sGPO: Trading Inference FLOPs for Training Efficiency in RLVR

Anonymous ACL submission

Abstract

Standard Reinforcement Learning with Verifiable Rewards (RLVR) training allocates a fixed rollout budget to every query, without regard for what each query's difficulty means for the current policy. This leads to two symmetric failure modes: easy queries produce near-zero advantage because the policy already solves them, while unsolvable queries produce no signal because the policy never solves them. Both regimes waste training FLOPs without contributing to a learning gradient. We introduce *sorted Group Policy Optimization* (sGPO), a compute-efficient strategy that trades a small budget of inference FLOPs for a large reduction in wasted training FLOPs. The key insight is that cheap inference compute can serve as a single offline proxy for query difficulty. By generating a small batch of parallel samples per query under the initial policy, we obtain a model-aware empirical success rate. This motivates setting the training rollout group size to the inverse of this success rate, a practical rule that maximizes sample efficiency by extracting the most advantage per generated rollout. This single profiling pass simultaneously drives data filtering (removing trivial queries and sub-sampling unsolvable ones), adaptive group size allocation, and curriculum construction (scheduling queries from easy to hard). sGPO matches or exceeds baseline performance while reducing total training compute by 2.5–3.1×—with the upfront inference profiling cost included.

1 Introduction

Reinforcement Learning has emerged as a crucial paradigm for enhancing the reasoning capabilities and alignment of Large Language Models (Guo et al., 2025; Jaech et al., 2024). Specifically, Reinforcement Learning with Verifiable Rewards (RLVR; Lambert et al., 2024) and techniques like Group Relative Policy Optimization (GRPO; Shao et al., 2024) optimize models by sampling multiple rollouts per query and updating the policy based on the relative advantage of each generated response.

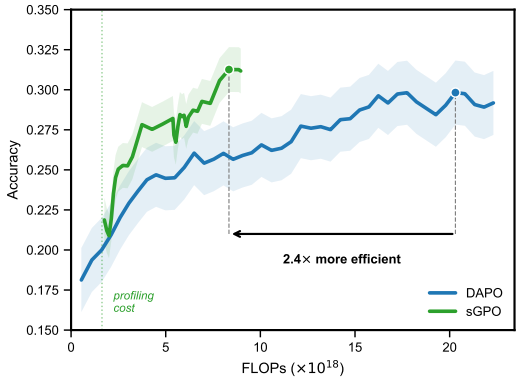
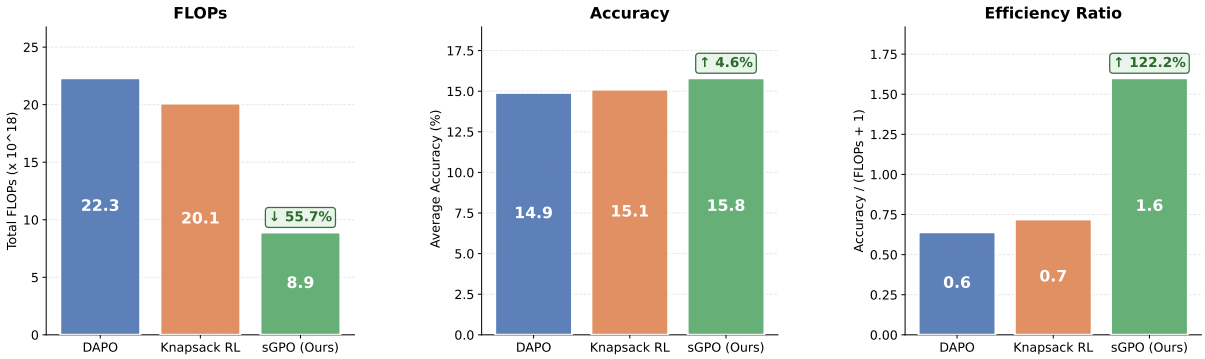


Figure 1: Accuracy-compute frontier for sGPO vs. DAPO on Qwen2.5-Math-7B. sGPO dominates throughout training, achieving the same accuracy at substantially lower total FLOPs. Appendix F for more efficiency results.

Standard RLVR training allocates a fixed rollout budget to every query regardless of difficulty, which leads to two symmetric failure modes. For trivial queries, the policy already solves them, so all rollouts succeed, and the relative advantage collapses to zero. For unsolvable queries, the policy never succeeds, again yielding no gradient. Both regimes waste training FLOPs without contributing a learning signal (Figure 1).

Compute Asymmetry This waste is particularly costly due to a fundamental *compute asymmetry* in large language models: inference requires remarkably less FLOPs per token compared to policy training. Standard RLVR squanders these expensive training FLOPs on uninformative queries early on during training. The key insight is to exploit this asymmetry: spend a small upfront budget of *cheap* inference FLOPs to profile the dataset, then drastically avoid wasting the con-



(a) Total FLOPs (↓). (b) Average Accuracy (↑). (c) Efficiency Ratio (↑).

Figure 2: Compute cost, accuracy, and efficiency of sGPO vs. baselines on Qwen2.5-Math-7B. *(a)* sGPO requires 8.9 EF total - 2.5× less than DAPO (22.3 EF) and 2.3× less than Knapsack RL (20.1 EF), including a one-time profiling cost of 1.6 EF. *(b)* Despite this reduction, sGPO achieves comparable average accuracy (15.8%) across 5 math benchmarks. *(c)* sGPO yields a 122.2% better efficiency ratio, confirming it extracts substantially more learning per unit of compute.

sumption of *expensive* training FLOPs.

Existing methods address parts of this problem; online filtering (Zheng et al., 2026) removes uninformative queries but retains uniform G, saving only 2% of FLOPs in our experiments; online solvers (Li et al., 2025) adapt G per query but require a blind first epoch and achieve only 10% savings. Both approaches operate online, recomputing decisions from training-time statistics at every step (Table 1).

We introduce *sorted Group Policy Optimization* (sGPO), which optimizes the total computation of RLVR from a compute-allocation perspective. A single *offline* profiling pass, costing ∼7% of the baseline compute, generates N samples per query under the initial policy to obtain an empirical success rate $\hat{p}(q)$. This one quantity determines filtering thresholds, per-query group sizes ($\hat{G} \approx 1/\hat{p}$), and curriculum ordering, reducing total FLOPs by 60% while maintaining accuracy and heavily boosting compute efficiency (Figure 2).

Contributions Our contributions are:

(i) We propose sGPO, a compute-efficient framework that uses a single offline profiling pass to jointly filter data, allocate adaptive group sizes, and construct an easy-to-hard curriculum.

(ii) We derive a sample-efficient heuristic for rollout group sizing in RLVR, showing that $\hat{G}(q) \approx 1/\hat{p}(q)$ provides the large non-zero advantage for a single successful rollout.

(iii) We empirically validate sGPO on math and science reasoning benchmarks, where it matches or improves upon fixed-budget DAPO while reducing training FLOPs by 60% without loss of performance.

2 Background

Policy Optimization GRPO-like (Guo et al., 2025) algorithms sample a group of G rollouts per task from the model, $\{\tau_i\}_{i=1}^{G} \sim p_\theta(\tau \mid q)$, and normalize their rewards into advantages:

$$a_i = \frac{r_i - \bar{r}}{\sigma_r}, \quad i = 1, \ldots, G, \tag{1}$$

where $\bar{r} = \frac{1}{G}\sum_{j=1}^{G} r_j$ and $\sigma_r^2 = \frac{1}{G}\sum_{j=1}^{G}(r_j - \bar{r})^2$ are sample statistics for a finite group G.

More generally, GRPO-like objectives may also include additional components, such as divergence regularization with respect to a reference model, clipping, importance weighting for batched training, and alternative definitions of the advantage (Guo et al., 2025; Liu et al., 2025; Yu et al., 2025). In this work, however, we focus on the core ingredients of the method—policy optimization and advantage computation—because our approach depends only on statistics that can be computed for any GRPO-like algorithm, as long as a rollout group of size G is sampled.

Each token in rollouts i receives the same advantage a_i, and the policy is updated via a clipped surrogate objective. This approach works well when at least some rollouts succeed. However, when all G rollouts receive the same reward, every advantage is zero and the gradient vanishes—leaving the policy with no learning signal.

Given a dataset $\mathcal{D} = \{q_j\}_{j=1}^{|\mathcal{D}|}$ and a fixed group size G, the batch estimator (Williams and Peng,

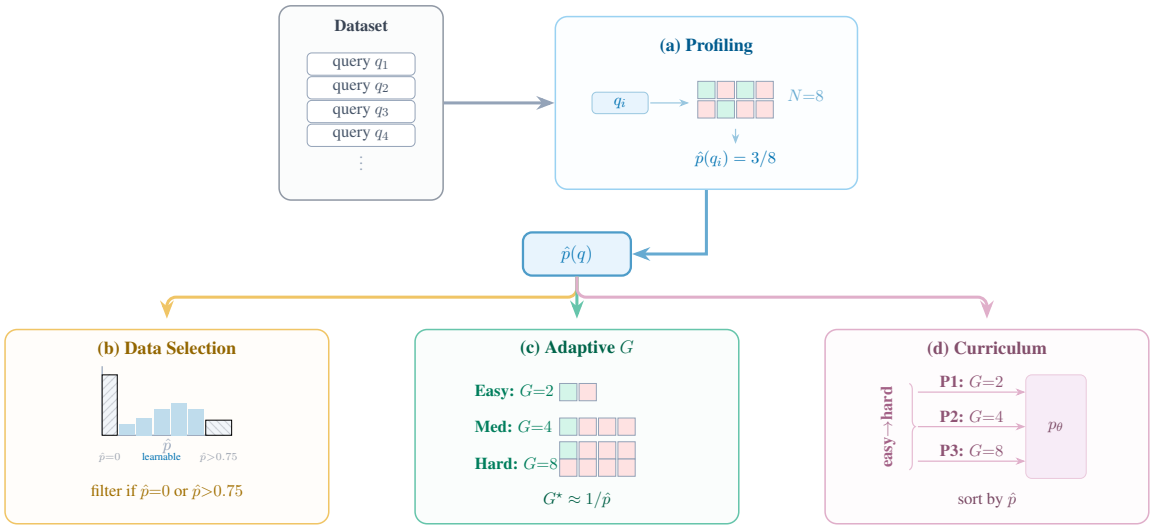


Figure 3: The SGPO pipeline. **(a) Profiling:** N=8 samples per query yield an empirical success rate $\hat{p}(q)$. This single signal drives all downstream decisions: **(b)** filtering queries with $\hat{p}$=0 or $\hat{p}$>0.75, **(c)** assigning rollout group sizes $\hat{G} \approx 1/\hat{p}$, and **(d)** ordering the curriculum from easy to hard by $\hat{p}$.

1991) is:

$$\hat{\mathcal{F}}(\theta) = \frac{1}{|\mathcal{D}|\, G} \sum_{j=1}^{|\mathcal{D}|} \sum_{i=1}^{G} a(\tau_i, q_j), \tag{2}$$

and corresponding empirical gradients $\nabla_\theta \hat{\mathcal{F}}(\theta)$:

$$\frac{1}{|\mathcal{D}|\, G} \sum_{j=1}^{|\mathcal{D}|} \sum_{i=1}^{G} a(\tau_i, q_j) \nabla_\theta \log p_\theta(\tau_i \mid q_j). \tag{3}$$

These design choices arise because the sampler lacks a model-aware estimate of query difficulty—a gap that SGPO fills.

Inference-Time Scaling Inference-time scaling (ITS, (Brown et al., 2024; Snell et al., 2024)) improves reasoning by allocating additional compute at generation time, typically through multiple sampled candidates and verifier-based selection (Lightman et al., 2023). In tasks with verifiable rewards, ITS can expose latent capability without changing model parameters. Following such properties, we use a cheap ITS pass before training to estimate how often the initial policy solves each query.

3 Related Work

Adaptive Rollout Allocation Methods for non-uniform compute allocation in RLVR fall into two families. The first filters queries: GRESO (Zheng et al., 2026) skips prompts with consistent reward history, and DEPO (Zhao et al., 2026) combines PageRank-based selection with DPP diversity sampling. The second varies G per query using online difficulty estimates: Knapsack RL (Li et al., 2025) via knapsack optimization, VIP (Nguyen et al., 2026) and CoBA-RL (Yao et al., 2026) via probabilistic models, AR3PO (Zhang et al., 2025b) via Bayesian and replay-based approaches, and GDRO (Panaganti et al., 2026) via adversarial DRO under a fixed mean-budget constraint. All operate *online*, recomputing decisions from training-time statistics at every step or epoch, which is computationally suboptimal. SGPO instead profiles once offline: a single N=8 inference pass jointly determines filtering, group sizes, and curriculum ordering before training begins (Table 1).

Method	Offline Profiling	Data Selection	Adaptive Allocation	Curriculum Building
GRESO	✗	✓	✗	✗
Knapsack RL	✗	✗	✓	✗
SGPO (Ours)	✓	✓	✓	✓

Table 1: Comparison of compute allocation strategies for RLVR training. GRESO uses online profiling, filter-based data selection, uniform allocation, and random ordering; Knapsack RL uses online profiling, no data selection, solver-based allocation, and random ordering; SGPO uses offline profiling, filter + mix data selection, inference-based allocation, and easy-to-hard curriculum building. Only SGPO combines all four components through a cheap offline ITS profiling pass.

Curriculum Learning for Reasoning Curriculum learning (Bengio et al., 2009) proposes

scheduling data from easy to hard. In RLVR, E2H Reasoner (Parashar et al., 2025) provides convergence bounds for easy-to-hard ordering, SEC (Chen et al., 2025) constructs self-evolving curricula from advantage proxies, VCRL (Jiang et al., 2025) uses reward variance, TACLer (Lai and Nissim, 2026) and ADCL (Zhang et al., 2025a) re-estimate difficulty during training, and Light-R1 (Wen et al., 2025) stages curriculum across SFT, DPO, and RL. All treat ordering as an independent design choice; in sGPO, the curriculum is a direct consequence of $\hat{p}(q)$—the same signal that drives filtering and group sizing.

Compute-optimal Scaling Snell et al. (2024) show that adaptive inference-time compute outperforms uniform scaling, and Damani et al. (2024) extend this to input-adaptive computation. sGPO applies this perspective at *training* time, using the profiling pass to concentrate expensive training FLOPs on queries where the learning signal is highest.

4 Method

sGPO optimizes the compute allocation of RLVR training from a sample efficiency perspective: rather than assigning a fixed rollout budget to every query, a small upfront profiling pass estimates $\hat{p}(q)$, which then guides filtering, rollout allocation, and curriculum design from a single offline signal (Figure 3).

4.1 Profiling via Inference-Time Scaling

The foundation of sGPO is a single, offline profiling pass that measures the empirical success rate of the initial policy p_θ on every query in the dataset. For each query q, we generate N parallel samples and compute the empirical success rate $\hat{p}(q)$:

$$\hat{p}(q) = \frac{n_{\text{profiling}}(q)}{N}, \tag{4}$$

where $n_{\text{profiling}}(q)$ is the number of correct responses among the N profiling samples.

Rather than treating difficulty as a fixed extrinsic property, this profiling defines difficulty relative to the base policy: a query is only as hard as how often p_θ fails on it.

The profiling pass is cheap: at $N = 8$ samples per query, profiling the full dataset costs approximately one GPU-hour—a one-time cost amortized across all subsequent training phases.

4.2 Sample-Efficient Advantage

For binary rewards[1] and finite group G, the realized advantage a_i of rollout i with $n = \sum_{j=1}^{G} r_j$ successes in the group is

$$a_i = \frac{r_i - \bar{r}}{\sigma_r}, \quad \bar{r} = \frac{1}{G}\sum_{j=1}^{G} r_j, \quad \sigma_r = \sqrt{\bar{r}(1-\bar{r})}. \tag{5}$$

For a successful rollout ($r_i = 1$), this simplifies to

$$a_i\big|_{r_i=1} = \frac{1 - n/G}{\sqrt{n/G\,(1-n/G)}} = \sqrt{\frac{G-n}{n}}, \tag{6}$$

which vanishes when $n = G$ (all rollouts correct)—confirming the zero-gradient collapse that the filtering step targets. Conversely, if $n = 0$, there are no successful rollouts to carry the advantage, again yielding no useful signal.

Between these extremes, the advantage of a successful rollout $a_i|_{r_i=1} = \sqrt{(G-n)/n}$ strictly decreases as the number of successes n increases. *Therefore, to maximize sample efficiency, we want the smallest group that still surfaces a success.* Targeting exactly $n = 1$ achieves this, yielding the maximum non-zero advantage of $\sqrt{G-1}$ for a successful rollout.

Learning Signal vs Efficiency While a balanced split of successes and failures ($n = G/2$) can maximize the intra-group reward variance σ_r^2 (Eq. 5) to provide strong gradient update in terms of global learning signal, achieving this balance on hard queries ($p \ll 1$) requires prohibitively large group sizes. Instead, targeting $n = 1$ explicitly prioritizes compute efficiency by extracting the maximum advantage from a single successful rollout. The goal of sGPO is to *maximize learning per FLOP*: it targets the smallest G that reliably produces a non-zero gradient, rather than the G that maximizes the total learning signal.

Under a Binomial model, the expected number of successes in a group of size G is $\mathbb{E}[n] = Gp$, where p is the true underlying success probability of the policy for task q. By setting this expectation to our target of a single success ($\mathbb{E}[n] = 1$), we obtain the ideal theoretical group size, under the base policy, for sample efficiency: $G = \frac{1}{p}$.

Since the true probability p is unknown, we substitute it with the empirical estimate $\hat{p}(q)$ obtained

[1] We assume a Bernoulli model over the sample rewards, i.e., $r \sim \mathcal{B}(p)$.

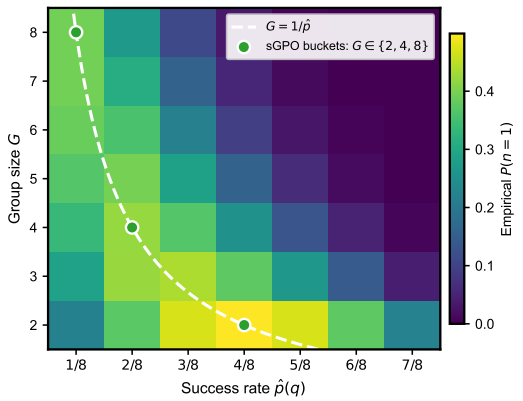


Figure 4: Empirical single-success probability $P(n{=}1)$ as a function of $\hat{p}(q)$ and G, computed by subsampling the profiling data. The bright ridge tracks the theoretical frontier $G = 1/\hat{p}$ (dashed), where exactly one success per group is most likely. sGPO's discrete assignments $G \in \{2, 4, 8\}$ (green dots) follow this ridge. Above the frontier, groups contain redundant successes; below it, groups risk no correct rollout.

from the offline profiling pass ($N = 8$). This motivates the approximate training group size heuristic:

$$\hat{G}(q) \approx \frac{1}{\hat{p}(q)}, \quad \hat{p}(q) = \frac{n_{\text{profiling}}(q)}{N}, \tag{7}$$

where $\hat{p}(q)$ is estimated from an offline profiling pass with budget N, and $\hat{G}(q)$ denotes the resulting approximate per-query training allocation. Because N is fixed during profiling whereas G is chosen per query during training, the relation $\hat{G}(q) = 1/\hat{p}(q)$ should be understood as a practical heuristic.

Choosing $G \approx 1/p$ balances the two failure modes identified above: too few rollouts risk no successes, too many produce redundant ones.

4.3 Training Strategies from $\hat{p}$

The profiled success rate $\hat{p}(q)$ determines three training decisions, each derived directly from the same quantity: which queries to train on, how many rollouts each receives, and in what order they are presented.

4.3.1 Data Selection

Using the profiled success rates $\hat{p}_j$ and a filtering threshold t, we partition the training dataset $\mathcal{D}$ into three subsets:

$$\begin{aligned} \mathcal{D}_{\text{trivial}} &= \{q_j \in \mathcal{D} \mid \hat{p}_j > t\}, \\ \mathcal{D}_{\text{unsolved}} &= \{q_j \in \mathcal{D} \mid \hat{p}_j = 0\}, \\ \mathcal{D}_{\text{learnable}} &= \{q_j \in \mathcal{D} \mid 0 < \hat{p}_j \leq t\}. \end{aligned} \tag{8}$$

Trivial queries ($\hat{p}_j > 0.75$) are removed entirely. By the analysis in Section 4.2, they produce near-zero advantage and consume training FLOPs without benefit.

Unsolved queries ($\hat{p}_j = 0$ at $N = 8$) produce no learning signal in expectation under the current policy and is therefore excluded from the primary training clusters. However, discarding them entirely would forgo long-horizon exploration. Instead, we subsample a fraction α of $\mathcal{D}_{\text{unsolved}}$ and mix it into every training phase as $\tilde{\mathcal{U}}_g \subset \mathcal{D}_{\text{unsolved}}$, *encouraging the policy to continue searching over high-complexity reasoning paths throughout training*. We use $\alpha = 10\%$ and ablate this choice in Section 5.3. The training cluster for phase g, augmented with the unsolved subsample, is:

$$\bar{C}_g = C_g \cup \tilde{\mathcal{U}}_g, \quad g \in \{2, 4, 8\}. \tag{9}$$

4.3.2 Adaptive Group Size

Learnable queries ($0 < \hat{p}_j \leq t$) form the core training set. Since training requires a fixed group size per batch, we discretize $\hat{G}(q) \approx 1/\hat{p}(q)$ into three power-of-two buckets:

$$b(q) = \begin{cases} 2 & \text{if } \hat{p}(q) \in (1/4,\ t], \\ 4 & \text{if } \hat{p}(q) \in (1/8,\ 1/4], \\ 8 & \text{if } \hat{p}(q) \in (0,\ 1/8], \end{cases} \quad G_j = b(q_j), \tag{10}$$

where $t = 0.75$ is the trivial-query threshold. With profiling budget $N{=}8$, the success rate $\hat{p}(q)$ takes discrete values in $\{0, 1/8, \ldots, 1\}$, so the bucket boundaries align with the profiling resolution. At $\hat{p} \in \{1/8, 1/4, 1/2\}$, the heuristic $\hat{G} = 1/\hat{p}$ coincides exactly with the assigned G and $\mathbb{E}[n] = 1$. At other values (e.g., $\hat{p} = 3/8$ with $G{=}2$), $\mathbb{E}[n] = 0.75$, which remains in the region where gradient signal is non-zero (Figure 4).

Applying the bucket map partitions $\mathcal{D}_{\text{learnable}}$ into three rollout clusters:

$$C_g = \{q_j \in \mathcal{D}_{\text{learnable}} \mid G_j = g\}, \tag{11}$$

for $g \in \{2, 4, 8\}$.

4.3.3 Curriculum Ordering

The three clusters are traversed in ascending difficulty order:

$$\theta^{\star} = \text{SeqTrain}\left(\hat{\mathcal{F}}_2 \ \rightarrow\ \hat{\mathcal{F}}_4 \ \rightarrow\ \hat{\mathcal{F}}_8\right). \tag{12}$$

Starting with $G = 2$ clusters (high-$\hat{p}$ queries) ensures the policy first refines capabilities where the

advantage signal from Eq. (6) is reliable and dense, before progressively encountering harder queries that require larger groups to surface successful trajectories. The consistent injection of unsolved queries across all phases prevents the policy from overfitting to easy data in early phases.

4.4 Training Objective

For each phase g, we optimize the cluster-specific advantage objective:

$$\hat{\mathcal{F}}_g(\theta) = \frac{1}{|\bar{C}_g|} \sum_{q_j \in \bar{C}_g} \frac{1}{G_j} \sum_{i=1}^{G_j} a(\tau_i, q_j), \qquad (13)$$

with $G_j = g$ for all $q_j \in \bar{C}_g$. When $G_j = G$ for all j and no partitioning or mixing is applied, this reduces exactly to the standard fixed-budget objective in Eq. (2).

Gradients The gradients for the SGPO estimator $\hat{\mathcal{F}}_g(\theta)$ for group cluster g are:

$$\frac{1}{|\bar{C}_g|} \sum_{q_j \in \bar{C}_g} \frac{1}{G_j} \sum_{i=1}^{G_j} a(\tau_i, q_j) \nabla_\theta \log p_\theta(\tau_i \mid q_j). \qquad (14)$$

By construction, $a(\tau_i, q_j)$ provides an effective learning signal over each group query G_j and group cluster $\tilde{C}_g$ in expectation.

Implementation We instantiate SGPO on top of DAPO (Yu et al., 2025). The full pipeline is summarized in Algorithm 1.

5 Experiments

We evaluate SGPO on mathematical reasoning and scientific question answering. Our experiments ask whether offline profiling improves accuracy per FLOP over online and uniform baselines, and whether these gains generalize across domains with different structures.

Models We conduct our experiments with both pretrained and instruction-tuned models. We use Qwen2.5-Math (base) series (Yang et al., 2024) as our primary models for mathematical reasoning experiments and Qwen3-4B-Instruct-2507 (Yang et al., 2025) for cross-domain science experiments.

Datasets For mathematics, we train on DAPO-Math-17k (Yu et al., 2025), a curated dataset of 14,116 English mathematical reasoning problems with verifiable answers. For science reasoning, we use SciKnowEval (Feng et al., 2024), which contains undergraduate-level scientific multiple-choice questions across four domains: chemistry, physics, biology, and materials science.

Baselines We compare SGPO against three baselines (Table 1). DAPO (Yu et al., 2025) uses a uniform rollout group size G=8 with dynamic sampling, filtering zero-variance groups only after rollout generation and ordering data randomly. GRESO (Zheng et al., 2026) performs filtering before rollout by probabilistically skipping prompts with zero-variance history, but retains uniform G and random ordering. Knapsack RL (Li et al., 2025) allocates G online via knapsack optimization over per-query success rates from training history, but without pre-training profiling its first epoch is identical to uniform allocation.

Evaluation We evaluate on mathematical reasoning (AIME 2024/2025/2026 and HMMT February 2025/2026 (Balunović et al., 2025)) and scientific reasoning (SciKnowEval-L3 (Feng et al., 2024) across chemistry, physics, biology, and materials science). Throughout, we report avg@16 accuracy; full setup is in Appendix G.

5.1 Main Results

Mathematical Reasoning Table 2 shows that SGPO achieves the best compute-performance trade-off across both model scales. At 7B, it obtains the highest average accuracy (15.8%) while requiring only 8.9 EF total - 2.5× less than DAPO and 2.3× less than Knapsack RL. The one-time profiling cost of 1.6 EF reduces the RL training budget to 7.3 EF, well below DAPO (22.3 EF) and Knapsack RL (20.1 EF), and SGPO achieves the best individual scores on AIME 2024, AIME 2026, and HMMT 2026.

At 1.5B, SGPO again leads on AIME 2024 (16.7%) at just 1.5 EF total - roughly 3× less than DAPO (4.4 EF) and Knapsack RL (4.7 EF) - with a profiling overhead of only 0.4 EF. GRESO provides negligible savings and weaker accuracy at both scales; Knapsack RL matches SGPO in accuracy but captures only modest FLOPs reductions. Figure 1 confirms the trend: SGPO maintains a superior accuracy-compute frontier throughout training and reaches comparable peak performance substantially earlier.

Method	Efficiency (FLOPs) ↓			Performance (Accuracy) ↑					
	Profiling	Training	Total	AIME24	AIME25	AIME26	HMMT25	HMMT26	Avg
Qwen2.5-Math-1.5B									
Base model	—	—	—	9.4	4.0	6.7	0.6	1.5	4.4
GRESO	0.0	4.3	4.3	11.5	5.8	8.5	0.2	**3.4**	5.9
DAPO	0.0	4.4	4.4	15.0	4.6	8.1	0.8	1.7	6.0
Knapsack RL	0.0	4.7	4.7	16.5	**6.8**	**9.0**	0.8	3.0	**7.2**
sGPO (ours)	0.4	**1.1**	**1.5**	**16.7**	5.8	8.8	**1.0**	2.7	7.0
Qwen2.5-Math-7B									
Base model	—	—	—	16.9	5.4	7.1	1.0	2.7	6.6
GRESO	0.0	21.9	21.9	26.7	12.9	11.5	2.1	6.1	11.9
DAPO	0.0	22.3	22.3	29.8	12.5	14.0	2.9	5.4	14.9
Knapsack RL	0.0	20.1	20.1	28.9	**14.8**	11.9	**3.8**	6.1	15.1
sGPO (ours)	1.6	**7.3**	**8.9**	**31.0**	13.5	**15.4**	2.5	**6.6**	**15.8**

Table 2: Main results on mathematical reasoning benchmarks with different model scales. FLOPs are reported in ExaFLOPs ($\times 10^{18}$); evaluation scores are avg@16 accuracy (%). Profiling FLOPs denotes the one-time inference cost; Train FLOPs denotes the RL training loop cost. sGPO matches or exceeds the strongest baseline across both model scales while requiring 2.5–3.1$\times$ fewer total FLOPs than the strongest baseline, even after accounting for the upfront profiling budget.

Method	Efficiency (FLOPs) ↓	Performance (Accuracy) ↑					
	Total (Profiling)	Chemistry	Physics	Biology	Materials	Avg	w Avg
Base model	—	36.8	53.0	25.5	64.9	45.1	44.2
DAPO	5.2 (0.0)	63.3	**66.6**	26.8	**74.8**	57.9	61.6
sGPO (ours)	**1.9 (0.4)**	**64.3**	65.7	**27.8**	74.0	**58.0**	**61.9**

Table 3: Cross-domain results on SciKnowEval using a generalist model, Qwen3-4B-Instruct. FLOPs are total ExaFLOPs ($\times 10^{18}$), including sGPO's one-time profiling cost of 0.4 EF; scores are avg@16 accuracy (%). sGPO matches DAPO at 2.7$\times$ fewer FLOPs.

Scientific Reasoning To evaluate *cross-domain generalization*, we apply sGPO to SciKnowEval using Qwen3-4B-Instruct (Table 3). sGPO requires only 1.9 EF total FLOPs, compared with 5.2 EF for DAPO, yielding a 2.7$\times$ reduction in compute. Despite this substantially lower budget, sGPO slightly improves over DAPO in both average accuracy and weighted average accuracy. The one-time profiling cost is just 0.4 EF, accounting for 21% of the total budget. sGPO also achieves the best results in Chemistry, Biology, average accuracy, and weighted average accuracy, while remaining competitive in Physics and Materials. These results suggest that the efficiency gains from sGPO extend beyond mathematical reasoning to scientific reasoning tasks with different domain structure and reward characteristics.

Key Takeaways Across both domains, sGPO delivers the best compute-performance trade-off: 2.5–3.1$\times$ fewer total FLOPs than DAPO on mathematics and 2.7$\times$ fewer on science, while matching or slightly exceeding accuracy in both settings. The consistent gains confirm that offline profiling generalizes beyond mathematical reasoning - the same cheap inference pass that drives efficiency on AIME and HMMT transfers directly to multi-domain scientific QA.

5.2 Analysis

Inference-training Cost Asymmetry At 7B scale, 1.6 EF of inference-only profiling ($\sim$7% of DAPO's total budget) determines which queries to keep, what group size each needs, and in what order to present them. Because every training step uses these decisions, the model learns efficiently from step one: no wasted rollouts on trivial or unsolvable queries, no oversized groups on easy problems. The result is a training reduction from 22.3 EF to 7.3 EF, with the 1.6 EF profiling cost paid in cheap inference tokens (Figure 2c).

Online Baselines GRESO and Knapsack RL both estimate difficulty online from training-time

rewards. GRESO filters queries but retains uniform G=8 for all kept queries, saving 2% of FLOPs (21.9 vs. 22.3 EF at 7B) while degrading AIME 2024 accuracy from 29.8% to 26.7%. Filtering alone does not address per-query compute waste: an easy query at $\hat{p} = 0.5$ still generates $\mathbb{E}[n] = 4$ redundant successes per group. Knapsack RL adapts G per query but requires a blind first epoch at uniform G=8 before its solver has signal, limiting total savings to 10% (20.1 vs. 22.3 EF).

Efficiency Decomposition sGPO's 15.0 EF training reduction over DAPO at 7B decomposes into two sources. Filtering (removing trivial and unsolved queries) reduces the training set from 14,116 to 7,525 queries, accounting for 10.4 EF (69% of the savings). Adaptive group sizing reduces the average G from 8 to 4.1 across the remaining queries, saving an additional 4.6 EF (31%). Neither component alone matches sGPO's total savings: filtering without adaptive G still wastes rollouts on easy queries (as GRESO demonstrates), while adaptive G without filtering still trains on unsolvable and trivial problems.

5.3 Ablations

Method	FLOPs ($\times 10^{18}$)	AIME24$_{\text{avg@16}}$
sGPO	**8.9**	**31.0**
− Curriculum	8.9	29.6
− Adaptive G	10.7	27.5
− Filtering (DAPO)	22.3	29.8

Table 4: Component ablation of sGPO on Qwen2.5-Math-7B. Rows are cumulative: each removes one additional component from the full method, ending at DAPO.

Table 4 cumulatively removes sGPO components.

Curriculum Removing easy-to-hard ordering drops accuracy from 31.0% to 29.6% with no change in FLOPs. Ordering queries by difficulty costs nothing in compute but improves sample efficiency by 1.4pp: the model builds capability on high-$\hat{p}$ queries in the G=2 phase before encountering low-$\hat{p}$ queries in the G=8 phase.

Adaptive Group Additionally removing per-query group sizes (reverting to uniform G=8) increases FLOPs from 8.9 to 10.7 EF and drops accuracy to 27.5%. Without adaptive sizing, every kept query generates eight rollouts regardless of difficulty, wasting compute on queries where fewer rollouts would suffice.

Mixing α	FLOPs ($\times 10^{18}$)	AIME24$_{\text{avg@16}}$
0%	5.9	28.3
10%	8.9	**31.0**
25%	9.4	29.8
30%	10.2	29.2

Table 5: Effect of unsolved mixing ratio α on Qwen2.5-Math-7B. α controls the fraction of $\mathcal{D}_{\text{unsolved}}$ mixed into each curriculum phase.

Filtering Filtering without adaptive G (27.5%) performs worse than no filtering at all (DAPO, 29.8%). Removing queries shrinks the dataset but does not reduce per-query cost, so the model sees fewer problems with the same per-problem waste. Filtering is beneficial only when adaptive G reduces the cost of the queries that remain (consistent with GRESO; Section 5.2).

Unsolved Mixing Ratio Table 5 varies α, the fraction of $\mathcal{D}_{\text{unsolved}}$ mixed into each curriculum phase. Accuracy peaks at α=10% (31.0%). Lower values drop accuracy (e.g., 28.3% at α=0%) by starving the policy of hard exploration targets, though α=0% remains viable for extreme compute constraints (5.9 EF). Higher values degrade performance by diluting the learnable set with zero-gradient rollouts.

Multiple Epochs All main results use two epochs with stale profiling assignments (same $\hat{G}$ as epoch 1). As detailed in Appendix D, re-profiling the dataset after epoch 1 degrades accuracy ($\sim$25% vs. $\sim$32% for stale replay) because it over-concentrates training on overly hard, low-signal problems. Retaining the original assignments preserves necessary difficulty diversity.

6 Conclusion

Standard RLVR training allocates compute uniformly, wasting FLOPs on queries the policy has already mastered or cannot yet solve. By using a single offline profiling pass to drive data filtering, group sizing, and curriculum ordering, sGPO trades cheap inference FLOPs for expensive training FLOPs to match baseline accuracy while reducing total compute by 2.5-3.1×. More broadly, these results suggest that a small investment in offline difficulty estimation can substitute for a large portion of online compute in RLVR.

Limitations While sGPO significantly improves reinforcement learning efficiency, it has a few inherent limitations. First, it is designed for Reinforcement Learning with Verifiable Rewards (RLVR) using objective, binary metrics, making it difficult to apply to subjective human preference alignment (RLHF). Second, the framework relies on static difficulty estimates from a single offline pass. Because the model improves during training, these initial estimates can become outdated, requiring re-profiling to maintain optimal compute allocation. In our experiments, we profile once at initialization and find that the single-pass estimates remain predictive enough to yield large compute savings; however, for longer training runs or rapidly shifting difficulty distributions, periodic re-profiling at epoch boundaries would be a natural extension.

Ethical Considerations By reducing training compute and minimizing wasted rollouts, sGPO lowers the energy consumption and carbon footprint of aligning large language models. This efficiency also democratizes AI development, allowing researchers with fewer hardware resources to train highly capable reasoning models. However, this accessibility raises dual-use risks, potentially making it easier for malicious actors to optimize models for harmful applications. Furthermore, the method's aggressive data filtering, which discards trivial prompts and subsamples unsolvable ones, risks introducing representational harms. If the initial policy's success correlates with specific cultural or linguistic contexts, this filtering could inadvertently amplify existing biases and degrade performance on underrepresented data distributions.

References

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

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

Xiaoyin Chen, Jiarui Lu, Minsu Kim, Dinghuai Zhang, Jian Tang, Alexandre Piché, Nicolas Gontier, Yoshua Bengio, and Ehsan Kamalloo. 2025. Self-evolving curriculum for llm reasoning. *arXiv preprint arXiv:2505.14970*.

Mehul Damani, Idan Shenfeld, Andi Peng, Andreea Bobu, and Jacob Andreas. 2024. Learning how hard to think: Input-adaptive allocation of lm computation. *arXiv preprint arXiv:2410.04707*.

Kehua Feng, Xinyi Shen, Weijie Wang, Xiang Zhuang, Yuqi Tang, Qiang Zhang, and Keyan Ding. 2024. Sciknoweval: Evaluating multi-level scientific knowledge of large language models. *arXiv preprint arXiv:2406.09098*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, DDL Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, and 1 others. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10.

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*.

Guochao Jiang, Wenfeng Feng, Guofeng Quan, Chuzhan Hao, Yuewei Zhang, Guohua Liu, and Hao Wang. 2025. Vcrl: Variance-based curriculum reinforcement learning for large language models. *arXiv preprint arXiv:2509.19803*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Huiyuan Lai and Malvina Nissim. 2026. Tacler: Tailored curriculum reinforcement learning for efficient reasoning. *arXiv preprint arXiv:2601.21711*.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, and 1 others. 2024. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.

Ziniu Li, Congliang Chen, Tianyun Yang, Tian Ding, Ruoyu Sun, Ge Zhang, Wenhao Huang, and Zhi-Quan Luo. 2025. Knapsack rl: Unlocking exploration of llms via optimizing budget allocation. *arXiv preprint arXiv:2509.25849*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

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*.

Hieu Trung Nguyen, Bao Nguyen, Wenao Ma, Yuzhi Zhao, Ruifeng She, and Viet Anh Nguyen. 2026. Adaptive rollout allocation for online reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2602.01601*.

Kishan Panaganti, Zhenwen Liang, Wenhao Yu, Haitao Mi, and Dong Yu. 2026. Group distributionally robust optimization-driven reinforcement learning for llm reasoning. *arXiv preprint arXiv:2601.19280*.

Shubham Parashar, Shurui Gui, Xiner Li, Hongyi Ling, Sushil Vemuri, Blake Olson, Eric Li, Yu Zhang, James Caverlee, Dileep Kalathil, and 1 others. 2025. Curriculum reinforcement learning from easy to hard tasks improves llm reasoning. *arXiv preprint arXiv:2506.06632*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Tanglifu Tanglifu, Xiaowei Lv, and 1 others. 2025. Light-r1: Curriculum sft, dpo and rl for long cot from scratch and beyond. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 318–327.

Ronald J Williams and Jing Peng. 1991. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, and 1 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Zhiyuan Yao, Yi-Kai Zhang, Yuxin Chen, Yueqing Sun, Zishan Xu, Yu Yang, Tianhao Hu, Qi Gu, Hui Su, and Xunliang Cai. 2026. Coba-rl: Capability-oriented budget allocation for reinforcement learning in llms. *arXiv preprint arXiv:2602.03048*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale, 2025. *URL https://arxiv.org/abs/2503.14476*, 1:2.

Enci Zhang, Xingang Yan, Wei Lin, Tianxiang Zhang, and Lu Qianchun. 2025a. Learning like humans: Advancing llm reasoning capabilities via adaptive difficulty curriculum learning and expert-guided self-reformulation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 6630–6644.

Yuheng Zhang, Wenlin Yao, Changlong Yu, Yao Liu, Qingyu Yin, Bing Yin, Hyokun Yun, and Lihong Li. 2025b. Improving sampling efficiency in rlvr through adaptive rollout and response reuse. *arXiv preprint arXiv:2509.25808*.

Yu Zhao, Fan Jiang, Tianle Liu, Bo Zeng, Yu Liu, Longyue Wang, and Weihua Luo. 2026. Difficulty-estimated policy optimization. *arXiv preprint arXiv:2602.06375*.

Haizhong Zheng, Yang Zhou, Brian Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. 2026. Act only when it pays: Efficient reinforcement learning for llm reasoning via selective rollouts. *Advances in Neural Information Processing Systems*, 38:124321–124346.

A Algorithm

Algorithm 1 sorted Group Policy Optimization (sGPO)

Require: Training dataset $\mathcal{D}$, Initial policy p_θ, Profiling budget $N = 8$, Group sizes $\mathcal{G} = \{2, 4, 8\}$, Threshold $t = 0.75$

Profiling via Inference-Time Scaling

for each query $q \in \mathcal{D}$ **do**
  $\{\tau_i\}_{i=1}^{N} \sim p_\theta(\tau \mid q)$ // Generate N parallel samples
  $\hat{p}(q) = \dfrac{n_{\text{profiling}}(q)}{N}$ // Estimate empirical success rate
end for

Dataset Partitioning and Clustering

$\mathcal{D}_{\text{trivial}} \leftarrow \{q \in \mathcal{D} \mid \hat{p}(q) > t\}$ // Remove trivial
$\mathcal{D}_{\text{unsolved}} \leftarrow \{q \in \mathcal{D} \mid \hat{p}(q) = 0\}$ // Remove unsolved
$\mathcal{D}_{\text{learnable}} \leftarrow \{q \in \mathcal{D} \mid 0 < \hat{p}(q) \leq t\}$ // Use Learnable
Apply bucket map $b(q)$ to partition $\mathcal{D}_{\text{learnable}}$ into clusters C_2, C_4, C_8

Curriculum Mixing and Training

$\tilde{\mathcal{U}} \leftarrow \text{Subsample}(\mathcal{D}_{\text{unsolved}})$
for each group size $G \in \{2, 4, 8\}$ in ascending order **do**
  $\bar{C}_G \leftarrow C_G \cup \tilde{\mathcal{U}}$ // Mix unsolved queries into the cluster
  Train policy p_θ for one epoch on $\bar{C}_G$ using fixed group size G
end for

return Optimized policy p_θ

B Training Dynamics

Figure 5 shows four training metrics across the three curriculum phases for Qwen2.5-Math-7B. Solid lines are epoch 1; dashed lines are epoch 2 (same data, fresh rollouts).

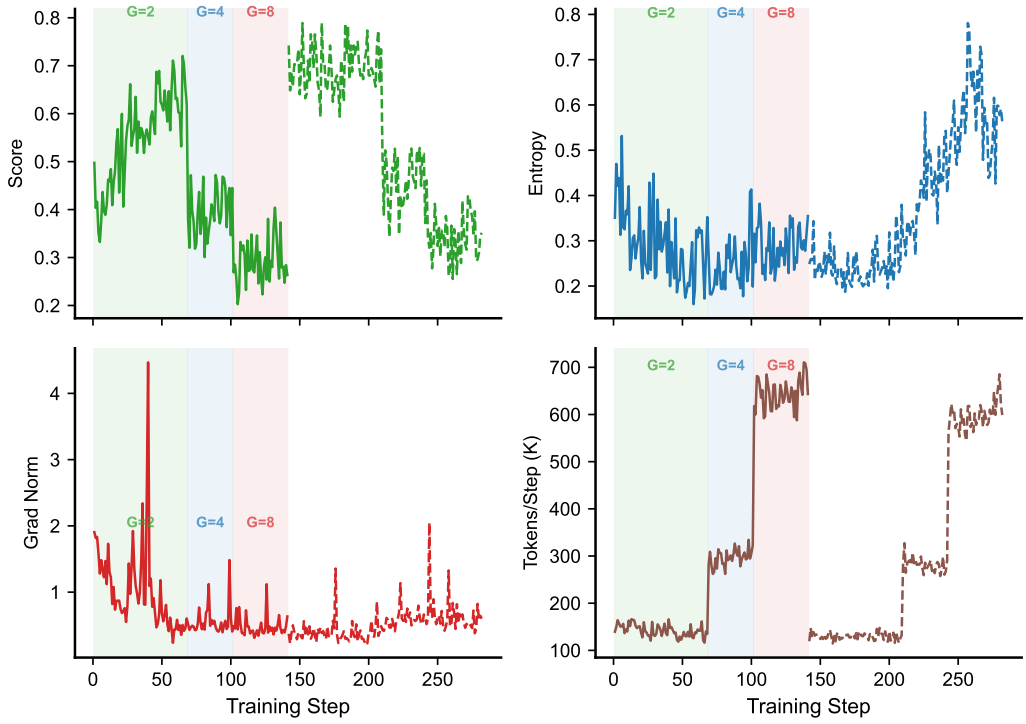


Figure 5: Training dynamics across curriculum phases for SGPO on Qwen2.5-Math-7B. Top-left: mean reward score, top-right: policy entropy, bottom-left: gradient norm, bottom-right: tokens per step. Solid: epoch 1, dashed: epoch 2. Phase bands indicate $G{=}2$ (green), $G{=}4$ (blue), $G{=}8$ (red).

Score Mean reward score increases within the $G{=}2$ and $G{=}4$ phases and drops at phase transitions as harder queries are introduced. The $G{=}2$ phase peaks at 0.72 and $G{=}4$ at 0.45. The $G{=}8$ phase peaks at 0.40 but ends at 0.26, close to its starting value of 0.27. This flat trajectory suggests that the hardest learnable queries ($\hat{p} = 1/8$) provide limited per-step improvement at this scale, though the ablation in Table 4 confirms that including this phase still contributes to downstream accuracy. In epoch 2, the $G{=}2$ phase starts at 0.74 (vs 0.50 in epoch 1), reflecting retained capability.

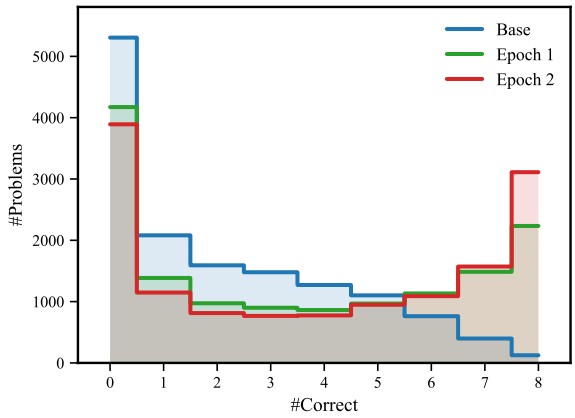
Entropy Policy entropy remains between 0.20 and 0.35 during the $G{=}2$ phase. During $G{=}4$, entropy decreases from 0.27 to 0.20, consistent with the model narrowing its output distribution as it learns to solve medium-difficulty queries more reliably. During the $G{=}8$ phase, entropy rises (0.32→0.36 in epoch 1, 0.45→0.56 in epoch 2), coinciding with the model encountering queries where no profiled success exists.

Gradient norm Gradient norms peak in the $G{=}2$ phase (avg 1.02) and decrease in later phases (0.56 for $G{=}4$, 0.48 for $G{=}8$). The largest single-step norm (4.47) occurs at training step 1. Later phases produce smaller gradients because the policy has already been updated on easier queries.

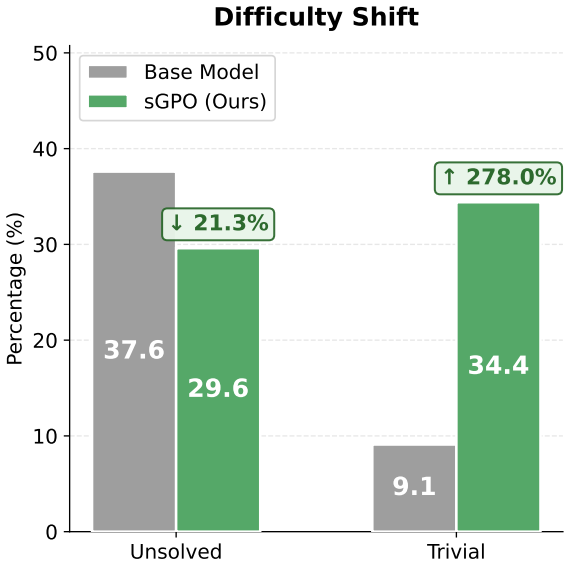
Tokens per step Token count scales approximately with G: 142K for $G{=}2$, 297K for $G{=}4$, 645K for $G{=}8$. The slight super-linear scaling (2.1× and 2.2× per doubling of G) reflects longer average responses on harder queries. The curriculum front-loads the cheapest phase, spending fewer tokens per step when score improvements are largest.

C Learning Effectiveness

We re-profile the full dataset after one epoch of sGPO training on Qwen2.5-Math-7B to measure how the difficulty distribution changes.



(a) Full per-bin distribution before (left) and after (right) training.



(b) Unsolved and trivial fractions before and after training.

Figure 6: Difficulty distribution shift after one epoch of sGPO on Qwen2.5-Math-7B. (a) The full profiling distribution across all $\hat{p}$ bins. (b) The two extreme categories: unsolved drops from 37.6% to 29.6% (-8.0pp), trivial grows from 9.1% to 34.4% ($+25.3$pp).

Figure 6a shows the full per-bin distribution before and after training. The base model concentrates at two uninformative extremes: 37.6% unsolved ($\hat{p}$=0) and 9.1% trivial ($\hat{p} \geq 0.75$). After one epoch, probability mass shifts from the unsolved region toward intermediate and high-correct bins. Figure 6b summarizes the endpoints: the trivial fraction grows by 25.3pp (9.1% $\rightarrow$ 34.4%) and the unsolved fraction shrinks by 8.0pp (37.6% $\rightarrow$ 29.6%).

Two findings:

1. **Learnable queries are mastered.** The 25.3pp increase in trivial queries shows that the G=2 phase converts high-$\hat{p}$ problems from learnable to solved. These queries would produce zero gradient under the same G^* assignment in a second epoch.

2. **Unsolved queries become solvable.** 8.0pp of previously unsolved queries ($\hat{p}$=0) become solvable after training. The 10% unsolved mixing (α=10%) in each phase provides direct exposure, while capability built on easier queries during the G=2 and G=4 phases may also transfer. The ablation in Table 5 confirms that removing unsolved mixing (α=0%) drops accuracy by 2.7pp.

The post-training distribution differs substantially from the pre-training profile, confirming that $\hat{G}$ assignments become stale over training. This motivates the re-profiling direction discussed in Section 6.

D Multi-Epoch Training

All main results in this paper use two epochs of SGPO training. This section describes the epoch 2 strategy selection that led to this default.

We evaluated three strategies for continuing into epoch 2 on Qwen2.5-Math-7B, all starting from the same epoch 1 checkpoint:

1. **Stale replay:** Reuse the original profiling assignments ($\hat{p}$ and $\hat{G}$ from the base model). The curriculum repeats with identical data ordering and group sizes.

2. **Re-profiled:** Re-profile the dataset using the epoch 1 checkpoint to obtain updated $\hat{p}$ values, then recompute $\hat{G}$, filtering, and curriculum ordering.

3. **Re-profiled + fresh optimizer:** Same as (2) but reset the optimizer state before epoch 2.

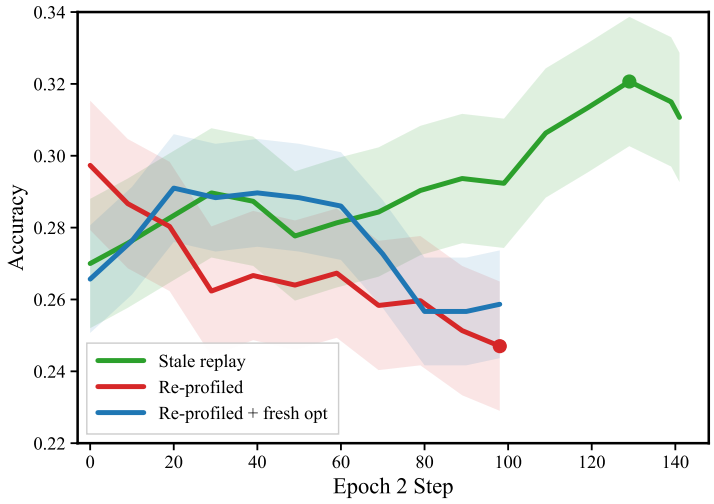


Figure 7: Epoch 2 accuracy on Qwen2.5-Math-7B under three continuation strategies. Stale replay (green) improves to $\sim$32%. Both re-profiled variants (red, blue) degrade to $\sim$25%.

Figure 7 shows the epoch 2 accuracy trajectories. Stale replay improves from 31.0% to $\sim$32%. Both re-profiled variants degrade to $\sim$25%, below the epoch 1 checkpoint.

Comparing variants (2) and (3) isolates the effect of optimizer state: both use re-profiled assignments, but (3) resets the optimizer. The two perform identically ($\sim$25%), indicating that optimizer momentum does not drive the difference. Comparing variants (1) and (2) isolates the effect of assignments: both continue the same optimizer, but (1) uses stale profiling. Stale replay outperforms by $\sim$7pp. The profiling assignments, not the optimizer state, determine epoch 2 performance.

Why re-profiling fails The $\mathbb{E}[n] \approx 1$ analysis in Section 4.2 explains this. After epoch 1, 25.3pp of previously learnable queries reach $\hat{p} \geq 0.75$ under the trained model and are filtered as trivial (Appendix C). The re-profiled dataset concentrates on hard and unsolved problems. The G=2 cluster shrinks, eliminating the easy queries that produced the densest gradient signal in epoch 1. Most remaining queries fall into the $\mathbb{E}[n] \approx 0$ dead zone, where groups produce all-incorrect rollouts and zero advantage.

Why stale replay works Consider a query that was $\hat{p} = 1/8$ (G=8, $\mathbb{E}[n]$=1) at profiling time and now has true success probability $p \approx 3/8$ after epoch 1. Its stale assignment G=8 yields $\mathbb{E}[n] = 3$: suboptimal but still producing non-zero advantage, because the group contains a mix of correct and incorrect rollouts. The advantage landscape (Figure 4) has a broad ridge around $\mathbb{E}[n]$=1, and $\mathbb{E}[n]$=3 remains in the productive region. Re-profiling, by contrast, concentrates the entire distribution near $\mathbb{E}[n]$=0.

Implication In this setting, retaining difficulty diversity outperforms recalibrating $\hat{G}$ to the model's current capability. A training distribution with a range of $\mathbb{E}[n]$ values (some at 1, some at 2–3, some near 0) produces gradient on average, while a distribution concentrated at $\mathbb{E}[n] \approx 0$ produces nothing. Based on this finding, all main results use stale replay for epoch 2.

E Self-Consistency vs. Verified Profiling

A natural question is whether SGPO's profiling pass requires ground-truth verification, or whether a cheaper signal such as self-consistency (SC (Wang et al., 2022)) could substitute. SC measures agreement among sampled responses without checking correctness: if all N responses agree, the query is classified as easy regardless of whether the consensus answer is right.

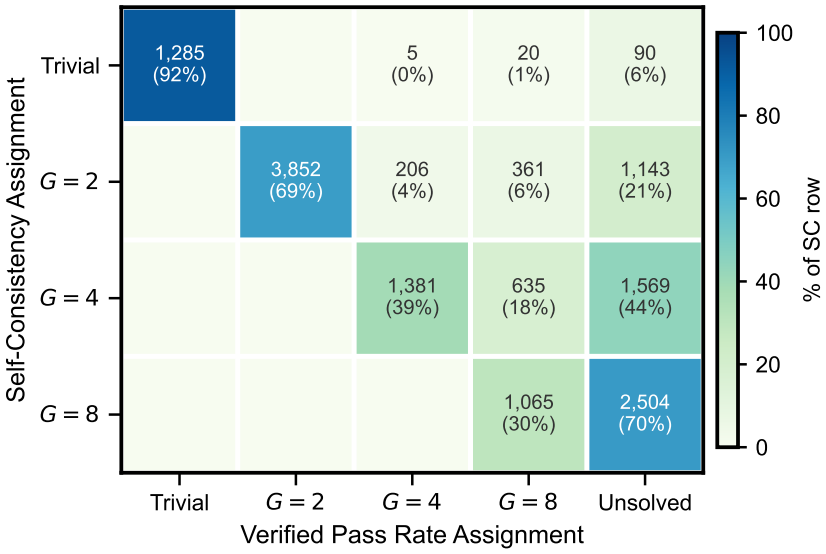


Figure 8: Confusion matrix between self-consistency (SC) and verified pass-rate difficulty assignments on DAPO-Math-17k (Qwen2.5-Math-7B, N=8). Each cell shows the count and row-normalized percentage. SC never classifies queries as unsolved (no "Unsolved" row exists) because it cannot distinguish consistent correctness from consistent error. The "Unsolved" column intensifies downward: 21% of SC's G=2 queries, 44% of G=4, and 70% of G=8 are actually unsolved.

Figure 8 cross-tabulates SC-based and pass-rate-based difficulty assignments. SC agrees with verified profiling for trivial queries (92% correct) and G=2 queries (69% correct). The disagreement concentrates in the "Unsolved" column: 1,143 queries that SC assigns to G=2, 1,569 to G=4, and 2,504 to G=8 are in fact unsolved ($\hat{p}$=0). In these cases, the model produces consistent but incorrect answers, and SC interprets the agreement as evidence of capability.

SC-based profiling would allocate training compute to these queries, generating zero gradient signal because no correct response exists in the rollout group. Verified profiling avoids this: queries with $\hat{p}$=0 are identified and filtered (or subsampled at α=10%). The cost of verification is low when rewards are binary and automatically checkable, the standard setting in RLVR.

F Uniform Group Scaling

Does increasing the uniform group size close the gap with adaptive allocation? Figure 9 compares SGPO ($G \in \{2, 4, 8\}$, max group size 8) against DAPO with G=16 on Qwen2.5-Math-7B.

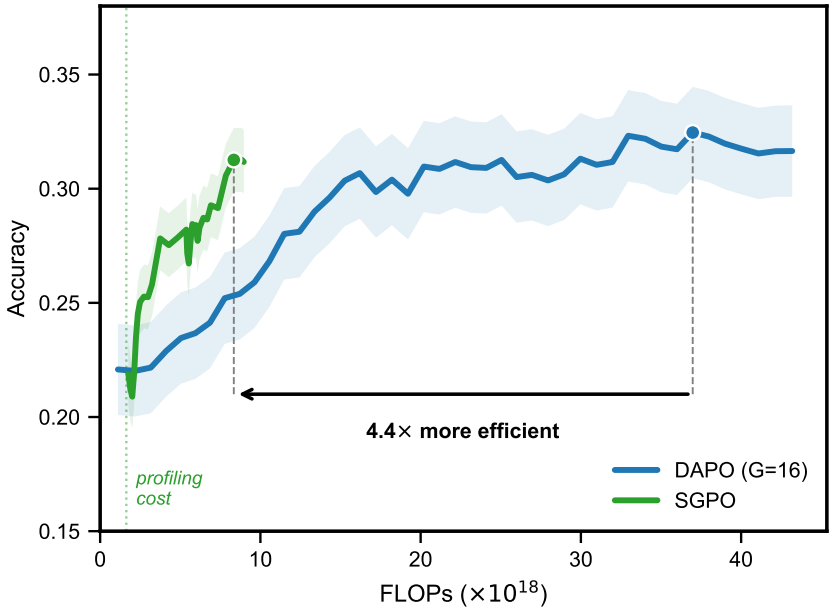


Figure 9: Accuracy vs. cumulative FLOPs for SGPO and DAPO (G=16) on Qwen2.5-Math-7B. SGPO reaches comparable peak accuracy 4.4× faster.

DAPO G=16 peaks at ∼32%, 1pp above SGPO's 31.0%, but consumes ∼36 EF, which is 4.4× more compute than SGPO's 8.9 EF. Doubling the group size from G=8 to G=16 increases DAPO's total FLOPs by 61% (22.3 → 36 EF) while improving peak accuracy by only ∼2pp. Most of the additional rollouts land on queries where the model already succeeds or consistently fails, producing zero advantage.

SGPO matches this accuracy with a maximum group size of 8 by allocating G=8 only to the hardest learnable queries ($\hat{p} = 1/8$). Easier queries receive G=2 or G=4, avoiding the redundant rollouts that dominate uniform G=16 training.

G Experimental Details

G.1 Datasets and Models

Tables 6, 7, and 8 summarize the datasets, benchmarks, and models used in this work.

Dataset	Problems	HuggingFace Url
DAPO-Math-17k	14,116	`BytedTsinghua-SIA/DAPO-Math-17k`
SciKnowEval	3901	`hicai-zju/SciKnowEval`

Table 6: Training datasets. DAPO-Math-17k contains 14,116 English problems after deduplication. SciKnowEval uses a 90/10 train-test split of the L3 (reasoning) subset.

Benchmark	Problems	HuggingFace Url
AIME 2024	30	`Maxwell-Jia/AIME_2024`
AIME 2025	30	`MathArena/aime_2025`
AIME 2026	30	`MathArena/aime_2026`
HMMT Feb 2025	30	`MathArena/hmmt_feb_2025`
HMMT Feb 2026	33	`MathArena/hmmt_feb_2026`
SciKnowEval-L3 Chemistry	210	`hicai-zju/SciKnowEval`
SciKnowEval-L3 Physics	80	`hicai-zju/SciKnowEval`
SciKnowEval-L3 Biology	50	`hicai-zju/SciKnowEval`
SciKnowEval-L3 Material Sciences	94	`hicai-zju/SciKnowEval`

Table 7: Evaluation benchmarks. SciKnowEval counts reflect the 10% test split.

Model	Params	HuggingFace Path
Qwen2.5-Math-1.5B (Yang et al., 2024)	1.5B	`Qwen/Qwen2.5-Math-1.5B`
Qwen2.5-Math-7B (Yang et al., 2024)	7B	`Qwen/Qwen2.5-Math-7B`
Qwen3-4B-Instruct-2507 (Yang et al., 2025)	4B	`Qwen/Qwen3-4B-Instruct-2507`

Table 8: Models used in our experiments. Qwen2.5-Math models are base (non-instruct) checkpoints. Qwen3-4B is instruction-tuned.

G.2 Prompt

Prompt Template

```
<|im_start|>system
Please reason step by step, and put your final answer within \boxed{}.
<|im_end|>
<|im_start|>user
Please solve the following math problem: {{Question}}. The assistant first thinks about the reasoning process step
by step and then provides the user with the answer. Return the final answer in \boxed{} tags, for example \boxed{1}.
Let's solve this step by step.
<|im_end|>
<|im_start|>assistant
```

Figure 10: Prompt template used for training and evaluation. The system message is auto-injected by the Qwen2.5-Math tokenizer. The user message wraps the problem in the instruction format.

G.3 Training and Evaluation Setup

All experiments use DAPO as the underlying RL optimizer, trained on $8\times$ H100 GPUs. sGPO adds profiling, filtering, rollout allocation, and curriculum ordering on top of the standard DAPO pipeline; all other hyperparameters (learning rate, KL penalty, clipping) remain unchanged.

The profiling budget is fixed at $N = 8$ samples per query. We define trivial queries as those with $\hat{p}(q) > 0.75$, unsolved queries as those with $\hat{p}(q) = 0$, and learnable queries as those with $0 < \hat{p}(q) \leq 0.75$. Learnable queries are bucketed into rollout groups $G \in \{2, 4, 8\}$, and we mix an $\alpha = 10\%$ subsample of unsolved queries into each curriculum phase, training sequentially over $\bar{C}_2 \to \bar{C}_4 \to \bar{C}_8$.

Evaluation uses avg@16 accuracy: for each query, we generate 16 independent samples at temperature 1.0 with top-$p = 0.7$ and report the fraction of samples that are correct, averaged across all queries.

G.4 FLOP Accounting

We report compute in ExaFLOPs ($\times 10^{18}$), where P denotes the number of model parameters. We follow the standard approximation that a transformer forward pass costs $\sim 2P$ FLOPs per token (Kaplan et al., 2020; Hoffmann et al., 2022).

Profiling cost. The profiling pass performs inference only (autoregressive generation, no gradient computation). Each token costs $2P$ FLOPs:

$$\text{FLOPs}_{\text{profiling}} = 2P \cdot T_{\text{profiling}},$$

where $T_{\text{profiling}}$ is the total number of tokens generated across all $N{=}8$ samples for all queries.

Training cost. Each GRPO-style training step processes generated tokens through multiple stages. For DAPO (Yu et al., 2025), which uses a frozen reference model for KL regularization, the per-token cost breaks down as:

1. **Rollout generation** (actor, forward only): $2P$
2. **Reference log-probabilities** (reference model, forward only): $2P$
3. **Actor update** (forward + backward): $2P + 4P = 6P$

This totals $\sim 10P$ per token. In practice, implementation overhead (gradient accumulation, KL computation, advantage normalization) and framework-level inefficiencies raise the effective cost. We use $12P$ as a conservative upper bound, consistent with estimates in prior work (Li et al., 2025). Methods that eliminate the reference model (e.g., Dr. GRPO (Liu et al., 2025)) reduce this to $\sim 8P$, which would further increase sGPO's relative advantage.

$$\text{FLOPs}_{\text{train}} = 12P \cdot T_{\text{train}}$$

Total cost.

$$\text{FLOPs}_{\text{total}} = \text{FLOPs}_{\text{profiling}} + \text{FLOPs}_{\text{train}} = 2P \cdot T_{\text{profiling}} + 12P \cdot T_{\text{train}}$$

The $6\times$ gap between inference ($2P$) and training ($12P$) per token is the cost asymmetry that sGPO exploits: profiling tokens are cheap, training tokens are expensive.