NUDGING BEYOND THE COMFORT ZONE: EFFICIENT STRATEGY-GUIDED EXPLORATION FOR RLVR

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) is a scalable paradigm for improving the mathematical reasoning of large language models, but it is fundamentally limited by exploration: the policy can only improve on trajectories it has already sampled. Sampling more rollouts alleviates this problem, but at a prohibitive computational cost, while objective-level modifications offer little control over *what* is explored. We propose NUDGERL, a framework for structured, diversity-driven exploration in RLVR whose core component, *Strategy Nudging*, conditions each rollout on a lightweight strategy-level context, without requiring the context generator to solve the problem itself. To learn from such exploration, we decompose the advantage into inter- and intra-context terms and add a policy correction term that transfers discovered behaviors back to the base policy. Across five mathematical reasoning benchmarks, NUDGERL with 8 rollouts matches the strongest GRPO baseline using 32 rollouts with roughly $5\times$ fewer total tokens and $2.9\times$ less training compute. On code generation, it also outperforms GRPO with 64 rollouts using only one-eighth as many. These results establish strategy-guided exploration as a compute-efficient alternative to brute-force rollout scaling.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) has emerged as a powerful paradigm for improving the reasoning capabilities of large language models (LLMs) (Team et al., 2025; Lambert et al., 2024). By leveraging automatic verification, most naturally the correctness of a final answer in mathematical reasoning, methods such as *Group-Relative Policy Optimization* (GRPO) (Shao et al., 2024; Guo et al., 2025) enable scalable post-training without requiring dense supervision.

Despite this success, RLVR remains fundamentally limited by its ability to explore the space of reasoning trajectories (Wu et al., 2025; Yue et al., 2025): the policy can only learn from trajectories it has actually sampled. A natural approach is therefore to increase the number of rollouts, which improves the chance of discovering rare but useful trajectories (Hu et al., 2025), but brute-force rollout scaling quickly becomes computationally prohibitive. Recent work has explored alternatives such as entropy regularization and decoupled clipping (Zhang et al., 2025a; Yu et al., 2025), which broaden exploration at the distribution level but provide limited control over *what* is explored. A complementary line of work uses *privileged information*, such as oracle solutions or intermediate reasoning steps, to make successful trajectories easier to discover (Zhang et al., 2025b; Qu et al., 2026; Liao et al., 2026; Song et al., 2026). While effective, these methods primarily improve feasibility through strong supervision signals that can be costly to obtain and difficult to scale.

We address this exploration bottleneck by explicitly structuring the reasoning space. We propose NUDGERL, a framework whose core mechanism, *Strategy Nudging*, conditions each rollout on a lightweight strategy-level context, such as a mathematical theorem or technique (e.g., *"Generating functions"*). By assigning different contexts across rollouts, Strategy Nudging steers the policy toward distinct reasoning modes that it might otherwise ignore under naive sampling. Crucially, unlike methods based on privileged supervision (Chen et al., 2025; Qu et al., 2026), the context generator only provides high-level strategic guidance and does not need to solve the problem itself.

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

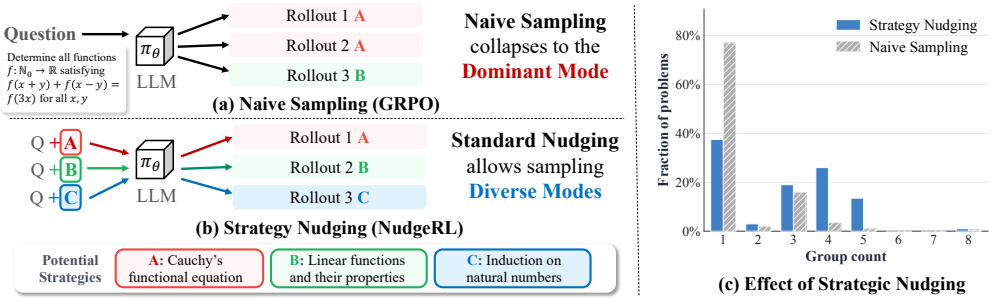


Figure 1: **Improving exploration diversity through Strategy Nudging.** (a) Naive sampling methods (e.g., GRPO) often collapse to a dominant reasoning mode, limiting the exploration of the reasoning space. (b) NUDGERL introduces Strategy Nudging, which appends lightweight strategy-level context to the input, forcing the model to traverse diverse reasoning modes. (c) As a result, Strategy Nudging significantly increases the number of distinct reasoning approaches discovered compared to the baseline, effectively mitigating the exploration bottleneck. Additional details are in Sec. A.

Learning from such context-conditioned exploration introduces two challenges. First, because Strategy Nudging assigns different contexts to rollouts within the same problem batch, the resulting samples form context-specific sub-groups. Reward variation therefore reflects both trajectory quality and context-level effects, making standard group-wise advantage estimation unsuitable for credit assignment across these groups. Second, trajectories are sampled from context-conditioned prompts, whereas the target policy must operate without external contexts at inference time. Improvements discovered under forced exploration therefore need not automatically transfer to the base policy.

To address these challenges, we introduce two complementary learning mechanisms. First, an *Inter-Intra Group Advantage* decomposes credit assignment into an intra-context component that measures trajectory quality within a strategy and an inter-context component that captures the relative reliability of different strategies. Second, a *policy correction-augmented objective* transfers successful behaviors discovered under context-conditioned exploration back to the policy under the original prompt. Together, these components allow NUDGERL to explore diverse reasoning modes during training while retaining a context-free policy at inference time.

Across five mathematical reasoning benchmarks and two model families, NUDGERL with only 8 rollouts matches or exceeds the strongest GRPO baseline using 32 rollouts, while requiring approximately $5\times$ fewer total tokens and $2.9\times$ less training compute. The benefit also extends beyond mathematics: on code generation, NUDGERL with 8 rollouts outperforms GRPO even with 64 rollouts. These results suggest that explicitly controlling *where* the policy explores can be substantially more efficient than simply scaling *how much* it samples.

2 Related Work

2.1 Reinforcement Learning with Verifiable Reward

Reinforcement learning with verifiable reward (RLVR) (Lambert et al., 2024) has been widely adopted as a scalable and effective post-training paradigm for improving the reasoning capability of LLMs (Shao et al., 2024; Yu et al., 2025; Blakeman et al., 2025). As RLVR has become the mainstream approach to LLM post-training, its limitations arising from insufficient exploration such as marginal performance gains (Hu et al., 2025) and mode collapse (Yue et al., 2025; Wu et al., 2025) have likewise become apparent. To address this issue, a recent line of work (Zhang et al., 2025a; Deng et al., 2025; Walder & Karkhanis, 2025; Yu et al., 2025) promotes exploration at the objective level through entropy bonuses, direct pass@k objectives, or decoupled clipping. Such objective-level mechanisms, however, afford little control over *what* is explored.

2.2 Exploration with Privileged Information

A complementary line of work (Chen et al., 2025; Qu et al., 2026; Zhang et al., 2025b; Liao et al., 2026; Setlur et al., 2026) leverages privileged information to enhance exploration. These methods typically use solution-derived signals, such as oracle solutions or expert prefixes to increase

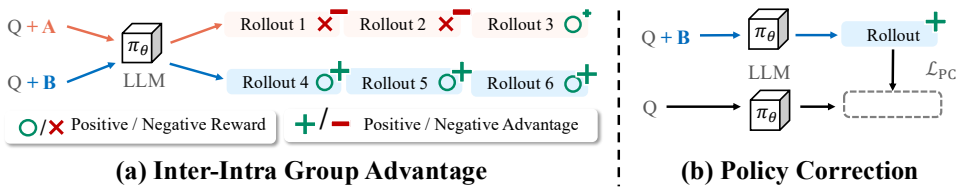


Figure 2: **Overview of the NudgeRL learning mechanism. (a) Inter-Intra Group Advantage:** Demonstrates credit assignment that emphasizes reliable contexts (i.e., $\lambda \in (1, 2]$). A successful rollout from a consistently high-reward context (Strategy B) receives a larger positive advantage than a rare success from a low-reward context (Strategy A). **(b) Policy Correction:** High-quality trajectories discovered via context-conditioned exploration ($Q + B$) are distilled back into the base policy using $\mathcal{L}_{\text{PC}}$, allowing the model to internalize effective reasoning modes.

the likelihood of sampling successful trajectories on hard problems. However, they are primarily *feasibility*-oriented and rely on privileged supervision that can be costly or difficult to scale. In contrast, NUDGERL requires only strategy-level guidance from an external model, which need not be capable of solving the problem itself.

3 PRELIMINARIES

3.1 GROUP-RELATIVE POLICY OPTIMIZATION (GRPO)

We consider an empirical distribution of prompts $x_0 \in \mathcal{D}$. For each prompt x_0, a policy π_θ generates a group of N rollouts $\{y_i\}_{i=1}^N$, where each rollout is sampled as $y_i \sim \pi_\theta(\cdot \mid x_0)$. Each rollout is evaluated by a verifiable reward function $R(x_0, y_i) \in \{0, 1\}$. Unlike standard PPO (Schulman et al., 2017), which typically estimates advantages using a learned value function, GRPO (Shao et al., 2024; Guo et al., 2025) derives advantages from group-wise rewards. For rollouts sampled from the same prompt x_0, let $r_i = R(x_0, y_i)$ denote the reward of rollout y_i. The group-wise advantage is then defined as:

$$\hat{A}_i = \frac{r_i - \mu}{\sigma + \delta}, \tag{1}$$

where μ and σ are the reward mean and standard deviation within the group, and $\delta > 0$ is used for numerical stability. This yields a relative advantage estimate without training a value function.

The policy is then optimized with a PPO-style clipped objective:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\mathbb{E}_{y \sim \pi_\theta(\cdot|x_0)} \left[\min \left(r_0 \hat{A}, \text{clip}(r_0, 1-\epsilon, 1+\epsilon) \hat{A} \right) \right], \text{ where } r_0 = \frac{\pi_\theta(y|x_0)}{\pi_{\text{old}}(y|x_0)}. \tag{2}$$

3.2 MOTIVATION: FROM EXPLORATION TO PERFORMANCE GAIN

To understand why exploration is a fundamental bottleneck in RLVR, we look beyond trajectory-level rewards and examine how the probability mass of generated tokens shifts during training. Hu et al. (2025) characterize the expected one-step performance improvement (ΔQ_{pos}) in RLVR as:

$$\Delta Q_{\text{pos}} = \frac{\eta}{N} \left[(1 - S_R) Q_{\text{neg}} A_2 + S_R Q_{\text{pos}} B_2 + S_R (Q_{\text{pos}} U_{\text{neg},2} - Q_{\text{neg}} U_{\text{pos},2}) \right], \tag{3}$$

where Q_{pos} and $Q_{\text{neg}} = 1 - Q_{\text{pos}}$ denote the total probability mass of correct and incorrect tokens, η is the learning rate, and N is the number of rollouts. A_2 and B_2 are the second moments of *sampled* correct and incorrect tokens, while $U_{\text{pos},2}$ and $U_{\text{neg},2}$ are those of *unsampled* correct and incorrect tokens. $S_R \in [0, 1]$ represents the net reward contribution from sampled tokens.

Since $S_R \in [0, 1]$, the first two terms in Eq. 3 are non-negative and drive learning forward. The third term, however, acts as a potential penalty. Because incorrect tokens typically dominate the probability mass ($Q_{\text{neg}} \gg Q_{\text{pos}}$), a large $U_{\text{pos},2}$, meaning the model has significant probability mass on correct trajectories that it simply *fails to explore*, creates a dominant negative force that hinders performance gain. Therefore, the core bottleneck of RLVR lies in the *unexplored* correct regions.

Limitations of rollout scaling. To mitigate this penalty, a naive solution is to increase the rollout size N. Hu et al. (2025) show that for a collection of tokens with probabilities $\{p_i\}$, the expected unsampled second moment after N draws is:

$$\sum_i p_i^2 (1 - p_i)^N, \tag{4}$$

which decreases monotonically with N. However, tokens with small p_i decay slowly, so fully covering long-tail correct trajectories requires prohibitively large rollout budgets.

This highlights the limitation of blindly scaling N to reduce the unexplored correct mass ($U_{\text{pos},2}$). Long-tail correct trajectories remain unlikely to be sampled even under large N, suggesting the need for a *structured exploration* mechanism that can *efficiently* expose such latent trajectories.

4 NUDGERL

We introduce **NUDGERL**, a framework for structured exploration and learning in RLVR. NUDGERL consists of three components: (i) **Strategy Nudging**, which conditions rollout generation on *strategy-level* contexts to induce diverse reasoning trajectories; (ii) **Inter-intra Group Advantage**, a credit assignment method that enables controlled exploration and exploitation of strategies; and (iii) **Policy Correction augmented RL objective** that learns from context-conditioned rollouts and distill effective strategies into the policy under the original prompt for inference without external context.

4.1 STRATEGY NUDGING: STRUCTURED EXPLORATION VIA STRATEGY-LEVEL CONTEXTS

Given that prior work (Hu et al., 2025) alleviates the exploration bottleneck by reducing unsampled probability mass through larger rollout budgets, a natural question arises: *how many rollouts are required to reliably discover a rare trajectory?* To quantify this discovery cost, consider a rare trajectory y with $\pi(y|x_0) \ll 1$. The expected number of rollouts required to observe y at least once is $\mathbb{E}[\#\text{ rollouts}] = \frac{1}{\pi(y|x_0)}$. This implies that for low-probability trajectories, the required rollout budget grows prohibitively large. In practice, naive rollout scaling repeatedly samples from high-probability modes of the current policy, leading to diminishing returns in covering rare trajectories.

This motivates conditioning generation on a context c that can shift the sampling distribution toward otherwise rare trajectories. If such a context increases the probability of a trajectory y, i.e., $\pi(y|x, c) \gg \pi(y|x)$, then its expected number of rollouts becomes $\mathbb{E}[\#\text{ rollouts} \mid c] = \frac{1}{\pi(y|x_0,c)} \ll \frac{1}{\pi(y|x_0)}$. Thus, contexts need not provide a solution; they can serve as lightweight controls that alter the sampling distribution and reduce the cost of discovering rare trajectories.

Strategy Nudging. Even though context conditioning can improve exploration efficiency in principle, simply placing multiple contexts in a single prompt leaves the choice of strategy to the policy, which may ignore some contexts and repeatedly follow dominant reasoning patterns. To enforce coverage over contexts, we instead assign a single sampled context to each rollout before generation.

Let $\mathcal{C}(x_0) = \{c_1, \ldots, c_M\}$ denote a pool of strategy-level contexts for the original prompt x_0. For each rollout index i, we begin by sampling $c^{(i)} \sim \text{Uniform}(\mathcal{C}(x_0))$. To avoid relying exclusively on the context pool and to retain compatibility with the original prompt, we further apply context dropout. Specifically, we sample a mask $b^{(i)} \sim \text{Bernoulli}(1 - p_{\text{drop}})$ and define the context as:

$$z^{(i)} = \begin{cases} c^{(i)}, & b^{(i)} = 1, \\ \emptyset, & b^{(i)} = 0. \end{cases} \tag{5}$$

We then construct the final prompt $x_1^{(i)} = (x_0, z^{(i)})$, and generate $y_i \sim \pi_\theta(\cdot \mid x_1^{(i)})$. By varying $z^{(i)}$ across rollout indices, Strategy Nudging induces diversity at the input-conditioning level, rather than relying solely on sampling from a single prompt. Details on generating $\mathcal{C}$ are in Sec. A.

Context-induced rollout diversity. To verify that Strategy Nudging induces the intended diversity, we compare it against naive sampling without context conditioning. For each prompt, both methods generate 8 rollouts in total: Strategy Nudging samples 4 rollouts from each of 2 contexts without

context dropout, whereas the baseline samples all 8 rollouts from the base policy under the original prompt. We then cluster the reasoning structures using an LLM-as-a-judge (`gpt-4o-mini` (OpenAI, 2024)) and measure the number of distinct clusters; additional details are provided in Sec. A.

As shown in Fig. 1, Strategy Nudging more often increases the number of distinct reasoning structures relative to naive sampling, whereas the base policy frequently collapses to similar patterns. This suggests that Strategy Nudging diversifies exploration before any policy update is applied, allowing the rollout set to cover a broader range of reasoning modes under the same rollout budget.

4.2 Inter-Intra Group Advantage: Learning to Balance Exploration between Strategies

To effectively learn from context-forced exploration, we propose the *Inter-Intra Group Advantage*, which assigns credit through two complementary signals: an *intra*-context signal, capturing trajectory quality under the same conditioning context, and an *inter*-context signal, capturing the relative reliability of the context itself.

Given sampled rollouts $\{y_i\}_{i=1}^N$ with rewards $r_i = R(x_0, y_i)$, we group them according to their assigned contexts. The set of context groups is defined as

$$\mathcal{G}(x_0) = \text{Unique}(\{z^{(i)}\}) \subseteq \mathcal{C}(x_0) \cup \{\emptyset\}. \tag{6}$$

For each group $g \in \mathcal{G}(x_0)$, we define the index set $I_g = \{i \mid z^{(i)} = g\}$, which partitions all rollouts. We then compute both context-level and global reward baselines:

$$\bar{r}_g = \frac{1}{|I_g|} \sum_{i \in I_g} r_i, \quad \bar{r} = \frac{1}{N} \sum_{i=1}^{N} r_i. \tag{7}$$

Using these baselines, we define the advantage as:

$$\hat{A}_i = \frac{A_i - \mu_A}{\sigma_A + \delta}, \quad \text{where } A_i = \begin{cases} (r_i - \bar{r}_{z^{(i)}}) + \lambda(\bar{r}_{z^{(i)}} - \bar{r}) & \text{if } z^{(i)} \neq \emptyset, \\ r_i - \bar{r} & \text{if } z^{(i)} = \emptyset. \end{cases} \tag{8}$$

μ_A and σ_A are the mean and standard deviation of $\{A_i\}$, and $\delta > 0$ ensures numerical stability.

Because advantages determine the direction of the policy update, they should remain consistent with the underlying rewards while allowing context-level preferences to affect credit assignment.

Proposition 4.1. *Consider two trajectories y and y' sampled from context groups z and z', with rewards r and r', respectively. Let $\bar{r}_z$ and $\bar{r}_{z'}$ denote the corresponding context reward means, and let A and A' denote their advantages. In the binary reward setting, if $\lambda \in (0, 2]$, then:*

$$r > r' \Rightarrow A > A'. \tag{9}$$

Thus, for $\lambda \in (0, 2]$, a higher reward always receives a higher advantage, ensuring consistency with the underlying objective; context only affects the relative ordering among equal-reward trajectories. For equal-reward trajectories, λ controls the context-level preference: $\lambda < 1$ favors successes from lower-reward contexts, encouraging exploration of less typical contexts, whereas $\lambda > 1$ favors successes from higher-reward contexts, emphasizing more reliable contexts. The neutral case $\lambda = 1$ treats equal-reward trajectories identically across contexts; the $\lambda > 1$ case is illustrated in Fig. 2 (a). Proof of proposition 4.1 and generalization to more diverse reward designs are discussed in Sec. B.

4.3 Training Objective

Although Strategy Nudging improves exploration by sampling rollouts from context-conditioned prompts $x_1 = (x_0, z)$, the target policy at inference time should operate without external contexts. Therefore, useful trajectories discovered under x_1 must be transferred to the base policy $\pi_\theta(\cdot \mid x_0)$.

To bridge this gap, we introduce a policy correction term adapted from the advantage-weighted distillation approach of Song et al. (2026), which directly updates the policy using trajectories sampled by the context-conditioned input x_1:

$$\mathcal{L}_{\text{PC}}(\theta) = -\mathbb{E}_{y \sim \pi_\theta(\cdot|x_1)} \left[\hat{A} \log \pi_\theta(y|x_0) \right]. \tag{10}$$

Table 1: **Main results** comparing rollout scaling and context-based exploration. We report *pass*@1 estimated using 128 rollouts. Best results are represented as **bold** and second best as underline.

Model	Method	Rollouts (N)	AIME24	AIME25	AMC23	MATH500	APEX	Average
Qwen3-4B-Instruct	Base model	–	0.374	0.352	0.653	0.592	0.036	0.402
	GRPO	8	0.444	0.367	0.749	0.668	0.040	0.454
		16	0.454	0.355	0.840	0.655	0.045	0.470
		32	0.451	0.370	**0.881**	**0.674**	**0.058**	0.487
		64	0.415	0.324	0.848	0.641	0.027	0.451
	NUDGERL	8	**0.482**	**0.393**	0.857	0.660	0.053	**0.489**
Olmo3-7B-Instruct-SFT	Base model	–	0.134	0.118	0.467	0.384	0.021	0.225
	GRPO	8	0.187	0.159	0.537	0.434	0.025	0.268
		16	0.188	0.176	0.548	0.461	0.023	0.279
		32	**0.195**	0.176	0.553	0.459	0.024	0.281
		64	0.081	0.053	0.349	0.291	**0.027**	0.160
	NUDGERL	8	0.190	**0.179**	**0.563**	**0.468**	0.025	**0.285**

Table 2: **Results on LiveCodeBench v6** comparing rollout scaling and context-based exploration. We report *pass*@1 estimated using 128 rollouts, including both the per-source breakdown and the overall performance. Best results are shown in **bold**, and second-best results are underlined.

			Per-Source Breakdown		LiveCodeBench v6
Model	Method	Rollouts (N)	AtCoder *pass*@1	LeetCode *pass*@1	Total *pass*@1
Qwen2.5-Coder-7B-Instruct	Base model	–	0.325	0.254	0.299
	GRPO	8	0.325	0.276	0.308
		16	0.332	0.285	0.315
		32	0.335	0.277	0.314
		64	0.334	0.287	0.317
	NUDGERL	8	**0.352**	**0.304**	**0.335**

Unlike standard behavior cloning, this formulation selectively emphasizes trajectories with high normalized advantage, ensuring that only useful behaviors discovered under diverse contexts contribute to the update of $\pi_\theta(\cdot \mid x_0)$. Further discussion is provided in Sec. C.

In parallel, we optimize the reinforcement learning objective on the *context-conditioned policy*:

$$\mathcal{L}_{\text{RL}}(\theta) = -\mathbb{E}_{y\sim\pi_\theta(\cdot|x_1)}\big[\min(r_1\hat{A}, \text{clip}(r_1, 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}})\hat{A})\big], \text{ where } r_1 = \frac{\pi_\theta(y|x_1)}{\pi_{\text{old}}(y|x_1)}. \quad (11)$$

The final objective combines both terms:

$$\mathcal{L}_{\text{NUDGERL}} = \mathcal{L}_{\text{RL}} + \lambda_{\text{PC}}\mathcal{L}_{\text{PC}}. \quad (12)$$

This objective induces a complementary learning dynamic. The RL term operates on the context-conditioned policy, improving exploration and reinforcing successful trajectories within each context. In contrast, the policy correction term projects these improvements onto the base-prompt policy, enabling test-time generalization. As a result, the model learns to reproduce effective reasoning strategies without relying on explicit context at inference time. Further discussion on $\mathcal{L}_{\text{PC}}$ is in Sec. C.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Baselines. We compare our method against (i) the base model without optimization, (ii) GRPO with increasing rollout budgets, where $N \in \{8, 16, 32, 64\}$, which evaluates naive rollout scaling as a brute-force exploration strategy. Further details are provided in Sec. D.

Evaluation Datasets and Metrics. For mathematical reasoning, we evaluate on `AIME24` and `AIME25`, 30-problem olympiad-style high-school competitions (MAA, 2025); `AMC23`, a 40-problem

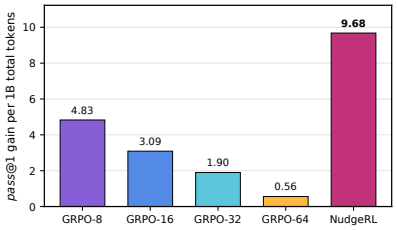


Figure 3: **Token efficiency.** We report average pass@1 gain from base model per 1B tokens across GRPO with increasing rollouts and NUDGERL.

Table 3: **Training compute comparison.** We report estimated training compute in EFLOPs (10^{18} FLOPs), broken down into rollout generation, RL optimization, and policy correction (PC).

Method	Rollouts (N)	Training Compute (EFLOPs)			
		Generation	RL	PC	Total
GRPO	8	0.084	0.258	0	0.342
	16	0.172	0.529	0	0.701
	32	0.349	1.072	0	1.421
	64	0.684	2.102	0	2.786
NUDGERL	8	0.068	0.213	0.211	0.492

high-school contest benchmark (MAA, 2023); the level-5 subset of `MATH500`, containing 134 difficult MATH problems (Hendrycks et al., 2021); and the `Apex Shortlist`, consisting of 48 advanced competition-style problems (Balunović et al., 2025). We report *pass*@1, estimated from 128 rollouts using the unbiased estimator of Chen et al. (2021). All solutions are automatically graded using `math-verify` (HuggingFace, 2024). Additional details are provided in Sec. F.

Models. We apply NUDGERL to `Qwen3-4B-Instruct-2507` (Team, 2025) and `Olmo-3-7B-Instruct-SFT` (Olmo et al., 2025) using `DAPO-17k-Processed` as a training set (Yu et al., 2025). To construct the pool of contexts, we used `gpt-4o-mini` (OpenAI, 2024) to generate two strategy-level contexts per problem (*e.g.*, Pythagorean theorem), and used them without additional verification (*i.e.*, $\forall x \in \mathcal{D}, |\mathcal{C}(x)| = 2$).

5.2 MAIN RESULTS

As shown in Tab. 1, NUDGERL matches the performance of the best GRPO configuration while requiring only 8 rollouts per prompt, demonstrating substantially better training efficiency than scaling rollout budgets alone. On `Qwen3-4B-Instruct-2507`, NUDGERL attains an average pass@1 of 0.489, matching or slightly exceeding the best GRPO configuration (0.487 at $N = 32$) while using only one-fourth of the rollout budget. It also outperforms GRPO with 64 rollouts (0.451), highlighting the instability of simply scaling the number of sampled rollouts as introduced in Hu et al. (2025). A similar trend is observed on `Olmo3-7B-Instruct-SFT`, where NUDGERL achieves the best average performance of 0.285, comparable to the strongest GRPO baseline (0.281 at $N = 32$). These results suggest that improving the quality of exploration through Strategy Nudging is substantially more compute-efficient than brute-force rollout scaling.

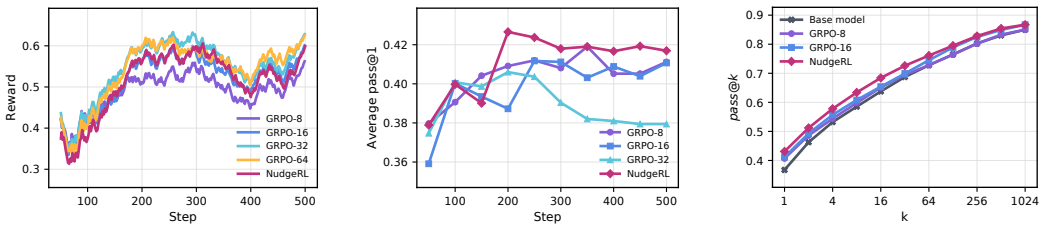
5.3 RESULTS ON CODE GENERATION

To demonstrate that NUDGERL generalizes beyond mathematical reasoning, we train `Qwen2.5-Coder-7B-Instruct` (Hui et al., 2024) on `LiveCodeBench v1-v5` (Jain et al., 2024) and evaluate on `v6`. As shown in Tab. 2, NUDGERL consistently outperforms GRPO across all rollout budgets while using only 8 rollouts, achieving the best overall *pass*@1 of 0.335 compared to the strongest GRPO baseline of 0.317 at 64 rollouts. The improvement is consistent across both AtCoder and LeetCode subsets, suggesting that the benefit of Strategy Nudging extends beyond mathematical reasoning to code generation, where improving exploration quality is again more effective than simply scaling rollout budgets.

5.4 ANALYSIS ON EFFICIENCY

To better understand the efficiency gains of NUDGERL, we analyze its behavior from three complementary perspectives: (i) token efficiency, (ii) training efficiency, and (iii) reasoning-mode coverage during training. All analyses are conducted using the training and evaluation data from `Qwen3-4B-Instruct-2507`.

Token Efficiency. As shown in Fig. 3, NUDGERL achieves performance comparable to the best-performing GRPO configuration ($N = 32$) approximately $5\times$ more efficiently, even after accounting



(a) Training reward (b) *pass*@1 (c) *pass*@k

Figure 4: **Training and inference-time behavior** on `AIME24/25`. (a) EMA-smoothed training reward (decay 0.99). (b) *pass@1* during training. (c) Average *pass@k*.

for the one-time offline context generation cost. Another notable observation is that the average *pass@1* improvement over the base model consistently decreases as the rollout budget increases in GRPO, which is consistent with our theoretical analysis in Sec. 3.2. This trend also aligns with the findings of Cheng et al. (2026), such that the benefit of increasing the number of rollouts eventually saturates under a fixed sampling budget, further highlighting the effectiveness of Strategy Nudging.

Training Efficiency. Since NUDGERL introduces an additional forward and backward pass through the policy correction objective, analyzing training FLOPs is equally important. As reported in Tab. 3, the additional policy correction term incurs approximately 44% extra computation relative to GRPO with the same rollout budget. Nevertheless, NUDGERL remains substantially more compute-efficient than GRPO with larger rollout budgets while achieving comparable or better performance.

While the previous analyses quantify the efficiency gains, they do not explain where these gains originate. To better understand the underlying mechanism, we analyze the training dynamics and investigate how Strategy Nudging improves reasoning-mode coverage during RL training.

As shown in Fig. 4b, NUDGERL improves *pass@1* substantially faster than GRPO variants and remains the strongest method throughout most of training. This suggests that Strategy Nudging improves sample efficiency by exposing useful reasoning trajectories earlier, rather than merely increasing the number of sampled rollouts. This trend also extends to inference-time sampling. As shown in Fig. 4c, NUDGERL consistently outperforms GRPO-8 across the full range of sampling budgets, indicating that it requires fewer generated solutions to achieve the same level of *pass@k*. Details of the token-efficiency analysis and the FLOPs estimation are provided in Sec. G.

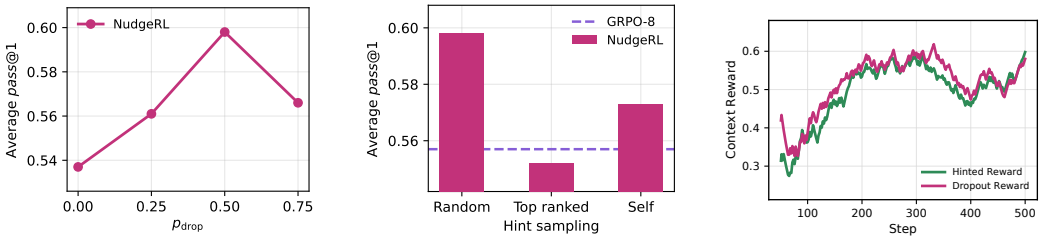
5.5 UNDERLYING MECHANISM OF NUDGERL

To further understand the source of performance gains in NUDGERL, we conduct a series of controlled experiments using `Qwen3-4B-Instruct-2507` on a subset of benchmarks.

p_{drop} Ablation. As shown in Fig. 5a, a moderate dropout rate ($p_{\text{drop}} = 0.5$) consistently yields the best performance across benchmarks. Context dropout plays a dual role: it enables exploration beyond fixed contexts by occasionally reverting to the base prompt, while also stabilizing group-wise statistics through a more balanced sample distribution. When $p_{\text{drop}} = 0$, exploration is restricted to predefined contexts, whereas large values diminish the influence of context forcing. These results suggest that maintaining a balanced mixture of context-conditioned and context-free samples is important for achieving both diverse exploration and stable optimization.

Hint Sampling. We also study how the quality of sampled contexts affects performance by comparing three strategies: *random* sampling, *top-ranked* selection, and *self*-generated contexts. In the top-ranked setting, we first generate a pool of five candidate contexts and select the two that yield the largest improvement in *pass*@16 for each problem. For self-generated contexts, we prompt the base model to produce its own strategy hints.

As shown in Fig. 5b, random sampling consistently achieves the best *pass*@1, outperforming both top-ranked selection and self-generated contexts. While top-ranked contexts ensure higher immediate correctness, utilizing them to *nudge* the training policy yields lower performance than the GRPO



(a) Ablation on p_{drop} (b) Ablation on sampling (c) Context reward

Figure 5: **Mechanism analyses.** (a, b) Average *pass*@1 on `AIME24/25, AMC23, MATH500`. (c) Time-weighted EMA reward mean (0.99) with and without context.

baseline. Self-generated contexts improve over the GRPO baseline, but remain less effective than randomly sampled external contexts, likely because they inherit the policy's existing exploration bias.

Effect of Contexts during training. We also report the dropout reward mean ($\mathbb{E}_i[r(x_0, y^{(i)}) \mid z^{(i)} = \emptyset]$) and the hinted reward mean ($\mathbb{E}_i[r(x_0, y^{(i)}) \mid z^{(i)} \neq \emptyset]$) during training with NUDGERL. As shown in Fig. 5c, both rewards improve together throughout training, suggesting that trajectories discovered under context-conditioned exploration are successfully transferred to the base policy through the policy correction objective. Interestingly, the dropout reward occasionally exceeds the hinted reward during training. This contrasts with prior feasibility-oriented methods based on privileged information (Qu et al., 2026; Chen et al., 2025).

These results suggest that, within our framework, the primary role of context is not to provide the single best hint, but to promote diversity in reasoning. The complete results of the evaluation are given in Sec. I and results of additional experiments including comparison to oracle prefix baseline, comparison with DAPO-style decoupled clipping (Yu et al., 2025) are reported in Sec. J.

6 CONCLUSION

In this work, we introduced NUDGERL, a framework for structured exploration in RLVR. Our approach leverages *Strategy Nudging* to induce diverse reasoning trajectories by sampling from lightweight, strategy-level context-conditioned distributions, and learns from them via policy correction augmented RL objective. Empirically, NUDGERL achieves performance comparable to that of the best GRPO configuration using $5\times$ fewer tokens and $2.9\times$ less training compute on mathematical reasoning tasks and even surpasses the best GRPO baseline, which uses an $8\times$ larger rollout budget, on the code-generation task.

Limitations & Future Work. A practical consideration of NUDGERL is the cost of generating strategy-level contexts. However, this is an *offline* process performed once prior to training, using a lightweight LLM (e.g., `gpt-4o-mini`), and the resulting contexts can be reused across training runs without additional overhead. A more fundamental limitation lies in how contexts are generated independently of the model being trained. The benefit of Strategy Nudging stems from inducing trajectories that are unlikely under the current policy. As training progresses, however, a fixed context pool may become less informative as the policy adapts. A promising direction for future work is *model-adaptive context generation*, which dynamically constructs contexts tailored to the current policy's blind spots, potentially yielding more consistent exploration gains throughout training.

AI USE STATEMENT

We used generative AI tools to assist with software implementation and code editing during the development of this work. All AI-assisted code was reviewed and tested by the authors.

Generative models also form part of the experimental methodology of this work. Specifically, we used external language models to generate *context nudges* for the training of NUDGERL. These uses, including the corresponding prompts, data-generation procedures, and experimental settings, are described in the main paper and appendix.

REFERENCES

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions, February 2025. URL https://matharena.ai/.

Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, et al. Nvidia nemotron 3: Efficient and open intelligence. *arXiv preprint arXiv:2512.20856*, 2025.

Justin Chih-Yao Chen, Becky Xiangyu Peng, Prafulla Kumar Choubey, Kung-Hsiang Huang, Jiaxin Zhang, Mohit Bansal, and Chien-Sheng Wu. Nudging the boundaries of llm reasoning. *arXiv preprint arXiv:2509.25666*, 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Zhoujun Cheng, Yutao Xie, Yuxiao Qu, Amrith Setlur, Shibo Hao, Varad Pimpalkhute, Tongtong Liang, Feng Yao, Zhengzhong Liu, Eric Xing, et al. Isocompute playbook: Optimally scaling sampling compute for llm rl. *arXiv preprint arXiv:2603.12151*, 2026.

Jia Deng, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, and Ji-Rong Wen. Decomposing the entropy-performance exchange: The missing keys to unlocking effective reinforcement learning. *arXiv preprint arXiv:2508.02260*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Jian Hu, Mingjie Liu, Ximing Lu, Fang Wu, Zaid Harchaoui, Shizhe Diao, Yejin Choi, Pavlo Molchanov, Jun Yang, Jan Kautz, et al. Brorl: Scaling reinforcement learning via broadened exploration. *arXiv preprint arXiv:2510.01180*, 2025.

HuggingFace. Math-verify: A toolkit for verifying mathematical reasoning. https://github.com/huggingface/Math-Verify, 2024. Accessed 2026-05-06.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Woosuk Kwon. *vLLM: An Efficient Inference Engine for Large Language Models*. PhD thesis, UC Berkeley, 2025.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Baohao Liao, Hanze Dong, Xinxing Xu, Christof Monz, and Jiang Bian. Self-hinting language models enhance reinforcement learning. *arXiv preprint arXiv:2602.03143*, 2026.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025a.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. *arXiv preprint arXiv:2505.24864*, 2025b.

MAA. American mathematics competitions. https://www.maa.org/math-competitions, 2023.

MAA. Aime: American invitational mathematics examination. https://www.maa.org/math-competitions, 2025.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3, 2025. URL https://arxiv.org/abs/2512.13961.

OpenAI. Gpt-4o mini. https://openai.com/ko-KR/index/gpt-4o-mini-advancing-cost-efficient-intelligence/, 2024. Accessed: 2026-05-04.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Yuxiao Qu, Amrith Setlur, Virginia Smith, Ruslan Salakhutdinov, and Aviral Kumar. Pope: Learning to reason on hard problems via privileged on-policy exploration. *arXiv preprint arXiv:2601.18779*, 2026.

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

Amrith Setlur, Zijian Wang, Andrew Cohen, Paria Rashidinejad, and Sang Michael Xie. Reuse your flops: Scaling rl on hard problems by conditioning on very off-policy prefixes, 2026. URL https://arxiv.org/abs/2601.18795.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv*, 2402.03300, 2024. URL https://doi.org/10.48550/arXiv.2402.03300.

Yuda Song, Lili Chen, Fahim Tajwar, Remi Munos, Deepak Pathak, J Andrew Bagnell, Aarti Singh, and Andrea Zanette. Expanding the capabilities of reinforcement learning via text feedback. *arXiv preprint arXiv:2602.02482*, 2026.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformers Reinforcement Learning, 2020. URL https://github.com/huggingface/trl.

Christian Walder and Deep Karkhanis. Pass@k policy optimization: Solving harder reinforcement learning problems. *CoRR*, abs/2505.15201, 2025. doi: 10.48550/ARXIV.2505.15201. URL https://doi.org/10.48550/arXiv.2505.15201.

Fang Wu, Weihao Xuan, Ximing Lu, Zaïd Harchaoui, and Yejin Choi. The invisible leash: Why RLVR may not escape its origin. *arXiv*, 2507.14843, 2025. URL https://doi.org/10.48550/arXiv.2507.14843.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Weinan Dai, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: an open-source LLM reinforcement learning system at scale. *arXiv*, 2503.14476, 2025. URL https://doi.org/10.48550/arXiv.2503.14476.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv*, 2504.13837, 2025. URL https://doi.org/10.48550/arXiv.2504.13837.

Xiaoyun Zhang, Xiaojian Yuan, Di Huang, Wang You, Chen Hu, Jingqing Ruan, Kejiang Chen, and Xing Hu. Rediscovering entropy regularization: Adaptive coefficient unlocks its potential for llm reinforcement learning. *arXiv preprint arXiv:2510.10959*, 2025a.

Xuechen Zhang, Zijian Huang, Yingcong Li, Chenshun Ni, Jiasi Chen, and Samet Oymak. Bread: Branched rollouts from expert anchors bridge sft & rl for reasoning. *arXiv preprint arXiv:2506.17211*, 2025b.

A Details on Strategy Nudging

Strategy Generating Prompt. We use `gpt-4o-mini` to generate keyword-level hints for each problem. For the main experiments, we generate two hints per problem, while in the top-ranked setting, we first generate five candidate hints and select a subset based on oracle evaluation.

The exact prompt used for hint generation is as follows:

```
f"""Given the following math problem, generate {num_hints} different
    keyword hints that would help solve it.
Each hint should be a specific mathematical concept, theorem, or
    technique (e.g., "Ceva's theorem", "Lifting the exponents",
    "Triangle inequality").

Problem:
{problem}

Please provide exactly {num_hints} hints in the following format (one
    hint per line, numbered):
1. [Hint 1]
2. [Hint 2]
...
{num_hints}. [Hint {num_hints}]

Make sure each hint is a distinct mathematical concept or theorem."""
```

Strategy Nudging prompt. Given a problem and an optional hint, we construct prompts that encourage the model to follow a specific reasoning strategy. The model is instructed to explicitly separate its reasoning process and final answer using predefined delimiters.

```
   reasoning_start = "<start_working_out>"
   reasoning_end = "<end_working_out>"
   solution_start = "<SOLUTION>"
   solution_end = "</SOLUTION>"

   system_prompt = f"""You are given a problem.
   Think about the problem and provide your working out.
   Place it between {reasoning_start} and {reasoning_end}.
   Then, provide your solution between {solution_start}{solution_end}"""

   def build_messages(problem: str, system_prompt: str, hint: str | None
       = None) -> list[dict[str, str]]:
      context_block = ""
      if hint:
         context_block = (
            "Context (exploration condition):\n"
            f"- Use this hint/approach: {hint}\n\n"
            "Important:\n"
            "- Follow this approach as your primary strategy.\n\n"
         )
      user_content = (
         "Problem:\n"
         f"{problem}\n\n"
         f"{context_block}"
         "Solve this step by step and provide your final numerical
             answer at the end."
      )
      return [
         {"role": "system", "content": system_prompt},
         {"role": "user", "content": user_content},
      ]
```

Effect of Strategy Nudging. To evaluate the effect of Strategy Nudging, we sample 8 rollouts from `Qwen3-4B-Instruct-2507` (Team, 2025) on 200 problems from `DAPO-17k-Processed` (Yu et al., 2025), both with and without Strategy Nudging, and analyze the resulting rollout diversity via LLM-as-a-judge.

LLM-as-a-judge prompt. To analyze the diversity of generated rollouts, we use `gpt-4o-mini` as an LLM-as-a-judge to cluster solutions based on their underlying reasoning strategies and count the number of distinct solution modes. Given a problem and a set of rollouts, the model is instructed to identify the number of *conceptually distinct* solution approaches, while ignoring superficial differences such as phrasing or minor computational variations.

```
prompt = (f"Problem:\n{problem_text}\n\n"
        f"Here are {len(rollouts)} student solutions to this
            problem:\n"
        f"{formatted_rollouts}\n"
        f"Analyze these solutions and determine how many *conceptually
            distinct* solution strategies are used across them.\n"
        f"Ignore minor calculation differences or phrasing variations.
            Focus on the core mathematical approach.\n"
        f"Provide the answer in the following format: 'Distinct
            Strategies: X' where X is the integer count.\n"
        f"Then briefly list the distinct strategies identified.")
messages = [
    {"role": "system", "content": "You are an expert math teacher
        evaluating the diversity of student solution methods."},
    {"role": "user", "content": prompt}
]
```

B Further Analysis of λ

In this section, we provide a detailed interpretation of the proposed advantage estimator, prove proposition 4.1, and extend the reward-ordering analysis beyond binary rewards.

For notational convenience, we define the effective group baseline as

$$b_z = \begin{cases} \bar{r}_z, & z \neq \emptyset, \\ \bar{r}, & z = \emptyset. \end{cases} \tag{13}$$

The unnormalized advantage of a rollout y_i with reward r_i and assigned context $z^{(i)}$ can then be written uniformly as

$$\begin{aligned} A_i &= (r_i - b_{z^{(i)}}) + \lambda\,(b_{z^{(i)}} - \bar{r}) \\ &= r_i + (\lambda - 1) b_{z^{(i)}} - \lambda \bar{r}. \end{aligned} \tag{14}$$

When $z^{(i)} = \emptyset$, we have $b_{z^{(i)}} = \bar{r}$, and Eq. 14 reduces to $A_i = r_i - \bar{r}$, consistent with the definition in the main text.

Interpretation as a hierarchical baseline. Under Strategy Nudging, rollouts are sampled from a mixture of context-conditioned distributions. The global mean $\bar{r}$ can therefore be interpreted as an empirical baseline for the overall mixture, whereas $\bar{r}_z$ serves as a component-specific baseline for rollouts generated under context z.

The intra-context term

$$r_i - b_{z^{(i)}} \tag{15}$$

measures the relative quality of a rollout within its assigned context, thereby removing between-context reward variation from the within-context comparison. The context-level term

$$\lambda\,(b_{z^{(i)}} - \bar{r}) \tag{16}$$

then reintroduces a controllable preference based on the relative reward of the corresponding context group.

At $\lambda = 1$, the two terms telescope:

$$A_i = (r_i - b_{z^{(i)}}) + (b_{z^{(i)}} - \bar{r}) = r_i - \bar{r}. \tag{17}$$

Thus, $\lambda = 1$ exactly recovers the standard GRPO estimator. Values of λ different from one control how strongly context-level reward differences affect credit assignment.

B.1 Proof of proposition 4.1

Consider two trajectories y and y' with rewards r and r', sampled from context groups z and z', respectively. From Eq. 14, their unnormalized advantage difference is

$$\begin{aligned} A - A' &= [r + (\lambda - 1)b_z - \lambda\bar{r}] - [r' + (\lambda - 1)b_{z'} - \lambda\bar{r}] \\ &= (r - r') + (\lambda - 1)(b_z - b_{z'}). \end{aligned} \tag{18}$$

The global baseline $\bar{r}$ cancels because both trajectories belong to the same optimization batch.

Advantage normalization preserves pairwise ordering, since

$$\widehat{A} - \widehat{A}' = \frac{A - A'}{\sigma_A + \delta}, \tag{19}$$

with $\sigma_A + \delta > 0$. It therefore suffices to establish the ordering of the unnormalized advantages.

For binary rewards $r, r' \in \{0, 1\}$, suppose $r > r'$. Then $r - r' = 1$, and since each group baseline is an average of binary rewards,

$$|b_z - b_{z'}| \leq 1. \tag{20}$$

Hence,

$$\begin{aligned} A - A' &= 1 + (\lambda - 1)(b_z - b_{z'}) \\ &\geq 1 - |\lambda - 1|. \end{aligned} \tag{21}$$

For every $\lambda \in (0, 2)$, we have $|\lambda - 1| < 1$, and thus

$$A - A' > 0. \tag{22}$$

It remains only to consider $\lambda = 2$. Because y is a successful trajectory contained in group z, its group mean satisfies $b_z > 0$. Likewise, because y' is a failed trajectory contained in group z', we have $b_{z'} < 1$. Therefore,

$$b_z - b_{z'} > -1, \tag{23}$$

and consequently

$$A - A' = 1 + b_z - b_{z'} > 0. \tag{24}$$

Thus,

$$r > r' \quad \Longrightarrow \quad A > A' \qquad \text{for all } \lambda \in (0, 2]. \tag{25}$$

Since normalization is a positive affine transformation, the same ordering holds for $\widehat{A}$ and $\widehat{A}'$. □

Ordering among equal-reward trajectories. When $r = r'$, Eq. 18 becomes

$$A - A' = (\lambda - 1)(b_z - b_{z'}). \tag{26}$$

Thus, λ affects only the relative ordering of trajectories with equal reward. For $\lambda < 1$, equal-reward trajectories from lower-reward context groups receive larger advantages, favoring behaviors discovered under less reliable contexts. For $\lambda > 1$, the ordering is reversed, assigning greater credit to trajectories from contexts that more consistently induce successful behavior. At $\lambda = 1$, the context-dependent term vanishes entirely.

This provides a direct interpretation of λ as controlling the optimization bias among equally rewarded trajectories. Strategy Nudging diversifies behavior at the sampling stage through context conditioning, while $\lambda > 1$ can selectively bias optimization toward trajectories induced by more reliable contexts.

B.2 Extension to General Reward Functions

The same argument extends directly beyond binary rewards. Define the smallest positive reward gap among sampled trajectories as

$$\Delta_{\min} = \min_{r_i > r_j} (r_i - r_j), \tag{27}$$

and the largest difference between effective group baselines as

$$D_{\max} = \max_{z, z' \in \mathcal{G}(x_0)} |b_z - b_{z'}|. \tag{28}$$

We consider $D_{\max} > 0$; when $D_{\max} = 0$, the context-dependent term is identical across groups and cannot alter reward ordering for any λ.

For any pair with $r > r'$,

$$\begin{aligned} A - A' &= (r - r') + (\lambda - 1)(b_z - b_{z'}) \\ &\geq (r - r') - |\lambda - 1|\,|b_z - b_{z'}| \\ &\geq \Delta_{\min} - |\lambda - 1| D_{\max}. \end{aligned} \tag{29}$$

Therefore, the sufficient condition

$$|\lambda - 1| < \frac{\Delta_{\min}}{D_{\max}} \tag{30}$$

guarantees

$$r > r' \quad \Longrightarrow \quad A > A'. \tag{31}$$

The same ordering holds after advantage normalization.

More generally,

$$|\lambda - 1| \leq \frac{\Delta_{\min}}{D_{\max}} \tag{32}$$

guarantees weak order preservation,

$$r > r' \quad \Longrightarrow \quad A \geq A', \tag{33}$$

although ties may occur at the boundary.

C Discussion on Policy Correction

The policy correction term in Eq. 10 is adapted from the advantage-weighted regression (AWR) (Peng et al., 2019)-style self-distillation objective of RLTF-SD (Song et al., 2026). RLTF-SD uses trajectories sampled from a feedback-conditioned policy to train the corresponding context-free policy. While importance sampling can correct this distribution mismatch and recover an unbiased objective, sequence-level importance ratios incur high variance as likelihood ratios compound across tokens. RLTF-SD therefore omits importance correction and adopts AWR-style distillation, trading mild bias for a substantially lower-variance learning signal. Our setting has the same structure: strategy conditioning serves as an exploration mechanism, and the resulting trajectories are distilled back into the base policy. We therefore adopt the same no-importance-sampling formulation for policy correction.

D Details on Baselines

Rollout Scaling in GRPO. For controlled experiments, we scale the number of rollouts per prompt while adjusting the gradient accumulation steps and generation batch size accordingly, as summarized in Tab. 4. This ensures that the total optimization dynamics remain comparable across different rollout settings.

Example of Generated Contexts. We provide an illustrative example of the strategy-level contexts used in our method. These contexts are lightweight, keyword-level hints that do not directly solve the problem, but instead steer the model toward distinct reasoning modes. Importantly, they are not intended to provide intermediate steps or solutions, but rather to act as high-level inductive biases that diversify exploration.

Problem

Let S be the set of triples (a, b, c) of non-negative integers such that $a + b + c$ is even. The value of the sum

$$\sum_{(a,b,c)\in S} \frac{1}{2^a 3^b 5^c}$$

can be expressed as $\frac{m}{n}$, where m and n are relatively prime positive integers. Compute $m + n$.

Strategy-level contexts:

"Generating functions", "Parity of sums"

E Training Details

Framework. We used TRL (von Werra et al., 2020) for implementing baselines and our algorithm.

Table 4: Hyperparameters for training.

Parameter	Value
LoRA(Hu et al., 2022) rank	32
Max prompt length	2,048
Max completion length	6,144
RL steps	500
Batch size	4
Rollouts per prompt	N
Gradient accumulation steps	$\frac{N}{8}$
Generation batch size	N
Temperature	1
Min-p	0.0
Top-p	0.95
Top-k	-1
Learning rate	2×10^{-5}
LR scheduler	cosine
Weight decay	0.001
Warmup ratio	0.05
Optimizer	AdamW (8-bit)
KL coefficient β	0
Epsilon low	0.2
Epsilon high	0.2 (unless specified)
λ	1.1
λ_{PC}	0.1
p_{drop}	0.5
Random seed	42

Hyperparameters. The hyperparameters we used in training are given in Tab. 4. λ is set to 1.1 for exploitation among strategies following the analysis in Sec. B, proposition 4.1 and λ_{PC} is set to 0.1 following form Song et al. (2026) without additional hyperparameter optimization. Other non method-specific hyperparameters are shared through NudgeRL and other baselines.

Compute resources. For all experiments, we used NVIDIA H200 140GB GPUs.

F Details on Evaluation

During evaluation using vLLM (Kwon, 2025), all hyperparameters are kept identical to Tab. 4, except for the temperature, which is set to 0.7.

G Details on Efficiency Analysis

This appendix provides the detailed numerical results and computation protocol for the efficiency analysis presented in the main paper. We analyze both *token efficiency* and *training efficiency* under a unified accounting scheme to enable fair comparison across different exploration methods.

G.1 Token Efficiency

The complete numerical results are reported in Tab. 5.

Table 5: **Token-efficiency comparison on `Qwen3-4B-Instruct-2507`.** We report the average *pass*@1 improvement over the base model, the number of training tokens, and the resulting improvement per billion total tokens. For NudgeRL, total tokens include the one-time offline context-generation cost. Best results are represented as **bold**.

Metric	GRPO-8	GRPO-16	GRPO-32	GRPO-64	NudgeRL
Average *pass*@1 gain	0.052	0.068	0.085	0.049	**0.087**
Completion tokens	10,455,528	21,497,141	43,590,745	85,450,770	8,523,721
Training tokens	10,762,224	22,034,469	44,665,401	87,600,082	8,860,902
Offline context-generation tokens	0	0	0	0	125,000
Total tokens	10,762,224	22,034,469	44,665,401	87,600,082	8,985,902
Average *pass*@1 gain per B total tokens	4.83	3.09	1.90	0.56	**9.68**

For NudgeRL, we separately account for the one-time offline cost of context generation. Unlike rollout generation during RL training, these contexts are produced only once for each training dataset and can subsequently be reused across different policy initializations, hyperparameter configurations, or repeated training runs. Consequently, this cost is amortized over all subsequent experiments and does not scale with the number of RL optimization steps. To provide a conservative comparison, however, all results reported in the main paper include this offline context-generation cost in the total token budget.

G.2 Training Efficiency

To compare computational efficiency across methods, we estimate the total training FLOPs using a simplified dense Transformer cost model. Following standard approximations, we assume a dense 4B-parameter Transformer ($P = 4 \times 10^9$), where

- one forward pass over T tokens requires approximately $2PT$ FLOPs,
- one forward–backward optimization step requires approximately $6PT$ FLOPs,
- autoregressive generation with KV caching requires approximately $2PC$ FLOPs,

where T denotes the number of tokens participating in optimization (prompt + completion), and C denotes the generated completion length.

Under this approximation, the per-step computational cost becomes

$$\text{GRPO} = 2PC + 6PT, \tag{34}$$

$$\text{NudgeRL} = 2PC + 6PT_{x_1} + 6PT_{x_0}, \tag{35}$$

where T_{x_0} and T_{x_1} denote the optimization tokens for the policy correction objective and the standard RL objective, respectively.

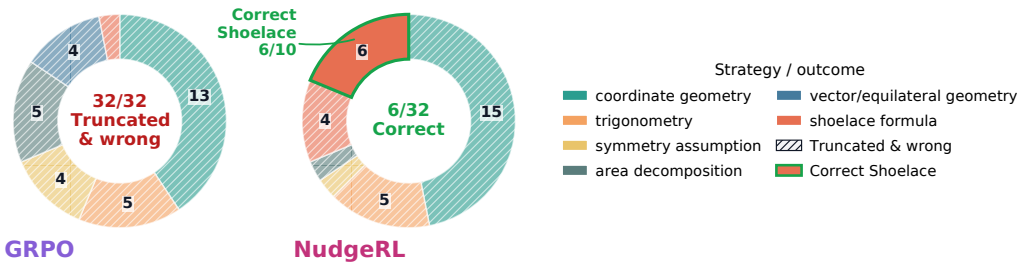


Figure 6: **NUDGERL internalizes effective test-time strategies.** Across 32 rollouts on a AIME25 problem, GRPO yields only incorrect and truncated trajectories. Conversely, NUDGERL produces 6 correct solutions using the shoelace formula.

For every training configuration, we record the average prompt length, average completion length, rollout budget, and the total number of optimization steps. These statistics are then aggregated to estimate the total training compute reported in Tab. 3. Note that our estimates intentionally exclude infrastructure-dependent overheads such as optimizer state updates, communication costs, activation checkpointing, and data loading, as these are shared across methods and do not materially affect the relative comparison. Accordingly, the reported EFLOPs should be interpreted as normalized estimates of model computation rather than exact hardware-level measurements.

H CASE STUDY

To examine the source of performance gains in NUDGERL, we analyze one AIME25 problem where the NUDGERL-trained model successfully sampled correct trajectories, while the GRPO-trained model entirely failed. We sampled 32 rollouts and categorized their dominant reasoning strategies. As shown in Fig. 6, both models predominantly relied on *coordinate geometry*. However, the GRPO-trained model additionally explored ineffective strategies such as *symmetry assumptions* and *area decomposition*, which consistently resulted in truncated solutions, causing all 32 trajectories to fail. While GRPO sampled the *shoelace formula* strategy only once, NUDGERL substantially increased its frequency and successfully exploited it to generate correct trajectories.

This behavior highlights the complementary roles of our framework: *Strategy Nudging* exposes rare but effective reasoning modes such as the shoelace-formula strategy, while the *Inter-Intra Group Advantage* reinforces and exploits such reliable strategies once discovered.

The GRPO-trained model predominantly relied on coordinate geometry combined with heuristic symmetry assumptions and case-by-case area decomposition. Although these approaches occasionally progressed toward partial solutions, they frequently resulted in excessively long derivations and truncated outputs before reaching the final answer.

In contrast, NUDGERL exploited the *shoelace-formula* strategy, which directly computes polygon areas from vertex coordinates. This strategy produced substantially shorter and more reliable reasoning trajectories, enabling successful completion within the generation budget.

Problem

Let $\triangle ABC$ be a right triangle with $\angle A = 90^\circ$ and $BC = 38$. There exist points K and L inside the triangle such that

$$AK = AL = BK = CL = KL = 14.$$

The area of the quadrilateral $BKLC$ can be expressed as $n\sqrt{3}$ for some positive integer n. Find n.

I FULL EVALUATION RESULTS

Full evaluation results on p_{drop} is given in Tab. 6 and results for different hint sampling algorithms are in Tab. 7.

Table 6: p_{drop} **ablation results.** We report *pass*@1 estimated using 128 rollouts. Best results are represented as **bold**.

p_{drop}	AIME24	AIME25	AMC23	MATH500	Average
0.00	0.418	0.344	0.759	0.628	0.537
0.25	0.461	0.354	0.773	0.658	0.561
0.50	**0.482**	**0.393**	**0.857**	**0.660**	**0.598**
0.75	0.458	0.361	0.796	0.649	0.566

Table 7: **Hint sampling ablation results.** We report *pass*@1 estimated using 128 rollouts. Best results are represented as **bold**.

Sampling	AIME24	AIME25	AMC23	MATH500	Average
GRPO(8 rollouts, baseline)	0.444	0.367	0.749	0.668	0.557
Random	**0.482**	**0.393**	**0.857**	**0.660**	**0.598**
Top ranked	0.448	0.355	0.774	0.632	0.552
Self	0.436	0.373	0.844	0.639	0.573

J Additional Experiments

J.1 Comparison with Oracle Prefix Baseline

To compare NudgeRL with methods that leverage privileged information, we implement an oracle prefix baseline following POPE (Qu et al., 2026). Specifically, we adopt the prompt format and dataset mixture (i.e., with and without oracle solutions) from the original work. Since the adaptive prefix-length selection strategy used in POPE is not publicly available, we instead evaluate a range of fixed prefix ratios. Oracle solutions are generated using `DeepSeek-R1-0528` (Liu et al., 2025a), and an example oracle solution for the problem presented in Sec. D is shown below.

Oracle solution:

We compute the sum

$$\sum_{(a,b,c)\in S} \frac{1}{2^a 3^b 5^c},$$

where S consists of non-negative integer triples (a, b, c) with $a + b + c$ even. Using the indicator

$$\mathbf{1}_{a+b+c \text{ even}} = \frac{1}{2}\left(1 + (-1)^{a+b+c}\right),$$

we write

$$\sum_{(a,b,c)\in S} \frac{1}{2^a 3^b 5^c} = \frac{1}{2}\left(\sum_{a,b,c\geq 0} \frac{1}{2^a 3^b 5^c} + \sum_{a,b,c\geq 0} \frac{(-1)^{a+b+c}}{2^a 3^b 5^c}\right).$$

The first sum factors as

$$\sum_{a\geq 0} \frac{1}{2^a} \cdot \sum_{b\geq 0} \frac{1}{3^b} \cdot \sum_{c\geq 0} \frac{1}{5^c} = \frac{1}{1-\frac{1}{2}} \cdot \frac{1}{1-\frac{1}{3}} \cdot \frac{1}{1-\frac{1}{5}} = 2 \cdot \frac{3}{2} \cdot \frac{5}{4} = \frac{15}{4}.$$

The second sum factors as

$$\sum_{a\geq 0} \frac{(-1)^a}{2^a} \cdot \sum_{b\geq 0} \frac{(-1)^b}{3^b} \cdot \sum_{c\geq 0} \frac{(-1)^c}{5^c} = \frac{1}{1+\frac{1}{2}} \cdot \frac{1}{1+\frac{1}{3}} \cdot \frac{1}{1+\frac{1}{5}} = \frac{2}{3} \cdot \frac{3}{4} \cdot \frac{5}{6} = \frac{5}{12}.$$

Thus, the total sum is

$$\frac{1}{2}\left(\frac{15}{4} + \frac{5}{12}\right) = \frac{1}{2} \cdot \frac{50}{12} = \frac{25}{12}.$$

Since $\frac{25}{12}$ is in lowest terms, we have $m = 25$, $n = 12$, and $m + n = 37$.

We train `Qwen3-4B-Instruct-2507` using prefix ratios $\{0.05, 0.10, 0.15, 0.20, 0.25\}$. The results are summarized in Tab. 8.

Table 8: **Comparison with oracle prefix baseline.** We report *pass*@1 estimated using 128 rollouts. Best results are represented as **bold**.

Method	Prefix Ratio	AIME24	AIME25	Average
Oracle Prefix	0.05	0.404	0.342	0.373
	0.10	0.405	**0.369**	0.387
	0.15	**0.460**	0.337	0.399
	0.20	0.439	0.364	**0.401**
	0.25	0.430	0.368	0.399
Best Oracle Prefix	–	**0.460**	**0.369**	**0.415**
NudgeRL	–	**0.482**	**0.393**	**0.438**

To provide an optimistic upper bound, we additionally report *Best Oracle Prefix*, which selects the best-performing prefix ratio independently for each benchmark. Even under this oracle selection protocol, the average performance reaches only 0.415, remaining below NUDGERL (0.438). Furthermore, NUDGERL consistently outperforms the best oracle-prefix configuration on both AIME24 (0.482 vs. 0.460) and AIME25 (0.393 vs. 0.369).

These results indicate that the advantage of NUDGERL cannot be explained by a particular choice of oracle prefix length. Rather than exposing a partial ground-truth solution, encouraging exploration through diverse reasoning contexts proves more effective for improving downstream policy optimization.

Interestingly, NUDGERL also does not require the context generator itself to solve the target problem. In our experiments, the context generator (`gpt-4o-mini`) achieves *pass*@16 = 0 on approximately 33% of the 200 `DAPO-17k-Processed` problems. Nevertheless, the resulting contexts still substantially improve downstream exploration and ultimately outperform oracle-prefix training. This observation suggests that the role of contextual hints is fundamentally different from that of oracle solutions: they need not encode the correct reasoning process, but only provide sufficiently diverse and plausible strategic directions that guide the policy toward exploring reasoning trajectories it would otherwise fail to discover.

J.2 COMPARISON WITH ϵ_{HIGH} SCALING.

We further compare our algorithm with decoupled clipping (Yu et al., 2025): $\text{clip}(r, 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}})$, where ϵ_{high} controls the strength of policy updates by amplifying the contribution of successful trajectories. Increasing ϵ_{high} therefore allows more aggressive policy updates toward positive-advantage trajectories. As shown in Tab. 9, increasing ϵ_{high} generally improves GRPO performance in the moderate regime used in prior works (Shao et al., 2024; Yu et al., 2025). However, our method with $\epsilon_{\text{high}} = 0.2$ consistently outperforms GRPO across the entire scaling range from moderate to extreme values. This suggests that improving exploration quality is more effective than simply increasing the magnitude of stochastic policy updates. Additionally, under the more extreme scaling adopted in recent RLVR settings (Liu et al., 2025b), GRPO sharply deteriorates at $\epsilon_{\text{high}} = 0.4$. We argue that this degradation highlights a limitation of purely stochastic distribution-level exploration: increasing update magnitude alone provides little control over *what* is explored.

J.3 ABLATION RESULTS

Finally, to further understand the source of performance gains in NUDGERL, we conduct a series of controlled experiments using `Qwen3-4B-Instruct-2507` on a subset of benchmarks.

Exploration-Exploitation trade-off via λ. Tab. 10 presents the effect of varying λ, where $\lambda = 1.1$ achieves the best performance. This trend aligns with our Proposition 4.1 in Sec. 4.2. Since strategy nudging already ensures sufficient diversity at the sampling stage, increasing λ does not hinder exploration across contexts. Instead, it strengthens exploitation within each problem by prioritizing

Table 9: ϵ_{high} scaling results. We report *pass*@1 estimated using 128 rollouts. Best results are represented as **bold**.

Algorithm	ϵ_{high}	AIME24	AIME25	AMC23	MATH500	Average
NudgeRL	0.2	**0.482**	**0.393**	**0.857**	**0.660**	**0.598**
GRPO	0.2	0.444	0.367	0.749	**0.668**	0.557
	0.24	0.451	**0.373**	0.795	0.645	**0.566**
	0.28	0.443	0.372	0.793	0.648	0.564
	0.32	**0.452**	0.358	0.813	0.640	**0.566**
	0.36	0.432	0.338	**0.845**	0.647	0.565
	0.40	0.406	0.341	0.781	0.638	0.541

trajectories from more reliable contexts. This leads to more consistent learning of high-quality solutions per instance, explaining the observed performance gains at $\lambda = 1.1$.

Table 10: λ **ablation results.** We report *pass*@1 estimated using 128 rollouts. Best results are represented as **bold**.

λ	AIME24	AIME25	AMC23	MATH500	Average
0.9	0.403	0.366	0.806	0.648	0.556
1.0	0.436	0.359	0.831	0.643	0.567
1.1	**0.482**	**0.393**	**0.857**	**0.660**	**0.598**

Policy Correction Coefficient. As shown in Tab. 11, removing the policy correction term ($\lambda_{\text{PC}} = 0$) results in a clear performance drop, indicating that explicitly transferring context-discovered trajectories to the base policy is essential. However, overly large values also degrade performance, likely due to over-constraining the policy toward sampled trajectories. A moderate coefficient ($\lambda_{\text{PC}} = 0.1$) achieves the best results, suggesting that policy correction should complement the underlying RL objective.

Table 11: λ_{PC} **ablation results.** We report *pass*@1 estimated using 128 rollouts. Best results are represented as **bold**.

λ_{PC}	AIME24	AIME25	AMC23	MATH500	Average
0.0	0.423	0.362	0.826	0.628	0.560
0.1	**0.482**	**0.393**	**0.857**	**0.660**	**0.598**
0.5	0.425	0.361	0.730	0.629	0.536

K Broader Impacts

This paper proposes an efficient framework for structured exploration in reinforcement learning with verifiable rewards (RLVR). On the positive side, our method improves exploration efficiency without relying on extremely large rollout budgets or expensive oracle supervision, which may help reduce the computational cost of training reasoning models and improve accessibility for smaller research groups.

However, improving exploration efficiency may also contribute to the development of increasingly capable reasoning systems, which could be misused in harmful or unintended ways. We therefore emphasize the importance of continued research on safety, oversight, and responsible deployment.