: Discourse-aware hierarchical retrieval for long document question answering. *arXiv preprint arXiv:2506.06313*.
- Huiyao Chen, Yu Zhao, Zulong Chen, Mengjia Wang, Liangyue Li, Meishan Zhang, and Min Zhang. 2024. Retrieval-style in-context learning for few-shot hierarchical text classification. *Transactions of the Association for Computational Linguistics*, 12:1214–1231.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, and 1 others. 2025a. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*.
- Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, and 1 others. 2025b. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*.
- Sergio Da Silva. 2023. System 1 vs. system 2 thinking. *Psych*, 5(4):1057–1076.
- C Nicolò De Sabbata, Theodore R Summers, and Thomas L Griffiths. 2024. Rational metareasoning for large language models. *arXiv preprint arXiv:2410.05563*.
- Hao Dong, Zihan Ding, and Shanghang Zhang. 2020. *Deep Reinforcement Learning: Fundamentals, Research and Applications*, 1 edition, volume 1 of *eBook Packages: Mathematics and Statistics*. Springer Singapore.
- Susan E Embretson and Robert C Daniel. 2008. Understanding and quantifying cognitive complexity level in mathematical problem solving items. *Psychology Science*, 50(3):328.
- Mingkuan Feng, Jinyang Wu, Siyuan Liu, Shuai Zhang, Ruihan Jin, Feihu Che, Pengpeng Shao, Zhengqi Wen, and Jianhua Tao. 2025. Two-stage regularization-based structured pruning for llms. *arXiv preprint arXiv:2505.18232*.
- Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A. Smith, Wei-Chiu Ma, and Ranjay Krishna. 2025. Blink: Multimodal large language models can see but not perceive. In *Computer Vision – ECCV 2024*, pages 148–166. Cham. Springer Nature Switzerland.
- Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. 2025. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*.

- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Xinyu Guan, Li Lina Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. 2025. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shitong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. 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)*, pages 3828–3850.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
- Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. 2025a. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *Preprint*, arXiv:2503.24290.
- Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. 2025b. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.
- P. I. Jaffe, R. A. Poldrack, R. J. Schafer, and others. 2023. Modelling human behaviour in cognitive tasks with latent dynamical systems. *Nature Human Behaviour*, 7:986–1000.
- Daniel Kahneman. 2011. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, New York, NY.
- Zilin Kang, Chonghua Liao, Tingqiang Xu, and Huazhe Xu. 2025. Entropy regularizing activation: Boosting continuous control, large language models, and image classification with activation as entropy constraints. *arXiv preprint arXiv:2510.08549*.
- Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*.
- Levente Kocsis and Csaba Szepesvári. 2006a. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer.
- Levente Kocsis and Csaba Szepesvári. 2006b. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pages 282–293, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Samuel Kotz and Norman L. Johnson, editors. 1992. *Breakthroughs in Statistics: Methodology and Distribution*. Springer New York, New York, NY.
- Fong-Lok Lee and Rex Heyworth. 2000. Problem complexity: A measure of problem difficulty in algebra by using computer. *EDUCATION JOURNAL-HONG KONG-CHINESE UNIVERSITY OF HONG KONG-*, 28(1):85–108.
- Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.
- Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q. Jiang, Ziju Shen, and 1 others. 2024. NuminaMath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. <https://huggingface.co/datasets/NuminaMath>.
- Chonghua Liao, Ke Wang, Yuchuan Wu, Fei Huang, and Yongbin Li. 2025. Moa: Multi-objective alignment for role-playing agents. *arXiv preprint arXiv:2512.09756*.
- Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*.
- Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2023. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*.
- Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. 2021. Inter-GPS: Interpretable geometry problem solving with formal language and symbolic reasoning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language*

- Processing (Volume 1: Long Papers)*, pages 6774–6786, Online. Association for Computational Linguistics.
- Rui Miao, Yixin Liu, Yili Wang, Xu Shen, Yue Tan, Yiwei Dai, Shirui Pan, and Xin Wang. 2025. Blindguard: Safeguarding llm-based multi-agent systems under unknown attacks. *arXiv preprint arXiv:2508.08127*.
- Davide Paglieri, Bartłomiej Cupiał, and 1 others. 2025. **BALROG: Benchmarking agentic LLM and VLM reasoning on games**. In *The Thirteenth International Conference on Learning Representations*.
- Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2024. Mutual reasoning makes smaller llms stronger problem-solvers. *arXiv preprint arXiv:2408.06195*.
- Yiwei Qin, Xuefeng Li, Haoyang Zou, Yixiu Liu, Shijie Xia, Zhen Huang, Yixin Ye, Weizhe Yuan, Hector Liu, Yuanzhi Li, and 1 others. 2024. O1 replication journey: A strategic progress report—part 1. *arXiv preprint arXiv:2410.18982*.
- Qwen Team. 2024. **Qwen2.5: A party of foundation models**.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. **GPQA: A graduate-level google-proof q&a benchmark**. In *First Conference on Language Modeling*.
- Stuart Russell and Eric Wefald. 1991. Principles of metareasoning. *Artificial intelligence*, 49(1-3):361–395.
- Alejandro Salado and Roshanak Nilchiani. 2014. The concept of problem complexity. *Procedia Computer Science*, 28:539–546. 2014 Conference on Systems Engineering Research.
- John Schulman, Xi Chen, and Pieter Abbeel. 2017. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseek-math: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Xu Shen, Yixin Liu, Yiwei Dai, Yili Wang, Rui Miao, Yue Tan, Shirui Pan, and Xin Wang. 2025. Understanding the information propagation effects of communication topologies in llm-based multi-agent systems. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 12358–12372.
- Yuhao Shen, Tianyu Liu, Junyi Shen, Jinyang Wu, Quan Kong, Li Huan, and Cong Wang. 2026. Double: Breaking the acceleration limit via double retrieval speculative parallelism. *arXiv preprint arXiv:2601.05524*.
- Richard S Sutton, Andrew G Barto, and 1 others. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.
- Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, and 1 others. 2025. Agent kb: Leveraging cross-domain experience for agentic problem solving. *arXiv preprint arXiv:2507.06229*.
- Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. 2024. Mathsacle: Scaling instruction tuning for mathematical reasoning. *arXiv preprint arXiv:2403.02884*.
- Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.
- Ziyu Wan, Yunxiang LI, Xiaoyu Wen, Yan Song, Hanjing Wang, Linyi Yang, Mark Schmidt, Jun Wang, Weinan Zhang, Shuyue Hu, and Ying Wen. 2025. ReMA: Learning to meta-think for LLMs with multi-agent reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. 2024a. Measuring multimodal mathematical reasoning with MATH-vision dataset. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, and 1 others. 2025a. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.
- Yiping Wang, Qing Yang, Zhiyuan Zeng, and 1 others. 2025b. Reinforcement learning for reasoning in large language models with one training example. *arXiv preprint arXiv:2504.20571*.
- Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, and 1 others. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*.

- Jinyang Wu, Mingkuan Feng, Shuai Zhang, Feihu Che, Zengqi Wen, and Jianhua Tao. 2024. Beyond examples: High-level automated reasoning paradigm in in-context learning via mcts. *arXiv preprint arXiv:2411.18478*.
- Jinyang Wu, Shuo Yang, Changpeng Yang, Yuhao Shen, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. 2026a. Spark: Strategic policy-aware exploration via dynamic branching for long-horizon agentic learning. *arXiv preprint arXiv:2601.20209*.
- Jinyang Wu, Guocheng Zhai, Ruihan Jin, Jiahao Yuan, Yuhao Shen, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. 2026b. Atlas: Orchestrating heterogeneous models and tools for multi-domain complex reasoning. *arXiv preprint arXiv:2601.03872*.
- Jinyang Wu, Shuai Zhang, Feihu Che, Mingkuan Feng, Pengpeng Shao, and Jianhua Tao. 2025. Pandora’s box or aladdin’s lamp: A comprehensive analysis revealing the role of RAG noise in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5019–5039, Vienna, Austria. Association for Computational Linguistics.
- Fangzhi Xu, Hang Yan, Qiushi Sun, Jinyang Wu, Zixian Huang, Muye Huang, Jingyang Gong, Zichen Ding, Kanzhi Cheng, Yian Wang, and 1 others. 2026. Odyssearena: Benchmarking large language models for long-horizon, active and inductive interactions. *arXiv preprint arXiv:2602.05843*.
- An Yang, Beichen Zhang, Binyuan Hui, and 1 others. 2024a. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.
- Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and Bin Cui. 2024b. Buffer of thoughts: Thought-augmented reasoning with large language models. *Advances in Neural Information Processing Systems*, 37:113519–113544.
- Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. 2021. Mastering atari games with limited data. In *Advances in Neural Information Processing Systems*, volume 34, pages 25476–25488. Curran Associates, Inc.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.
- Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, and 1 others. 2024. Mmmu: A massive multi-discipline multi-modal understanding and reasoning benchmark for expert agi. In *Proceedings of CVPR*.
- Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*.
- Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. 2025. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*.
- Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Yu Qiao, and 1 others. 2025. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? In *European Conference on Computer Vision*, pages 169–186. Springer.
- Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. 2023. Evaluating the performance of large language models on gaokao benchmark. *arXiv preprint arXiv:2305.12474*.
- Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. 2024. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*.
- Yaowei Zheng, Juntong Lu, Shenzhi Wang, Zhangchi Feng, Dongdong Kuang, and Yuwen Xiong. 2025. Easyr1: An efficient, scalable, multi-modality rl training framework. <https://github.com/hiyouga/EasyR1>.
- Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2024. Language agent tree search unifies reasoning, acting, and planning in language models. In *Forty-first International Conference on Machine Learning*.
- Yuxin Zuo, Kaiyan Zhang, Shang Qu, Li Sheng, Xuekai Zhu, Biqing Qi, Youbang Sun, Ganqu Cui, Ning Ding, and Bowen Zhou. 2025. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*.

Appendix

In this section, we provide a comprehensive elaboration of the TemplateRL algorithm’s technical details. We describe the specific implementation of the Monte Carlo Tree Search algorithm, the construction process of our thought template library, and the adaptive retrieval and instantiation mechanisms for thought patterns during reasoning. Additionally, we provide supplementary experiments and case studies to illustrate our points further. The contents are organized as follows:

Contents

A TL;DR: Main Contributions and Takeaways	13
B Theoretical Analysis: Multi-Guidance RL and Template Transfer	14
B.1 Notation and Assumptions	14
B.2 Proof of Proposition 3.1	14
B.3 Proof of Proposition 3.2	15
B.4 Why template transfer increases success probability	15
C More Details about TemplateRL	16
C.1 Monte Carlo Tree Search (MCTS)	16
C.2 Thought Template Library Construction	17
C.3 Adaptive Template Retrieval	18
C.4 Discussion on Additional Method Details	20
D Experimental Details	20
E Supplementary Results	21
E.1 Out-of-Domain Results	21
E.2 Extension to More Models	21
E.3 Extension to Multimodal Domains	21
E.4 Extension to Other RL Algorithms	21
E.5 Analysis on Highly Capable Models	23
E.6 Discussion on Template Construction Strategy	23
E.7 Computational Overhead Analysis.	24
E.8 Ablation Study on the Action Space	25
E.9 Sensitivity to Training Data Size	25
E.10 Sensitivity to Pattern-Matching Metrics	25
E.11 Sensitivity to Seed Data Size.	26
E.12 Statistics on Reasoning Actions	26
E.13 Statistical Analysis	27

F Case Study	27
G Additional Discussion	27
G.1 Justification for Choosing MCTS.	27
G.2 Justification of Key Design Choices	28
G.3 Why A Small Set of Generic Patterns Can Generalize?	28
G.4 Are Templates Optimized for Test Sets?	28
G.5 How does performance scale with template set size?	29
G.6 Robustness to Seed Set and Cross-Domain Generalizability	29
G.7 Distinction between TemplateRL and Related Works	29
G.8 Future Work	29

A TL;DR: Main Contributions and Takeaways

Takeaways

- 1. Structure-Guided RL Paradigm:** Augmenting policy optimization with explicit structured templates significantly improves training efficiency, stability, and generalization over unstructured self-sampling.
- 2. Transferable High-Level Strategic Patterns:** Abstract reasoning templates exhibit remarkable cross-domain and cross-task transferability, enabling knowledge accumulation and flexible expert intervention through interpretable, editable structures.
- 3. Effective Learning on Weaker Models:** Template-guided training compensates for limited model capacity, achieving stable training where standard RL frequently collapses.

Existing RL methods for LLM reasoning rely primarily on unstructured self-sampling to fit scalar rewards, producing inefficient rollouts that fail to capture transferable problem-solving strategies. While recent work has explored algorithmic refinements, data-level optimizations, and exploration enhancement, no prior approach has systematically integrated structured guidance into policy optimization to address the fundamental limitations of trajectory

quality, strategy transferability, and interpretability. Our core contributions are:

- **Novel Structure-Guided RL Framework:** We introduce TemplateRL, which constructs an explicit reasoning template library via offline MCTS on minimal seed data, then seamlessly integrates these structured templates as guidance during online RL training. This design steers policy optimization toward validated strategic patterns.
- **Superior Performance and Stability:** TemplateRL outperforms GRPO by +99% on AIME, +41% on AMC, and +17% on Minerva Math on a 7B backbone, and consistently improves across scales (1.5B-8B) and architectures (Qwen, Llama). It maintains stable training on weaker models (e.g., Llama-3.2-3B) where GRPO typically collapses.
- **Broad Generalization:** Structured templates transfer effectively across diverse domains (BALROG for agents, GPQA-D for science, MMLU-Pro for knowledge) and modalities (text, vision), highlighting the universality of high-level thought patterns.
- **Interpretability and Dynamic Extensibility:** TemplateRL produces structured reasoning trajectories with identifiable strategic patterns, enabling error diagnosis and expert intervention. The template library is interpretable, directly editable, and supports continuous updates during both training and inference.

B Theoretical Analysis: Multi-Guidance RL and Template Transfer

B.1 Notation and Assumptions

We summarize the notation used in the propositions below.

- π_θ : parametric policy, π_{ref} : fixed reference policy.
- For a single training query \mathbf{q} , GRPO collects a total of G trajectories (trials). These G trajectories are partitioned into m groups (small groups), where $m = \lfloor G \rfloor$ and we assume for simplicity that G is divisible by m ; denote the per-group size by $g_s := G/m$.
- Within group $h \in \{1, \dots, m\}$, trajectories may be dependent (e.g., because they share the same guidance template or retrieved seed); different groups

are assumed approximately independent (e.g., they come from different guidance templates or independent retrievals).

- For group h , let $X_{h,j}$ be the indicator that the j -th trajectory in group h is positive (i.e., has $A > 0$). Define the group-success event $Y_h := \mathbf{1}\{\exists j \in [1, g_s] : X_{h,j} = 1\}$ (i.e., at least one positive in group h).
- Denote the per-group success probability by $p_h := \Pr(Y_h = 1)$. In a symmetric setting we write p_{grp} for the common per-group success probability.
- Other notation (seed database \mathcal{S} , retrieval Retr_k , seed-transfer success probability r , etc.) follows the earlier appendix.

B.2 Proof of Proposition 3.1

Proposition 3.1. (Stability and positive-sample guarantee) *Under the grouped sampling setup above, suppose different groups are independent (or weakly dependent so that group-level covariances are negligible). Then:*

1. *The probability that the entire batch of G trajectories contains at least one positive sample equals*

$$P_{\geq 1}^{\text{batch}} = 1 - \prod_{h=1}^m (1 - p_h).$$

In particular, if groups are symmetric ($p_h \equiv p_{\text{grp}}$), then

$$P_{\geq 1}^{\text{batch}} = 1 - (1 - p_{\text{grp}})^m,$$

which is strictly increasing in m and approaches 1 as $m \rightarrow \infty$ for any fixed $p_{\text{grp}} > 0$.

2. *Let g_h denote the (vector-valued) gradient contribution from group h (e.g., the average of per-trajectory contributions within the group). Assume group contributions are independent with $\mathbb{E}[g_h] = \bar{g}_{\text{grp}}$ and $\text{Var}[g_h] = \Sigma_{\text{grp}}$ (bounded). Then the variance of the batch-averaged estimator $\hat{g} = \frac{1}{m} \sum_{h=1}^m g_h$ satisfies*

$$\text{Var}[\hat{g}] \leq \frac{1}{m} \Sigma_{\text{grp}} + o\left(\frac{1}{m}\right),$$

i.e. the leading-order variance scales as $\mathcal{O}(1/m)$ (controlled by the number of independent groups), even if within-group samples are dependent.

Proof sketch. (1) For each group h define the Bernoulli variable Y_h indicating whether the group contains at least one positive trajectory. By definition $\Pr(Y_h = 1) = p_h$. If groups are independent, the probability that no group contains a positive sample equals $\prod_{h=1}^m \Pr(Y_h = 0) = \prod_{h=1}^m (1 - p_h)$, and the complement yields the stated formula. In the symmetric case set $p_h \equiv p_{\text{grp}}$.

Monotonicity in m holds because multiplying an additional factor $(1 - p_{\text{grp}})$ decreases the product, increasing its complement; the limit follows as $(1 - p_{\text{grp}})^m \rightarrow 0$ when $p_{\text{grp}} > 0$.

(2) Let g_h denote the aggregated gradient contribution from group h , e.g., $g_h = \frac{1}{g_s} \sum_{j=1}^{g_s} g_{h,j}$ where $g_{h,j}$ is the per-trajectory contribution. Even if the $g_{h,j}$ within a group are dependent, g_h is a single random vector (with finite second moments). Under independence across groups,

$$\begin{aligned} \text{Var}\left(\frac{1}{m} \sum_{h=1}^m g_h\right) &= \frac{1}{m^2} \sum_{h=1}^m \text{Var}[g_h] \\ &= \frac{1}{m} \left(\frac{1}{m} \sum_{h=1}^m \text{Var}[g_h] \right) \\ &\preceq \frac{1}{m} \Sigma_{\text{grp}}, \end{aligned}$$

where Σ_{grp} is a uniform bound on $\text{Var}[g_h]$. If group covariances are weak (mixing), their contribution is lower-order, giving the $o(1/m)$ term. Thus the dominant variance reduction is governed by the number of independent groups m , not the raw total sample size G when within-group dependence is present. Intuitively: dependent samples within a group provide limited additional independent information, so averaging across independent groups is the effective variance reducer. \square

Remark. The theorem clarifies that when data are collected in dependent clusters (groups) the effective sample size is the number of independent clusters m . Therefore, to achieve variance reduction it is valuable to increase the number of independent groups (e.g., by using multiple distinct guidance templates or distinct retrievals) rather than only increasing correlated samples within a single group.

B.3 Proof of Proposition 3.2

Proposition 3.2. (Template transfer improves success probability under grouped sampling) *Consider the grouped sampling with m groups, and suppose group h uses PCC retrieval $\text{Retr}_k^{(h)}(\mathbf{q})$ returning k templates whose best trajectories are used as*

candidate rollouts within that group. Let $T(\mathbf{q})$ be a template produced by MCTS for a seed query \mathbf{q} , the per-mini-group template-transfer success probability is:

$$r_h = \Pr\left(\exists T \in \text{Retr}_k^{(h)}(\mathbf{q}) \text{ s.t. } A(T(\mathbf{q}), \mathbf{q}) > 0\right).$$

Assume different groups are independent. Then:

1. *Per-mini-group: the event that mini-group h contains at least one positive trajectory due to template transfer occurs with probability at least r_h . If $r_h > p_h^{\text{policy}}$ (the per-group success probability under policy rollouts), template transfer strictly improves per-group success.*
2. *Batch-level: the probability that the full batch has at least one positive sample is*

$$P_{\geq 1}^{\text{template}} = 1 - \prod_{h=1}^m (1 - r_h),$$

which exceeds the policy-rollout probability $1 - \prod_{h=1}^m (1 - p_h^{\text{policy}})$ whenever $r_h \geq p_h^{\text{policy}}$ for all h and strict for some h .

Proof sketch. (1) and (2) follow from the same complement argument as in the proof of Proposition 3.1 (B.2). \square

B.4 Why template transfer increases success probability

The key reason why template transfer improves the chance of positive samples is that MCTS produces high-quality trajectories that approximately maximize expected reward for their original problem. If a new query \mathbf{q} is similar to a seed query \mathbf{q}' , then the action sequence that solves \mathbf{q}' remains near-optimal for \mathbf{q} under mild regularity assumptions.

Lemma B.1 (Transfer success under smoothness). *Let $T^*(\mathbf{q}')$ be a template produced by MCTS for a seed query \mathbf{q}' such that $A(T^*(\mathbf{q}'); \mathbf{q}') > 0$. Suppose the advantage function $A(T; \mathbf{q})$ is L -Lipschitz continuous in \mathbf{q} for all trajectories τ :*

$$|A(T; \mathbf{q}) - A(T; \mathbf{q}')| \leq L d(\mathbf{q}, \mathbf{q}')$$

for some distance metric d . Then, if

$$A(T^*(\mathbf{q}'); \mathbf{q}') > L d(\mathbf{q}, \mathbf{q}'),$$

it follows that $A(T^(\mathbf{q}'); \mathbf{q}) > 0$.*

Proof sketch. By Lipschitz continuity:

$$A(T^*(\mathbf{q}'); \mathbf{q}) \geq A(T^*(\mathbf{q}'); \mathbf{q}') - L d(\mathbf{q}, \mathbf{q}').$$

If the right-hand side is positive, the inequality implies the transferred trajectory still has positive advantage on \mathbf{q} . \square

Remark (Template transfer as a high-probability event). MCTS templates $T^*(\mathbf{q}')$ are approximately optimal for \mathbf{q}' and often satisfy $A(T^*(\mathbf{q}'); \mathbf{q}') \gg 0$. Thus, for new queries \mathbf{q} that are close to \mathbf{q}' (small $d_{\text{PCC}}(\mathbf{q}, \mathbf{q}')$), the inequality in Lemma B.1 is likely to hold. This implies that retrieving high-quality templates from similar seed problems leads to transferred trajectories with positive advantage on the new problem. Consequently, the per-group template success probability r_h is strictly higher than the baseline policy success probability p_h^{policy} .

C More Details about TemplateRL

C.1 Monte Carlo Tree Search (MCTS)

As a heuristic search algorithm, MCTS has demonstrated remarkable success in complex reasoning and decision-making environments (Chaslot et al., 2008; Ye et al., 2021; Zhou et al., 2024). The algorithm conceptualizes search spaces as tree structures and has achieved significant breakthroughs across various domains, most notably in game-playing AI such as AlphaGo and AlphaZero (Dong et al., 2020). As described in Section 2.3 in the main text, we employ MCTS to generate solution trees based on a small set of 500 seed samples.

To implement MCTS effectively, we first define a predefined action set. Understanding human complex reasoning is crucial for modeling cognitive processes (Jaffe et al., 2023). Existing studies distinguish between two cognitive systems: System 1 and System 2 (Kahneman, 2011; Da Silva, 2023). While “System 1” represents fast, intuitive, yet error-prone thinking, “System 2” involves slow, deliberative thinking with superior performance. With the emergence of advanced models like OpenAI’s o1, developing efficient “System 2” approaches to emulate human cognitive processes has gained significant research attention (Qin et al., 2024; Guo et al., 2025).

Inspired by this and following previous work (Wu et al., 2024; Qi et al., 2024), we introduce five human-like reasoning actions to bridge the gap between model reasoning and human cognition:

- **Divide and Conquer (DC, a_1)**: Approaching complex problems by breaking them into manageable sub-problems for easier resolution.
- **Self-Reflection (SR, a_2)**: Assessing and refining prior solutions during the reasoning process to ensure correctness.
- **System Analysis (SA, a_3)**: Analyzing the overall structure of the problem and identifying the constraints and conditions before addressing it, thereby clarifying task requirements effectively.
- **One-Step Thought (OST, a_4)**: Aiming to address a single aspect of the problem through a focused and concise reasoning step.
- **Chain-of-Thought (CoT, a_5)**: Adopting a sequential reasoning process that builds a series of connected logical steps.

Based on the above predefined action set $\mathcal{A} = \{a_1, \dots, a_{|\mathcal{A}|}\}$ and model π_θ , for each question $s_i \in \mathcal{S}$, MCTS builds a search tree \mathcal{T}_i where: the root node represents question s_i , each edge denotes an action $a \in \mathcal{A}$, and each child node n contains partial solutions generated by π_θ under the corresponding action. A path from root s_i to leaf node $\mathbf{n}_{i,j,d_{i,j}}$ forms a solution trajectory:

$$\mathbf{t}_{i,j} = (s_i, a_{i,j,1}, \mathbf{n}_{i,j,1}, \dots, a_{i,j,d_{i,j}}, \mathbf{n}_{i,j,d_{i,j}}),$$

Each intermediate node $\mathbf{n}_{i,j,\ell}$ is generated based on the cumulative context of its parent nodes and current action:

$$\mathbf{n}_{i,j,\ell} = \pi_\theta([s_i, a_{i,j,1}, \mathbf{n}_{i,j,1}, \dots, a_{i,j,\ell}]).$$

Specifically, the MCTS algorithm involves an iterative search process with four key steps: selection, expansion, simulation, and backpropagation.

(1) *Selection.* Starting from the root node, we identify optimal nodes for expansion by traversing the tree level-by-level until reaching a leaf node (maximum depth or final answer). To balance exploration and exploitation, we employ Upper Confidence Bounds applied to Trees (UCT) (Kocsis and Szepesvári, 2006b):

$$\text{UCT}(\mathbf{n}) = Q(\mathbf{n}) + w \sqrt{\frac{\ln N(\mathbf{n}_p)}{N(\mathbf{n})}},$$

where $Q(\mathbf{n})$ is the reward value, $N(\mathbf{n})$ is the visit count, \mathbf{n}_p is the parent node, and w is the exploration weight. The node with the highest UCT value is selected.

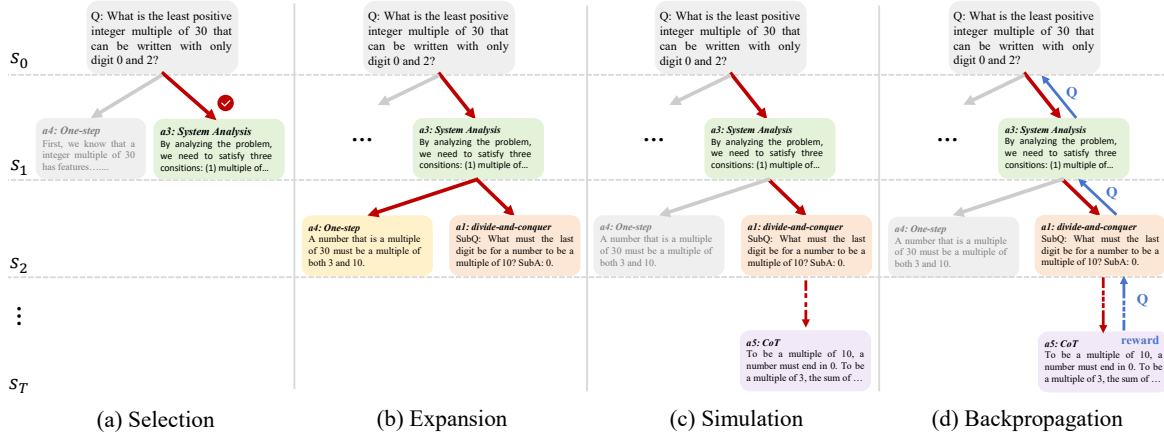


Figure 9: An illustration of four phases in an iteration of MCTS for complex reasoning tasks.

(2) *Expansion*. The selected node \mathbf{n} is expanded by sampling n actions from π_θ and generating corresponding reasoning outcomes. These n child nodes are then added to the tree.

(3) *Simulation*. Starting from the selected node, we iteratively sample and expand nodes until reaching a terminal state. To enhance efficiency, we implement an early termination strategy based on self-consistency (Wang et al., 2023): if the model’s consistency score exceeds threshold c (i.e., $\text{SC}(\mathbf{n}) > c$), the simulation terminates early.

(4) *Backpropagation*. Upon simulation completion, node information is updated along the simulation path. Visit counts are incremented ($N(\mathbf{n}) \leftarrow N(\mathbf{n}) + 1$), and node value $Q(\mathbf{n})$ is propagated backward to its parent node \mathbf{n}_p :

$$Q(\mathbf{n}_p) \leftarrow (1 - \alpha)Q(\mathbf{n}_p) + \alpha Q(\mathbf{n}),$$

where α is a discount factor. For terminal nodes, following Qi et al. (2024), we use the confidence of self-consistency majority voting as the reward value.

Through this process, we obtain diverse solution traces $\mathbb{T} = \{\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_t\}$ for each question $\mathbf{s}_i \in \mathcal{S}$. MCTS assigns a final reward $R(\mathbf{t}_j|\mathbf{s}_i)$ to each trace $\mathbf{t}_j \in \mathbb{T}$.

Figure 9 illustrates the four phases in an iteration, expanding the tree and then updating reward values.

C.2 Thought Template Library Construction

After MCTS exploration, we obtain multiple solution paths for each question $\mathbf{s}_i \in \mathcal{S}$. We then construct a thought template library, which can be used in RL optimization and update during both training and testing process.

Template Construction Process. The construction involves five key steps:

1. **Seed Set Selection:** We select a small set of seed problems $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_s\}$ (500 samples in our experiments). These problems are chosen to be complex, while also providing diverse coverage of possible solution strategies.
2. **MCTS Exploration:** For each seed problem $\mathbf{s}_i \in \mathcal{S}$, we apply MCTS to explore the solution space. MCTS recursively simulates different action sequences and evaluates resulting solution paths, generating a solution tree \mathcal{T}_i where each path represents a candidate reasoning trajectory.
3. **Optimal Path Selection:** From the multiple solution paths generated by MCTS, we select the best trajectory for each question using a balanced scoring metric:

$$\text{Score}(\mathbf{s}_i, \mathbf{t}_j) = b \cdot R(\mathbf{t}_j|\mathbf{s}_i) - (1 - b) \cdot C(\mathbf{t}_j),$$

where $R(\mathbf{t}_j|\mathbf{s}_i)$ is the reward, $C(\mathbf{t}_j)$ is trajectory complexity (action count), and $b = 0.95$ balances solution quality against conciseness. The optimal trace, $\mathbf{t}_{i,\text{best}}$, is selected as the one that maximizes this score:

$$\mathbf{t}_{i,\text{best}} = \arg \max_{\mathbf{t}_j} \text{Score}(\mathbf{s}_i, \mathbf{t}_j).$$

4. **Template Abstraction:** Since each node in $\mathbf{t}_{i,\text{best}}$ corresponds to an action $a_\ell \in \mathcal{A}$, we extract the action sequence as a high-level template $T_i = (a_1, \dots, a_d)$. For instance, a

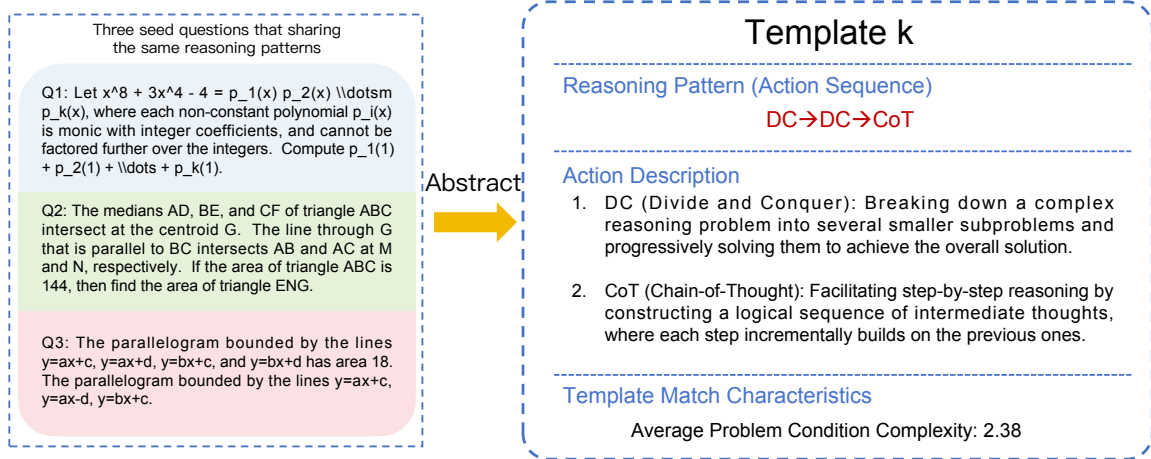


Figure 10: Thought Template Visualization. On the left, the seed dataset contains 3 questions with the same reasoning pattern, “DC→DC→CoT”, corresponding to template k on the right. The template includes both the reasoning pattern and characteristics used for matching during testing, based on cognitive complexity metrics. When a test question q matches template k, reasoning follows the corresponding pattern “DC→DC→CoT”.

successful solution might follow the pattern $a_1 \rightarrow a_2 \rightarrow a_4$ (System Analysis \rightarrow One-Step Thought \rightarrow Divide and Conquer). **Thus, a template is defined as an abstract, high-level sequence of actions that represents a strategic problem-solving approach to a particular task.**

- Template Aggregation:** To organize the extracted patterns, we use Problem Condition Complexity (PCC) (Lee and Heyworth, 2000; Embretson and Daniel, 2008) as a categorization metric. PCC quantifies the structural complexity of a problem based on the number of prior conditions and constraints, which can be computed by prompting model π_θ with the question. Templates sharing similar PCC values are grouped together, as problems with comparable structural complexity tend to benefit from similar reasoning strategies. Through this aggregation, each template \hat{T}_j stores both its action pattern and the average PCC of questions sharing this pattern:

$$\hat{T}_j = (\text{PCC}_{T_j}, T_j),$$

where $\text{PCC}_{T_j} = \frac{1}{|I_j|} \sum_{i \in I_j} \text{PCC}(s_i)$ and $I_j = \{i : T_i = T_j\}$.

Final Template Library. The final template library $\mathcal{L} = \{\hat{T}_1, \dots, \hat{T}_m\}$ consists of 103 unique templates derived from 500 seed samples. Each template represents a generalized problem-solving strategy characterized by both its structural pattern

(action sequence) and typical problem complexity (average PCC). These templates serve as structured guidance during RL training, enabling the policy to leverage proven strategic patterns when encountering similar problems. The final library thus provides a powerful tool for both training and dynamic adaptation during inference.

For better visualization, we provide a detailed example template in Figure 10.

C.3 Adaptive Template Retrieval

During RL training, we employ an adaptive retrieval mechanism to identify relevant reasoning strategies from our template library. This approach follows meta-reasoning principles (Russell and Wefald, 1991; De Sabbata et al., 2024), which emphasize selecting appropriate strategies based on problem characteristics.

Template Retrieval. For each training question q_t , we first compute its PCC metric, which characterizes the question’s structure and complexity. We then calculate the distance to each template $\hat{T}_j \in \mathcal{L}$:

$$d_j = |\text{PCC}_{q_t} - \text{PCC}_{T_j}|.$$

This distance quantifies similarity between the current question and problems from which each template was derived. Note that, the PCC prompt is:

You are an AI assistant to help me rephrase questions by splitting the question context into conditions. In your rephrased question, remember to fully express the information in the original question. That means, you should count the number of

Problem: "Find the 3D distance between (2,1,-4) and (5,8,-3)."

➤ Step 1: PCC Computation

Given conditions: (1) two 3D points and (2) Distance calculation typically requires the Euclidean distance
 → PCC(query) = 2

➤ Step 2: PCC Computation

- T_1 : PCC=1.8, actions: $a_1 \rightarrow a_4$ (decompose → one-step reasoning)
- T_2 : PCC=2.1, actions: $a_1 \rightarrow a_4 \rightarrow a_2$ (decompose → one-step reasoning → Verify)
- T_3 : PCC=3.5, actions: $a_3 \rightarrow a_4 \rightarrow a_2$ (complex multi-step)

Distances: $d(\text{query}, T_1)=0.2$, $d(\text{query}, T_2)=0.1$, $d(\text{query}, T_3)=1.5$

Selected: T_1 and T_2 (top-2)

➤ Step 3: Rollout Generation

- **Using $T_1(a_1 \rightarrow a_4)$**

[Action a_1 : divide-and-conquer]

"Decompose into coordinate differences:

$$\Delta x = 5-2 = 3, \Delta y = 8-1 = 7, \Delta z = -3-(-4) = 1"$$

[Action a_4 : one-step reasoning]

$$\text{"Distance} = \sqrt{3^2 + 7^2 + 1^2} = \sqrt{59}"$$

- **Using $T_2(a_1 \rightarrow a_4 \rightarrow a_2)$**

[Action a_1 : divide-and-conquer]

"Decompose into coordinate differences:

$$\Delta x = 5-2 = 3, \Delta y = 8-1 = 7, \Delta z = -3-(-4) = 1"$$

[Action a_4 : one-step reasoning]

$$\text{"Distance} = \sqrt{3^2 + 7^2 + 1^2} = \sqrt{59}"$$

Figure 11: Concrete example of adaptive template retrieval. Given a 3D distance calculation problem with PCC = 2, the system retrieves top-2 templates (T_1 and T_2) based on complexity similarity while filtering out the overly complex T_3 ($d = 1.5$). The retrieved templates guide distinct reasoning trajectories: T_1 provides a streamlined solution path, while T_2 incorporates an additional verification step for enhanced reliability.

distinct prior conditions or given facts that must be satisfied or considered to solve it.

Problem: <problem>

Number of prior conditions:

Template Selection and Application. We rank templates by distance and select the top- k most similar: $\{\hat{T}_{i_1}, \dots, \hat{T}_{i_k}\}$. These selected templates contain high-level patterns (a_1, \dots, a_d) proven effective for problems with similar complexity profiles. During rollout generation, retrieved templates serve as structured guidance that balances exploiting known successful strategies with model-internal exploration. This adaptive mechanism ensures the model leverages appropriate reasoning strategies based on problem characteristics, enabling more targeted and effective sampling across diverse prob-

lem types.

Concrete Example. We provide a detailed processing example in Figure 11. This example demonstrates how PCC-based retrieval identifies templates with appropriate complexity levels (rejecting the overly complex T_3), and how different templates induce varied reasoning structures. Template T_1 provides a concise solution path, while T_2 incorporates an additional verification step that enhances answer reliability—both are valid strategies, but T_2 may be preferred in high-stakes scenarios where correctness verification is critical. During RL training, the model learns to leverage these structural variations to maximize expected rewards across diverse problem instances.

System Prompt: A conversation between User and Assistant. The user asks a question, and the Assistant solves it step by step. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer, i.e., reasoning process here ... the answer is: `\boxed{answer}`.

Figure 12: The system prompt used for all experiments.

Template Updates. Regarding template updates, during RL training, we optionally leverage correct rollouts to continually enrich the template library. For each correct rollout, we apply keyword-based or lightweight model-based pattern extraction to automatically identify the underlying action sequence (template), which is then incorporated into the library for continuous expansion. Tables 3-4 empirically validate the effectiveness of dynamic updates during both training and inference.

C.4 Discussion on Additional Method Details

This section further clarifies several key details regarding template construction and abstraction.

Template Aggregation via Exact Action Sequence Matching. Templates are aggregated through exact matching. Two solution traces share the same template if and only if they follow identical action sequences. For example, all traces following “ $a_1 \rightarrow a_4 \rightarrow a_3$ ” are merged into a single template. The aggregation process treats each unique sequence T as a key, collects all questions matching T with their PCC values, and computes the average PCC to obtain the template’s characteristic complexity score, storing the (T, PCC) pair. This exact matching ensures templates represent genuinely identical reasoning structures.

Predefined Actions vs. Automatically Discovered Patterns. Our framework employs a two-level design: The action set \mathcal{A} (e.g., divide-and-conquer) comprises predefined atomic reasoning primitives. However, *templates* (sequences of these actions) are automatically discovered through MCTS exploration. For instance, while “ a_1 ” (divide-and-conquer) is predefined, the template “ $a_1 \rightarrow a_4 \rightarrow a_3$ ” emerges from successful solution paths. This design provides interpretability through recognizable reasoning concepts while enabling discovery of effective strategy combinations that may not be obvious to human experts.

Robustness to Paraphrasing through Action-Level Abstraction. Template matching operates

at the *action level* rather than natural language level. During MCTS, each reasoning step is generated by a specific action prompt and assigned the corresponding action label. Different surface expressions of the same reasoning step (e.g., “break into x-y coordinates” vs. “decompose dimension-wise” vs. “handle each axis separately”) are unified through their shared action label a_1 . When abstracting traces into templates, we extract action sequences rather than literal text, making templates inherently robust to paraphrasing. This contrasts with approaches that directly cluster natural language trajectories, which often require complex similarity metrics to handle linguistic variations.

D Experimental Details

In addition to the implementation details in the main paper, we provide supplementary details here:

Baseline Methods. For fair comparison, we benchmark TemplateRL with the following baselines on Qwen2.5-Math-7B: (1) *GRPO* (Shao et al., 2024), using the same 5.5k training samples; (2) *SimpleRL-Zero* (Zeng et al., 2025), applying GRPO to 24k samples from GSM8K and MATH; (3) *OpenReasoner-Zero* (Hu et al., 2025b), employing PPO with 129k samples; (4) *PRIME-Zero* (Cui et al., 2025a), utilizing implicit process rewards on 150k NuminaMath (Li et al., 2024) queries; (5) *Oat-Zero* (Liu et al., 2025), introducing Dr.GRPO to mitigate length bias on 8k MATH questions.

Implementation Details. During training, we generate with rollout parameters of temperature=0.8 and top-p=0.95, and a maximum generation of 1500 tokens. The reward function is a binary accuracy metric verified by Math-Verify, defined as $r(\mathbf{o}) = \mathbb{1}\{\mathbf{o} \text{ contains the correct final answer}\}$. Moreover, we employ a cosine learning rate decay with warm-up. The maximum learning rate is set at 3×10^{-6} , and the warm-up ratio is set at 0.1. We use the same system prompt in Figure 12 for all RL experiments. All experiments were conducted on a machine running Ubuntu 22.04, equipped with

Model	GPQA-D \uparrow	MMLU-Pro \uparrow	BALROG \uparrow
Qwen2.5-Math-7B-Base (Yang et al., 2024a)	15.2	19.2	12.5
Qwen2.5-Math-7B-Instruct (Yang et al., 2024a)	28.8	36.1	15.4
SimpleRL-Zero (Zeng et al., 2025)	21.2	36.9	17.4
OpenReasoner-Zero (Hu et al., 2025a)	30.3	54.6	28.3
PRIME-Zero (Cui et al., 2025a)	20.2	34.3	24.1
Oat-Zero (Liu et al., 2025)	25.8	41.8	26.2
GRPO (Shao et al., 2024)	31.3	42.1	25.4
TemplateRL (Ours)	37.9	49.5	31.5
Δ (\uparrow)	+21.1%	+17.6%	+24.0%

Table 5: Detailed results on three representative out-of-distribution benchmarks (Qwen2.5-Math-7B-Base).

NVIDIA A100-80GB GPUs.

E Supplementary Results

E.1 Out-of-Domain Results

We present supplementary out-of-distribution generalization results in Table 5.

E.2 Extension to More Models

In addition to the results presented in the main text based on Qwen2.5-Math-1.5B-Base, Qwen2.5-7B-Instruct, and Llama3.1-8B-Instruct, we further evaluate TemplateRL on two additional models: Qwen2.5-Math-1.5B-Instruct and Llama3.2-3B-Instruct. As illustrated in Table 6, TemplateRL achieves an average performance improvement of 3.2% on Qwen2.5-Math-1.5B-Instruct and 5.6% on Llama3.2-3B-Instruct.

E.3 Extension to Multimodal Domains

As shown in Table 2 in the main text, we extend TemplateRL to multimodal domains. We provide more details and analysis here.

Dataset Construction. We construct the MM-Train10K dataset based on Geometry3K (Lu et al., 2021), which contains geometry problems with textual descriptions and visual diagrams. To enhance data diversity and problem complexity distribution, we employ Gemini-2.5-Pro to augment the original dataset by systematically adding or removing auxiliary details from problem statements. This process adjusts problem difficulty while preserving geometric relationships and visual diagrams, ultimately expanding the dataset to 10K samples with varied complexity levels (MMTrain10K).

Implementation Setup. We utilize Qwen2.5-VL-3B-Instruct as the base model and follow the EasyR1 pipeline (Zheng et al., 2025) for RL training. Our training configuration maintains consistency with the text-only setting: batch size 128, 16 samples per question, $|g| = 2$ structured guidance templates, and maximum generation length of 4096 tokens to accommodate detailed multimodal reasoning processes.

Evaluation Benchmarks. We evaluate across five diverse benchmarks: (1) **Mathematical reasoning:** MathVision (Wang et al., 2024a), MathVerse (Zhang et al., 2025), and MathVista (Lu et al., 2023); (2) **General understanding:** MMMU (Yue et al., 2024); (3) **Visual perception:** BLINK (Fu et al., 2025).

Results and Analysis. As shown in Table 2, TemplateRL consistently outperforms both CoT and GRPO across all benchmarks, achieving +8.4% average improvement over GRPO. Notably, TemplateRL excels on mathematical reasoning tasks: MathVision (+12.0% vs. GRPO) and MathVerse (+14.8% vs. GRPO), demonstrating that structured template guidance effectively transfers to multimodal mathematical reasoning. The consistent improvements across diverse tasks (from math to visual perception), validate TemplateRL’s core principle: integrating structured reasoning guidance with policy optimization generalizes beyond text modality to multimodal scenarios.

E.4 Extension to Other RL Algorithms

To demonstrate the generalizability of our thought-augmented framework beyond GRPO, we investigate its compatibility with alternative policy

Method	AIME24 \uparrow	AMC \uparrow	MATH500 \uparrow	GSM8K \uparrow	Minerva \uparrow	Olympiad \uparrow	CollegeMath \uparrow	Gaokao23 \uparrow	Avg. \uparrow
Qwen2.5-Math-1.5B-Base (Yang et al., 2024a)									
CoT	10.0	42.5	59.0	74.6	24.3	27.6	39.5	49.6	40.9
GRPO	13.3	40.0	66.4	74.7	25.0	30.1	40.5	52.7	42.8
Ours	16.7	55.0	69.0	84.2	31.6	33.6	47.3	54.8	49.0
Δ (\uparrow)	+25.6%	+37.5%	+3.9%	+12.7%	+26.4%	+11.7%	+16.8%	+4.0%	+14.5%
Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024a)									
CoT	6.7	47.5	68.2	76.8	28.3	36.9	47.1	63.1	46.8
GRPO	13.3	52.5	76.8	85.9	28.3	36.7	45.9	65.2	50.6
Ours	16.7	55.0	76.0	86.5	29.4	39.7	48.3	65.7	52.2
Δ (\uparrow)	+25.6%	+4.8%	-1.0%	+1.0%	+3.9%	+8.2%	+5.3%	+1.0%	+3.2%
Qwen2.5-7B-Instruct (Qwen Team, 2024)									
CoT	13.3	47.5	73.2	90.0	30.5	38.8	46.9	64.2	50.5
GRPO	13.3	57.5	76.6	90.1	32.4	36.1	44.5	62.9	51.6
Ours	16.7	67.5	78.0	91.5	36.8	40.6	50.6	65.2	55.8
Δ (\uparrow)	+25.6%	+17.4%	+1.9%	+1.6%	+13.6%	+12.5%	+13.7%	+3.7%	+8.1%
Llama-3.1-8B-Instruct (Grattafiori et al., 2024)									
CoT	3.3	20.0	36.6	77.2	16.2	15.9	13.3	29.9	26.5
GRPO	3.3	22.5	45.0	82.9	21.0	16.1	31.7	40.8	32.9
Ours	6.7	30.0	52.2	85.2	26.8	17.3	34.1	42.6	36.7
Δ (\uparrow)	+103.0%	+33.3%	+16.0%	+2.8%	+27.7%	+7.5%	+7.6%	+4.5%	+11.6%
Llama-3.2-3B-Instruct (Grattafiori et al., 2024)									
CoT	6.7	20.0	38.3	69.3	11.8	12.6	23.8	33.5	27.6
GRPO	3.3	25.0	47.8	75.2	17.6	14.5	34.1	40.8	32.2
Ours	6.7	27.5	48.8	78.8	18.4	16.0	32.5	43.1	34.0
Δ (\uparrow)	+103.0%	+10.0%	+2.1%	+4.8%	+4.6%	+10.4%	-4.7%	+5.7%	+5.6%

Table 6: Accuracy (%) results of different LLMs across eight benchmarks. The best results in each box are highlighted in **bold**. We provide the relative improvement of our method compared to GRPO.

Method	AIME24 \uparrow	AMC \uparrow	MATH500 \uparrow	GSM8K \uparrow	Minerva \uparrow	Olympiad \uparrow	CollegeMath \uparrow	Gaokao23 \uparrow	Avg. \uparrow
Qwen2.5-Math-7B-Base (Yang et al., 2024a)									
RLOO	30.0	50.0	73.8	82.7	35.5	36.0	39.8	64.2	51.5
TemplateRL	33.3	67.5	78.8	85.8	36.7	41.2	41.8	68.1	56.7
Δ (\uparrow)	+11.0%	+35.0%	+6.8%	+3.7%	+3.4%	+14.4%	+5.0%	+6.1%	+10.1%
Qwen2.5-Math-1.5B-Base (Yang et al., 2024a)									
RLOO	20.0	50.0	68.0	82.6	28.7	32.0	41.4	51.2	46.7
TemplateRL	26.7	57.5	75.6	84.7	33.8	33.8	41.8	52.9	50.9
Δ (\uparrow)	+33.5%	+15.0%	+11.2%	+2.5%	+17.8%	+5.6%	+1.0%	+3.3%	+9.0%

Table 7: Extension to other RL algorithms. Results demonstrate consistent improvements when integrating thought-augmented training with alternative RL algorithms, highlighting the generalizability of our approach.

optimization algorithms. Specifically, we extend TemplateRL to REINFORCE Leave-One-Out (RLOO) (Ahmadian et al., 2024), a variance-reduced policy gradient method that has shown effectiveness in model fine-tuning. Following our default configuration, we set the number of guidance patterns to 2 (i.e., $|g| = 2$) and maintain consistent experimental settings across both model scales.

As shown in Table 7, TemplateRL consistently enhances RLOO performance across all benchmarks and model configurations. For Qwen2.5-Math-7B-Base, RLOO+TemplateRL achieves substantial improvements over vanilla RLOO, with no-

table gains on AMC (+17.5 points, +35.0% relative improvement) and AIME24 (+3.3 points, +11.0% relative improvement). Similar trends emerge for the smaller Qwen2.5-Math-1.5B-Base model, where RLOO+TemplateRL delivers consistent improvements across all metrics, achieving an average performance boost of +4.2 points (+9.0% relative improvement). These results demonstrate that the benefits of thought-augmented training extend beyond GRPO to other RL algorithms, highlighting the framework’s versatility and broad applicability.

Method	AIME25 \uparrow	GPQA-Diamond \uparrow	Avg. \uparrow
<i>Part I: Test-Time Exploration Analysis (w/o RL training)</i>			
CoT (pass@16)	80.0	76.8	78.4
Template-Guided Inference (pass@16)	83.3	82.8	83.1
Δ (%)	+3.3	+6.0	+4.7
<i>Part II: Post-Training Exploration Analysis (after RL)</i>			
GRPO (trained, pass@16)	83.3	80.8	82.0
TemplateRL (trained, pass@16)	85.7	84.3	85.0
Δ vs GRPO (%)	+2.4	+4.5	+3.0

Table 8: Exploration Analysis (Pass@16) on highly capable models (Qwen3-32B). Part I examines exploration quality via test-time inference, which mirrors RL rollout sampling. Part II evaluates post-RL performance.

E.5 Analysis on Highly Capable Models

A natural concern is whether template guidance might restrict the exploration space of larger, more capable models, potentially hindering their inherent reasoning abilities. To address this, we conduct a two-stage analysis on Qwen3-32B with pass@16 evaluation: (1) **Exploration Quality Analysis**: We compare standard CoT with template-guided test-time inference to assess the quality of exploration spaces *before* any RL training—this directly reflects the rollout generation quality during RL sampling; (2) **Post-Training Performance**: We evaluate models after full RL training using CoT, GRPO, and TemplateRL.

As shown in Table 8, template guidance consistently outperforms unrestricted approaches in both stages. In the exploration phase, template-guided sampling achieves 83.3 on AIME25 and 82.8 on GPQA-D, surpassing CoT baseline by 3.3% and 6.0%, respectively. This indicates that templates enhance the diversity and quality of candidate solutions. After RL training, TemplateRL further amplifies this advantage, demonstrating that structured guidance helps models navigate more efficiently toward high-reward regions without limiting their expressiveness. These results confirm that templates **enhance rather than restrict** effective exploration of solution spaces, even for larger powerful models. Future work will systematically investigate this scaling behavior across model sizes to further validate these preliminary findings.

E.6 Discussion on Template Construction Strategy

For TemplateRL, a natural question is whether the MCTS-based template construction could be re-

placed with a simpler end-to-end approach where the policy directly generates abstract templates before producing solutions. Actually, our choice of MCTS is motivated by both theoretical considerations and empirical evidence.

Design Rationale. Our MCTS-based approach is grounded in two key principles:

(1) *Quality of Reasoning Guidance.* MCTS-extracted templates are derived from *verified correct reasoning trajectories* obtained through systematic search and validation. The verification process ensures that each template encodes a genuinely effective problem-solving strategy that has been empirically demonstrated to lead to correct solutions. In contrast, end-to-end generation relies on the policy’s current capabilities to produce templates, which may:

- Generate suboptimal or erroneous templates, especially when the policy is weak or undertrained;
- Perpetuate existing biases in the model’s reasoning patterns rather than discovering novel strategies;

As demonstrated in Figure 5(b) (Section 4.3), weaker models under standard GRPO training frequently experience training collapse due to sparse positive signals from unstructured exploration. MCTS mitigates this by providing high-quality templates from the outset, enabling stable learning even for less capable models.

(2) *Computational Efficiency.* While MCTS appears more complex algorithmically, the computational overhead is minimal in practice. Template construction operates on only 500 seed samples—a one-time investment that incurs negligible cost compared to full RL training on 5K samples. This

small seed set yields a high-quality template library that immediately stabilizes subsequent RL training across thousands of iterations. In contrast, end-to-end approaches must learn template generation *concurrently* with solution generation throughout the entire training process, potentially leading to:

- Unstable training when template quality is poor;
- Wasted computational budgets on low-quality rollouts;
- Slower convergence due to dual optimization.

Empirical Validation. Additionally, we implement and evaluate an end-to-end baseline on Qwen2.5-Math-7B-Base. The end-to-end method follows a two-stage generation process: first, the policy generates an abstract template (e.g., “divide-and-conquer followed by verification”); second, it produces the complete solution conditioned on this template. Both the template and solution are jointly optimized during RL training. Table 9 presents comprehensive results across three benchmarks. We have two findings:

Method	AIME \uparrow	AMC \uparrow	Avg. \uparrow
GRPO (baseline)	16.7	55.0	35.9
End-to-End Generation	23.3	62.5	42.9
TemplateRL	33.3	77.5	55.4

Table 9: Comparison of template construction methods on Qwen2.5-Math-7B-Base. End-to-end generates templates via policy before solving; TemplateRL uses MCTS-verified templates.

(1) MCTS Significantly Outperforms End-to-End. TemplateRL with MCTS-based templates achieves 55.4 average performance, substantially higher than end-to-end generation (42.9). The gap is particularly pronounced on AIME (33.3 vs. 23.3, +43.0%), which requires complex multi-step reasoning where high-quality template guidance is most critical. This validates our hypothesis that verified template extraction is more effective than direct policy generation.

(2) End-to-End Still Improves Over Vanilla GRPO. The end-to-end method achieves 42.9 vs. GRPO’s 35.9 (+19.5%), suggesting that incorporating structured template generation is beneficial. However, the quality of templates matters significantly: MCTS provides an additional +29.1% improvement over end-to-end, demonstrating the value of verified guidance.

Action Space	MATH500 \uparrow	AIME24 \uparrow	AMC \uparrow	Avg. \uparrow
$\{a_5\}$ (i.e. CoT)	76.8	23.3	67.5	55.9
$\{a_1, a_5\}$	80.6	26.7	70.0	59.1
$\{a_1, a_4, a_5\}$	82.0	30.0	72.5	61.5
$\{a_1, a_3, a_4, a_5\}$	82.8	30.0	77.5	63.4
$\{a_1, a_2, a_3, a_4, a_5\}$	83.4	33.3	77.5	64.7

Table 10: Performance comparison across different action spaces on Qwen2.5-Math-7B-Base. Results demonstrate consistent improvements with expanded spaces.

Future Directions. While our results demonstrate MCTS’s current superiority, we acknowledge the potential of hybrid approaches. Future work could explore: (1) Warm-start strategies: Initialize with MCTS-constructed templates to ensure stable early training, then gradually transition to end-to-end refinement once the policy has developed sufficient reasoning capabilities; and (2) Curriculum learning: Start with high-confidence MCTS templates for foundational patterns, then allow the policy to discover novel templates for more complex problems.

E.7 Computational Overhead Analysis.

We provide a detailed quantitative breakdown of the additional computational costs introduced by TemplateRL relative to standard GRPO.

Training overhead.

- *Offline MCTS template construction (one-time cost):* Constructing the template library from 500 seed samples requires approximately 20 minutes on $8 \times A100$ -80GB GPUs (0.33 GPU-hours), representing <3% of the total training budget and amortized across all subsequent training runs and model configurations.
- *Per-iteration overhead:* PCC calculation adds ~ 1 s per batch (batch size 128); structured rollout generation takes $\sim 1.8 \times$ the time of unguided sampling due to iterative action prompting.
- *Total training time:* TemplateRL requires ~ 18 GPU-hours (500 steps on 5K samples) vs. ~ 15 GPU-hours for GRPO (+20% overhead). However, this yields a +27.4% relative performance gain (Table 1, 55.8 vs. 43.8 average accuracy), achieving 3.1 accuracy points per GPU-hour vs. 2.9 for GRPO—a favorable efficiency tradeoff.

Inference overhead.

- *Template retrieval:* ~ 0.05 s per question (PCC calculation + distance ranking), negligible compared to generation time.

- *Generation time*: Structured rollout takes $\sim 1.8\times$ unguided generation due to multi-step action prompting, comparable to self-consistency methods that sample multiple paths.
- *Dynamic updates (optional)*: Add $\sim 2\%$ additional training time (Table 3) and $\sim 0.3s$ per question at test-time (Table 4), acceptable for accuracy-critical applications.

In summary, the +20% training overhead is well justified by substantial performance gains, and inference overhead remains practical for real-world deployment.

E.8 Ablation Study on the Action Space

To evaluate the sensitivity of our method to the action space used for constructing the thought template library, we conduct a comprehensive ablation study using Qwen2.5-Math-7B-Base as the backbone model. We systematically test various action subsets, ranging from a single action to the complete five-action set, across three representative benchmarks.

As shown in Table 10, expanding the action space consistently yields performance improvements, with the full five-action set $\{a_1, a_2, a_3, a_4, a_5\}$ achieving optimal results (64.7 average). Notably, even minimal subsets demonstrate reasonable performance, with the single-action baseline $\{a_5\}$ (equivalent to standard Chain-of-Thought) achieving 55.9 average accuracy. The incremental gains from $\{a_5\}$ to $\{a_1, a_5\}$ (+3.2 points) and subsequent expansions demonstrate that each additional action contributes meaningfully to the reasoning process. These results highlight two key insights: (1) TemplateRL’s robustness to action space configuration, enabling deployment in resource-constrained scenarios with reduced action sets, and (2) the complementary nature of different reasoning actions, where diverse problem-solving strategies collectively enhance performance. This flexibility allows practitioners to adapt the framework based on task-specific requirements.

E.9 Sensitivity to Training Data Size

As mentioned in the main paper, we select training data exclusively from the MATH dataset (Hendrycks et al., 2021), using only 5k MATH level 3-5 problems for RL training. Given the prevalence of large-scale datasets, we investigate whether there exists a scaling law for perfor-

Data Size	MATH500 \uparrow	AIME24 \uparrow	AMC \uparrow	Avg. \uparrow
5K (Original)	83.4	33.3	77.5	64.7
10K (+ 5K Syn.)	84.0	36.7	77.5	66.1
15K (+ 10K Syn.)	84.6	36.7	80.0	67.1

Table 11: Performance scaling with training data size on Qwen2.5-Math-7B-Base. ‘Syn.’ represents synthetic data. Results demonstrate positive scaling behavior, with notable improvements on challenging benchmarks.

Metric	MATH500 \uparrow	AIME24 \uparrow	AMC \uparrow	Avg. \uparrow
SC-only	81.2	30.0	72.5	61.2
PS-only	80.8	26.7	72.5	60.0
PCC-only	83.4	33.3	77.5	64.7
Unified (Equal)	83.9	36.7	77.5	66.0

Table 12: Performance comparison across different pattern-matching metrics on Qwen2.5-Math-7B-Base. ‘Equal’ denotes equal weights for three indicators.

mance versus training data size. Specifically, we augment our original 5K sample set by leveraging the OpenAI-o1 model to generate an additional 10K synthetic problems of similar difficulty, along with their solutions. To ensure solution quality, we employ deepseek-R1 for further verification and correction, filtering out erroneous samples and refining incorrect solutions. This process yields 10k high-quality synthetic data that maintains consistency with the original MATH distribution. We then conduct experiments using mixed datasets of varying sizes: 5k (original), 10k (original + 5k synthetic), and 15k (original + 10k synthetic).

As shown in Table 11, TemplateRL exhibits positive scaling behavior with increased training data. Performance improves from 64.7% (5K) to 66.1% (10K) and further to 67.1% (15K), representing consistent gains of +1.4 and +2.4 percentage points respectively. Notably, the most substantial improvements occur on the challenging AIME24 benchmark (+3.4 percentage points from 5K to 10K), suggesting that additional training data particularly benefits complex reasoning tasks. While the scaling trend is encouraging, we observe that TemplateRL can achieve competitive performance even with limited training data, making it practical for resource-constrained scenarios. More comprehensive investigations into sample diversity and scaling to larger datasets are left for future work.

E.10 Sensitivity to Pattern-Matching Metrics

Beyond the Problem Condition Complexity (PCC) metric (Lee and Heyworth, 2000; Salado and Nilchiani, 2014) employed in the main paper for

retrieving optimal thought patterns, we explore alternative pattern-matching strategies to assess the robustness of our approach.

Metric Design Rationale. Our choice of PCC is motivated by its structural focus: unlike semantic embeddings that primarily capture surface-level similarity in problem statements, PCC quantifies the *structural complexity* of problems based on the number of prior conditions and constraints. This structural feature directly informs the appropriate reasoning strategy—high-PCC problems often require divide-and-conquer approaches to manage multiple interdependent conditions, while simpler low-PCC problems may need only direct reasoning. This explicit complexity-strategy mapping enhances both interpretability (templates are matched based on observable problem structure) and robustness (matching is less sensitive to superficial phrasing variations).

Alternative Metrics Comparison. To empirically validate PCC’s effectiveness, we investigate three additional pattern-matching strategies:

- **Subquestion-count (SC):** The number of subquestions an unstructured problem can be decomposed into, reflecting reasoning granularity;
- **Problem semantics (PS):** Semantic similarity-based matching using sentence embeddings (all-MiniLM-L6-v2), representing surface-level textual similarity;
- **Unified ranking (UR):** A composite metric that computes PCC, SC, and PS for each incoming question, performs triple relevance ranking across thought templates, and selects top candidates through weighted averaging ($w_{PCC}=0.5$, $w_{SC}=0.3$, $w_{PS}=0.2$).

As shown in Table 12, while the unified ranking metric achieves the highest overall performance (66.0 average), the PCC-only approach demonstrates competitive results (64.7 average) with significantly lower computational overhead. Notably, semantic-based matching (PS-only) shows the weakest performance (60.2 average), which aligns with our hypothesis: mathematical reasoning patterns are inherently structural rather than semantic—two problems may share similar wordings yet require entirely different reasoning strategies, while problems with distinct phrasings may share the same underlying structural complexity. Surface-level semantic similarity fails to capture

Seed Data	MATH500	AIME24 \uparrow	AMC \uparrow	Avg. \uparrow
100	77.4	20.0	65.0	54.1
200	81.0	26.7	72.5	60.0
500	83.4	33.3	77.5	64.7
1000	85.2	36.7	77.5	66.5

Table 13: Impact of seed data size.

these critical strategic distinctions, whereas PCC’s structural focus naturally aligns template matching with the actual reasoning demands of problems.

Given that PCC provides a simple, interpretable, and effective solution with minimal computational cost, we adopt it as our default pattern-matching metric. The strong performance of the unified approach (UR) suggests that combining multiple metrics can further enhance matching quality, and future work could explore more sophisticated weighting schemes or learned combination functions to optimize this trade-off between performance and efficiency.

E.11 Sensitivity to Seed Data Size.

As shown in Table 13, performance scales with seed data size: average accuracy improves from 38.7% (100 samples) to 51.4% (1,000 samples). Notably, TemplateRL achieves competitive performance even with 100 samples, highlighting practicality in resource-constrained settings. We use 500 samples by default, balancing performance enhancement and data efficiency.

E.12 Statistics on Reasoning Actions

We present the occurrence statistics of different reasoning actions in the constructed thought template library in Table 14. For brevity, DC, SR, SA, OST, and CoT denote DIVIDE_AND_CONQUER, SELF_REFLECTION, SYSTEM_ANALYSIS, ONE_STEP_THOUGHT, and CHAIN_OF_THOUGHT, respectively. The results reveal a relatively balanced distribution across actions, with DC and OST demonstrating the highest frequencies (31.4% and 30.3%), followed by CoT (25.5%), while SR and SA show lower occurrences (7.2% and 5.7%). This distribution pattern indicates that during MCTS exploration, OST, DC, and CoT are more effective at generating correct solutions and resulting in their elevated ratio within the thought template library.

Action	DC (a_1)	SR (a_2)	SA (a_3)	OST (a_4)	CoT (a_5)
Ratio	31.3%	7.2%	5.7%	30.3%	25.5%

Table 14: Statistics of reasoning actions: proportion of each action relative to all actions in the template library.

E.13 Statistical Analysis

To statistically evaluate the significance of performance differences between TemplateRL and baseline methods, we apply the nonparametric Wilcoxon signed-rank test (Kotz and Johnson, 1992). This statistical test is specifically designed to compare paired observations when data may not follow a normal distribution, making it particularly suitable for performance analysis across multiple benchmarks. The Wilcoxon signed-rank test evaluates whether there is a significant difference between paired observations through the following procedure:

1. **Calculate differences:** For each benchmark pair, compute the difference $D_i = X_i - Y_i$ where X_i represents TemplateRL performance and Y_i represents baseline performance.
2. **Rank differences:** Take absolute values $|D_i|$ and rank them from smallest to largest as R_i , with average ranks assigned for ties.
3. **Assign signs to ranks:** For each difference D_i , assign its sign to the corresponding rank: $R'_i = \text{sign}(D_i) \cdot R_i$.
4. **Calculate rank sums:** Compute positive and negative rank sums: $W^+ = \sum_{D_i > 0} R'_i$ and $W^- = \sum_{D_i < 0} R'_i$.
5. **Determine test statistic:** The test statistic is $W = \min(W^+, W^-)$.
6. **Calculate p-value:** Derive the p-value from the distribution of test statistic W .

We test the null hypothesis H_0 : no significant difference between methods against the alternative hypothesis H_1 : significant difference exists. Comparing TemplateRL with GRPO on Qwen2.5-Math-7B-Base across all benchmarks, we obtain a p-value of 0.0156. Using a significance level of $\alpha = 0.05$, we reject the null hypothesis ($p < 0.05$), providing strong statistical evidence that TemplateRL significantly enhances reasoning performance compared to standard GRPO training.

F Case Study

To further analyze the improvements of TemplateRL over conventional GRPO, we compare their reasoning processes on representative mathematical problems from the MATH dataset in Figure 13 and Figure 14.

In Figure 13, we observe how both methods approach an arithmetic mean problem. GRPO produces a solution with scattered notation and repetitive statements, particularly evident in its final steps where it repeatedly states “The problem involves ... final answer is 46.” In contrast, TemplateRL demonstrates a more structured approach by explicitly introducing a “step by step” thinking process. The thought-augmented process methodically builds upon each reasoning step, clearly identifying the relationship between the original sum (984), the highest score (98), and the remaining scores to derive the lowest score (46). This structured approach results in a more readable and logically coherent solution.

Figure 14 presents a more challenging problem involving complex numbers and Vieta’s formulas. Here, the limitations of GRPO become more pronounced. While GRPO initially applies the correct formula, its reasoning process deteriorates into incoherent text fragments and coding artifacts (e.g., “Tre localVEC?” and various non-mathematical expressions). This demonstrates how GRPO struggles with maintaining coherent reasoning for complex problems. In contrast, TemplateRL maintains its structured approach throughout, clearly stating the problem context, applying Vieta’s formulas with proper explanation, and presenting a clean, coherent solution without extraneous text or errors.

G Additional Discussion

G.1 Justification for Choosing MCTS.

The selection of MCTS is driven by both task-specific requirements and its comparison with other search algorithms.

Task Requirements: Our task involves constructing complex reasoning chains from a vast combinatorial action space. This setting demands

an algorithm that can efficiently search and optimize within such a large space. MCTS is particularly well-suited for this, as it incrementally explores the most promising trajectories while maintaining flexibility through stochastic sampling.

Limitations of Other Search Methods: We investigated alternatives such as Breadth-First Search (BFS) and Depth-First Search (DFS), and identified several issues:

- DFS is vulnerable to early-stage errors. A sub-optimal initial action can lead the search deep into unproductive paths, making recovery and backtracking difficult in multi-step reasoning where early decisions are often uncertain.
- BFS is computationally expensive, exploring the reasoning space level by level with exponential growth in candidate states, making it intractable for practical problems.

MCTS Advantages: MCTS offers a superior balance between exploration and exploitation:

- Compared to DFS, MCTS does not commit to a single path. It uses rollout and backpropagation to update the estimated value of nodes, allowing the search to adaptively shift away from poor paths.
- Compared to BFS, MCTS grows the search tree asymmetrically, guided by UCT (Upper Confidence Bound for Trees). This selective expansion enables it to focus computation on the most promising subtrees, significantly improving efficiency.

Computational Efficiency: Importantly, MCTS template construction incurs negligible computational overhead in our framework. On 8xA100-80GB GPUs, constructing the template library from 500 seed samples requires about *20 minutes*, compared to the *~10-20 hours* required for full RL training on 5K samples (500 steps). Moreover, this is a one-time cost: once constructed, the template library can be reused across different model scales, architectures, and even domains (as demonstrated by our cross-domain transfer results). In most scenarios, 500 seed samples are sufficient for building the initial library.

In summary, given its ability to handle large search spaces, its robustness to early decision errors, its structural alignment with sequential reasoning, and its minimal computational cost, MCTS

emerged as an effective and efficient approach for our reasoning framework.

G.2 Justification of Key Design Choices

We justify two critical decisions:

(1) Five-action space: The action set {DC, SR, SA, OST, CoT} was designed based on cognitive science literature (Kahneman, 2011) and validated through pilot studies. Testing alternatives on 500-sample validation (diverse task samples) showed: minimal 2-action set (62.4% average accuracy), our 5-action design (71.3%), extended 8-action set (72.4%, only +1.1). The design principle is domain-agnostic meta-strategies rather than domain-specific operations.

(2) PCC over semantic similarity: PCC (Problem Condition Complexity) correctly distinguishes problems by reasoning requirements rather than surface similarity. For example, “find x where $x^2 + 3x - 4 = 0$ ” (PCC=1) vs. “find x, y where $x + y = 5$ and $xy = 6$ ” (PCC=2) appear semantically similar but require different strategies (direct CoT vs. DC→CoT).

G.3 Why A Small Set of Generic Patterns Can Generalize?

Our templates encode high-level strategic patterns rather than problem-specific solutions. These templates capture abstract reasoning strategies such as “decompose → solve subproblems → verify” that operate above the surface-level content. While individual problems vary widely in their mathematical details, the underlying reasoning structures often follow recognizable strategic patterns, which is analogous to how humans apply divide-and-conquer or proof-by-contradiction across diverse domains. We believe that not only can individual actions generalize, but their combinations can as well, particularly for complex real-world scenarios where solutions emerge from abstracting more sophisticated action sequences.

G.4 Are Templates Optimized for Test Sets?

No. As shown in Figure 6, templates constructed from mathematical problems generalize effectively to diverse domains: BALROG (+6.1% performance gains for agentic reasoning), GPQA-D (+6.6% gains for graduate-level science), and MMLU-Pro (+7.4% gains for general knowledge). These substantial cross-domain improvements demonstrate that templates are general, not test-set optimized. These templates capture

transferable strategic patterns rather than domain-specific solutions.

G.5 How does performance scale with template set size?

In our preliminary experiments, we explored the impact of varying template set sizes and observed a clear pattern: performance improves substantially when increasing template set size below 100 templates, but gains begin to slow beyond this threshold. This suggests an optimal balance point where the template library is sufficiently diverse to cover common reasoning patterns without introducing retrieval complexity or redundancy. Future work could investigate adaptive mechanisms that dynamically adjust the template set based on problem domain or difficulty.

G.6 Robustness to Seed Set and Cross-Domain Generalizability

A potential concern is whether the template library is overly dependent on the quality or domain coverage of the initial 500 seed samples. However, our empirical results demonstrate that the high-level thought patterns abstracted via MCTS possess remarkable task-agnostic properties. As shown in Figure 6, templates derived exclusively from mathematical seeds significantly enhance performance in entirely different domains, such as agentic embodied reasoning (BALROG, +6.1%) and graduate-level science (GPQA-D, +6.6%).

This indicates that TemplateRL captures universal “System 2” strategic logic—including problem decomposition, iterative self-reflection, and constraint analysis—rather than surface-level heuristics. The high-level nature of these patterns mitigates the risk of the model being misled by specific seed biases into suboptimal paths. Therefore, in practical applications, selecting a seed set that covers a wide range of difficulty levels is sufficient to build a robust template library that facilitates effective policy optimization across diverse tasks.

G.7 Distinction between TemplateRL and Related Works

A related line of work, ReMA (Wan et al., 2025), learns meta-cognitive strategies through multi-agent reinforcement learning, where agents discover abstract reasoning operators via agent-level reward signals. While ReMA focuses on *implicit* meta-strategy discovery through agent interactions,

TemplateRL constructs *explicit*, interpretable template libraries via systematic MCTS exploration, enabling immediate human inspection, manual editing, and efficient cross-domain transfer without the multi-agent training overhead.

G.8 Future Work

In future work, we plan to extend TemplateRL along several directions. First, we aim to scale the framework to more complex agentic scenarios (Tang et al., 2025; Chen et al., 2025; Wu et al., 2026a; Xu et al., 2026), where dynamic branching and multi-step decision-making demand richer template structures beyond the current five-action space. We also hope to incorporate the entropy mechanism into training to further enhance template diversity (Kang et al., 2025). Second, we will explore integrating TemplateRL with heterogeneous model orchestration (Wu et al., 2026b), enabling structured template guidance across multi-domain reasoning tasks that require coordinating specialized tools and models. Additionally, we aim to extend templates to more general domains, such as dialogue, retrieval-augmented generation, role-playing, and optimize templates for more diverse objectives (Chen et al., 2024; Wu et al., 2025; Liao et al., 2025). When deployed in multi-agent settings, we further plan to investigate how communication topologies influence the propagation of structured templates across agents (Shen et al., 2025), and enhance the robustness of template-guided coordination against unknown adversarial conditions (Miao et al., 2025). Third, to improve deployment efficiency, we plan to combine TemplateRL with model compression (Feng et al., 2025) and inference acceleration techniques (Shen et al., 2026), reducing the computational footprint of structured rollout generation while preserving the performance gains demonstrated in this work.

Question The arithmetic mean of 12 scores is 82. When the highest and lowest scores are removed, the new mean becomes 84. If the highest of the 12 scores is 98, what is the lowest score?

GRPO

version),(elements票价

$$\sum_{i=1}^{12} x_i = 12 \times 82 = 984$$

2. Given that the highest score is 98:

$$\sum_{i=1}^{12} x_i - x_1 - x_{12} = 984 - x_1 - x_{12}$$

3. After removing the highest and lowest scores, there are 10 scores left:

$$\sum_{i=2}^{11} x_i = 10 \times 84 = 840$$

So, we have: $984 - x_1 - x_{12} = 840$

4. Solve for the sum of the highest and lowest scores:
 $x_1 + x_{12} = 984 - 840 = 144$

5. We know the highest score is 98: $x_1 = 98$

6. Substitute x_1 into the equation for the sum of the highest and lowest scores:
 $98 + x_{12} = 144$

Thus, the lowest score is: 46

The lowest score is 46. The problem involves ... final answer is 46. The problem involves ... final answer is 46.
...(repeat)

TemplateRL

Let's think step by step. We **consider the existing reasoning steps, then give the next one-step thought.**

1. The arithmetic mean of 12 scores is 82. This means that the sum of the 12 scores is $12 \times 82 = 984$.
2. When the highest and lowest scores are removed, the new mean becomes 84. Since there are now 10 scores, the sum of these 10 scores is $10 \times 84 = 840$.
3. The highest score is 98. Therefore, the sum of the remaining 11 scores (excluding the highest and lowest scores) is $984 - 98 - \text{lowest score} = 840$. Solving for the lowest score, we get:
 $984 - 98 - \text{lowest score} = 840$
 $\implies \text{lowest score} = 984 - 98 - 840 = 46$
4. Check what the question asks. The question asks for the lowest score, and we found it to be 46.
5. The answer is: 46

Figure 13: Comparison of GRPO and TemplateRL for a simple algorithm problem from the MATH dataset.

Question

$$2a^3 + 7a^2 - 8a + 5 = 0,$$

Let a, b, c be distinct complex numbers such that $2b^3 + 7b^2 - 8b + 5 = 0$, Compute the value of the product

$$2c^3 + 7c^2 - 8c + 5 = 0.$$

GRPO

To solve for the product abc , we start by noting that a, b , and c are the roots of the polynomial equation

$$2x^3 + 7x^2 - 8x + 5 = 0.$$

According to Vieta's formulas, for a cubic polynomial of the form $ax^3 + bx^2 + cx + d = 0$, the product of the roots taken one at a time (i.e., the product of the roots) is given by $-\frac{d}{a}$. Here, the polynomial is $2x^3 + 7x^2 - 8x + 5 = 0$, so we have $a = 2, b = 7, c = -8$, and $d = 5$.

$$\text{Therefore, the product of the roots is } abc = -\frac{5}{2}.$$

Thus, the value of the product abc is $abc = -\frac{5}{2}$.

The localVEC?

\end{aligned}\}

\.getLong local around? joint Playlist? local]. former]

\end{sh down\end{document}')local around'? local剩a by

Yuval保存 Four local around? \$a\$)\$ local certainly yet?

local^{a}\$ local广 \$ local[local]^ local越来越少 local}

TemplateRL

Let's think step by step. The numbers a, b , and c are the roots of the polynomial equation

$$2x^3 + 7x^2 - 8x + 5 = 0.$$

By Vieta's formulas, for the polynomial $ax^3 + bx^2 + cx + d = 0$, the product of the roots taken one at a time is given by $-\frac{d}{a}$, provided the roots are distinct. In this case, the polynomial is $2x^3 + 7x^2 - 8x + 5 = 0$, so the product of the roots abc is given by $abc = -\frac{5}{2}$.

Therefore, the answer is: $abc = -\frac{5}{2}$

Figure 14: Comparison of GRPO and TemplateRL for a difficult algorithm problem from the MATH dataset.