DELIBERATERL: SHAPING THE PRACTICE DISTRIBUTION FOR COMPUTE-EFFICIENT RLVR

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) typically assigns a fixed rollout budget to every query, even though success probabilities under the policy vary widely. As a result, many response groups are all-correct or all-incorrect: they consume generation compute but yield no group-relative learning signal. We frame allocation as shaping the *practice distribution*—how training effort is distributed across queries of different policy competence over the course of training. Inspired by deliberate practice, DELIBERATERL is a competence-aware offline framework that estimates each query's success rate once, with a verified inference-only profiling pass, and uses this estimate to decide *what* to practice, *how much* to practice each query, and *when*. We instantiate it with sorted-Group Policy Optimization (sGPO), which filters high-success queries, sizes rollout groups toward roughly one expected success, orders training from easier to harder queries, and, following spaced repetition, revisits a small fixed subset of zero-observed-success queries throughout the curriculum. Across mathematical and scientific reasoning tasks with Qwen- and Llama-family models, sGPO matches the accuracy of uniform and online-adaptive baselines while using $2.5\times$ less compute than uniform allocation over a two-epoch horizon, including the one-time profiling cost.

1 INTRODUCTION

Reinforcement learning has emerged as a central paradigm for improving the reasoning capabilities and alignment of large language models (Guo et al., 2025; Jaech et al., 2024). Reinforcement Learning with Verifiable Rewards (RLVR; Lambert et al., 2024) and group-relative methods such as Group Relative Policy Optimization (GRPO; Shao et al., 2024) sample multiple responses per query and update from their relative rewards, so their effectiveness depends on reward variation within sampled groups. We focus on binary outcome rewards, where an all-correct or all-incorrect group has zero reward variance and provides no group-relative learning signal.[1]

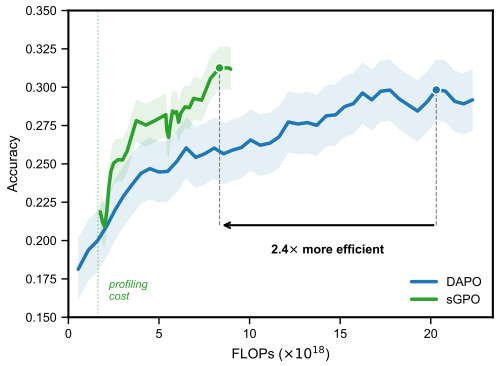
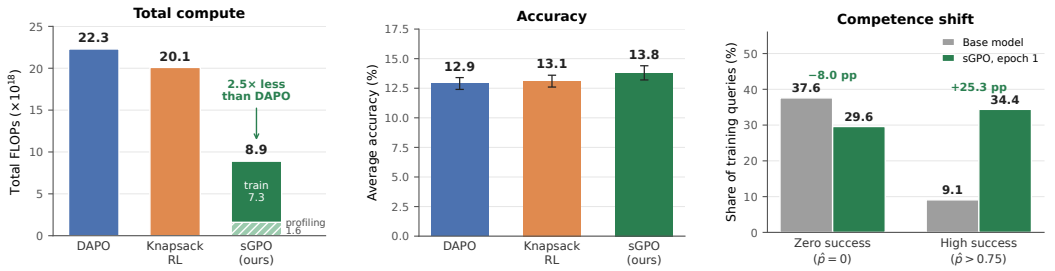


Figure 1: Accuracy–compute comparison between sGPO and DAPO on Qwen2.5-Math-7B. sGPO reaches comparable accuracy at lower cumulative estimated FLOPs, including its one-time profiling cost. Appendix G compares against a larger uniform rollout group.

Neither regime is rare: on DAPO-Math-17k (Yu et al., 2026), 37.6% of queries yield no success in eight samples from Qwen2.5-Math-7B, while a further 9.1% are solved in at least seven of eight (Figure 2c). Even if such groups are discarded before optimization, their generation cost has already been paid. The allocation problem

[1]This argument does not directly extend to process or dense rewards that distinguish trajectories with the same final outcome.

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.



(a) Total FLOPs (↓). **(b)** Average Accuracy (↑). **(c)** Competence Shift.

Figure 2: Compute, performance, and competence dynamics for sGPO on Qwen2.5-Math-7B. **(a)** sGPO uses 8.9 EF total estimated compute, including 1.6 EF of one-time profiling, compared with 22.3 EF for DAPO and 20.1 EF for Knapsack RL. **(b)** Average accuracy across five mathematical reasoning benchmarks. **(c)** After one epoch of sGPO, the zero-observed-success fraction falls from 37.6% to 29.6%, while the high-success fraction ($\hat{p} > 0.75$) rises from 9.1% to 34.4%.

is therefore not only *which* queries to train on, but also *how much* rollout compute each should receive and *when*.

Practice Distribution We view these decisions jointly as shaping the *practice distribution*: how training effort is distributed across queries of different policy competence, rollout budgets, and stages of training. For a query q with success probability $p(q)$ under the current policy and rollout budget $G(q)$, the pair determines the distribution of outcomes within a sampled group: with too few rollouts, low-success queries yield all-incorrect groups and high-success queries yield all-correct ones. A productive practice distribution places compute where groups are likely to contain reward variation; fixed-budget RLVR ignores this interaction.

An ideal allocation would track the current policy's success probabilities. sGPO approximates it with $\hat{p}(q)$, an offline estimate of initial-policy competence, and fixes the resulting assignments over the training horizon—making the allocation available before the first update, at the cost of a profile that becomes stale as the policy changes.

Trading Inference for Training Compute Under our FLOP accounting (Appendix M.5), an inference token costs about $2P$ FLOPs, whereas a token processed by the RL training loop costs about $12P$, making profiling roughly $6\times$ cheaper per token. This *compute asymmetry* creates an opportunity: spend a small amount of inference compute before training to estimate what the policy can already solve, and use it to shape the practice distribution consumed by the more expensive training phase.

Such a profile can reduce repeated sampling of low-yield queries, avoid oversized groups on easy queries, and fix a curriculum before the first RL update. It can also be reused across training runs that share the initial policy and dataset, amortizing its cost. The central question is whether the training compute it saves outweighs its upfront inference cost. Fig. 1 suggests that it does: on Qwen2.5-Math-7B, the profiling cost is paid once at the start, and sGPO reaches DAPO-level accuracy at a fraction of DAPO's cumulative FLOPs.

Previous Work Existing approaches address individual parts of this problem during RL training. Online filtering methods such as GRESO (Zheng et al., 2026) use reward histories to skip uninformative queries, while online budget allocation methods such as Knapsack RL (Li et al., 2025b) adapt the number of rollouts per query. These methods can track changing competence, but must first acquire difficulty information from training-time rollouts; our evaluated Knapsack RL implementation, for example, uses uniform allocation in the first epoch to collect statistics (Table 1). See Appendix A for additional related work.

This motivates our core question:

Can a small inference-only assessment shape a useful practice distribution before RL training?

Deliberate Repetition Inspired by deliberate practice (Ericsson et al., 1993; Ericsson, 2004; Campitelli &Gobet, 2011), which organizes effort around the learner's current competence, and by spaced repetition (Ebbinghaus, 1913; Cepeda et al., 2006; Amiri et al., 2017), which revisits the same (difficult) items at intervals rather than all at once, we introduce DELIBERATERL, a competence-aware framework for shaping this practice distribution. A single verified profiling pass generates N responses per query under the initial policy and estimates its empirical success rate $\hat{p}(q)$, which controls three coupled decisions (Fig. 3): ***what to practice*** (query selection), ***how much to practice*** (rollout-group allocation), and ***when to practice*** (curriculum ordering).

We instantiate DELIBERATERL with sorted-Group Policy Optimization (SGPO), which filters high-success prompts, assigns larger rollout groups to lower-success prompts, and presents queries from easier to harder. In the spirit of spaced repetition, a small fixed subset of zero-observed-success prompts is revisited in every curriculum phase, each time after further training on easier queries and with a larger rollout budget. The rollout-allocation rule is an inverse-success-rate heuristic motivated by singleton-success probability: under a Binomial model, we characterize the integer group sizes that maximize it exactly.[2]

Table 1: Compute-allocation strategies for the evaluated RLVR methods, organized by the three decisions of DELIBERATERL. The evaluated baselines modify at most one of these dimensions using training-time rewards; SGPO sets all three from a single offline competence profile before optimization begins.

Method	Offline profile	What (selection)	How much (group size)	When (ordering)
DAPO	✗	✗	✗	✗
GRESO	✗	✓	✗	✗
Knapsack RL	✗	✗	✓	✗
SGPO (ours)	✓	✓	✓	✓

Across mathematical and scientific reasoning tasks with Qwen- and Llama-family models, SGPO matches the accuracy of uniform and online-adaptive baselines while using 2.5–2.9× less total compute, including profiling, over a two-epoch horizon (Fig. 1). On Qwen2.5-Math-7B, the profile costs 1.6 EF and RL training 7.3 EF, for 8.9 EF total versus 22.3 EF for DAPO (Fig. 2). Because policy competence shifts substantially during training (Fig. 2c), we study DELIBERATERL as a short-horizon strategy and explicitly examine stale profiling and re-profiling.

Contributions Our contributions are:

- *(i)* We introduce the *practice distribution* view of RLVR compute allocation and DELIBERATERL, a framework that uses a single offline profile to jointly determine which prompts to train on, how much rollout compute they receive, and when they are presented.
- *(ii)* We develop SGPO, which couples competence-aware query selection, an inverse-success-rate rollout heuristic, and easy-to-hard curriculum ordering, supported by an exact discrete analysis of singleton-success probability and its correspondence with the implemented rollout buckets.
- *(iii)* We evaluate SGPO on mathematical and scientific reasoning across Qwen- and Llama-family models, finding comparable accuracy at substantially lower total estimated compute. We also characterize SGPO robustness and limitations.

2 BACKGROUND

Policy Optimization GRPO-like (Guo et al., 2025) algorithms sample a group of G rollouts for a query q, $\{\tau_i\}_{i=1}^{G} \sim p_\theta(\tau \mid q)$, and construct group-relative advantages by normalizing their rewards:

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

where $\bar{r} = \frac{1}{G}\sum_{i=1}^{G} r_i$ and $\sigma_r^2 = \frac{1}{G}\sum_{i=1}^{G}(r_i - \bar{r})^2$ are statistics of the finite rollout group. When all rewards are identical, $\sigma_r = 0$, so the advantage in Eq. (1) provides no usable relative reward signal. Such zero-variance groups contribute no policy update in practice, even though their generation cost has already been incurred. Given a dataset $\mathcal{D} = \{q_j\}_{j=1}^{|\mathcal{D}|}$, the expected-reward objective can be

[2] Section 3.2 develops this connection and distinguishes the heuristic from a guarantee of learning-optimal allocation.

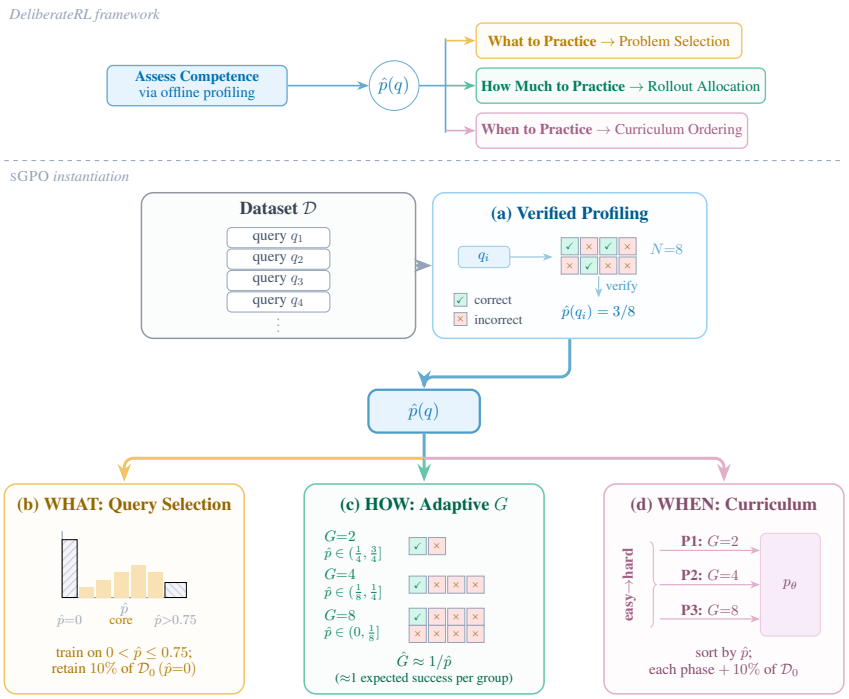


Figure 3: DeliberateRL framework and sGPO instantiation. Inspired by deliberate practice, DeliberateRL assesses the initial policy's competence once and uses the resulting estimate $\hat{p}(q)$ to decide *what* the policy practices, *how much* rollout compute each query receives, and *when* each query is presented. sGPO instantiates this with **(a)** a verified offline profiling pass using N=8 samples per query; **(b)** competence-aware query selection; **(c)** rollout-group allocation using the discretized $\hat{G} \approx 1/\hat{p}$ heuristic; and **(d)** an easy-to-hard curriculum, with a 10% sample of zero-observed-success queries $\mathcal{D}_0$ mixed into every phase.

written over the discrete trajectory space as

$$J(\theta) = \frac{1}{|\mathcal{D}|} \sum_{j=1}^{|\mathcal{D}|} \sum_{\tau} p_\theta(\tau \mid q_j)\, r(\tau, q_j). \tag{2}$$

We can compute the gradients of Eq. (2) using the score-function identity (Williams &Peng, 1991),

$$\nabla_\theta J(\theta) = \frac{1}{|\mathcal{D}|} \sum_{j=1}^{|\mathcal{D}|} \sum_{\tau} p_\theta(\tau \mid q_j)\, r(\tau, q_j) \nabla_\theta \log p_\theta(\tau \mid q_j). \tag{3}$$

GRPO-like methods approximate the update in Eq. (3) from sampled rollouts using group-relative advantages. For query q_j, let $\tau_{j,i}$ denote its i-th sampled rollout, with reward $r_{j,i}$ and corresponding group-relative advantage $a_{j,i}$. Treating the resulting advantages $a_{j,i}$ as fixed during the policy update, the corresponding Monte Carlo update is

$$\hat{\mathcal{F}}(\theta) = \frac{1}{|\mathcal{D}|G} \sum_{j=1}^{|\mathcal{D}|} \sum_{i=1}^{G} a_{j,i} \nabla_\theta \log p_\theta(\tau_{j,i} \mid q_j). \tag{4}$$

Practical GRPO-like objectives based on Eq. (4) may additionally use clipping, importance weighting, KL regularization, or related modifications (Guo et al., 2025; Liu et al., 2025; Yu et al., 2026). With a uniform G, the sampler assigns the same rollout budget to every query despite differences in policy-relative difficulty. This lack of a model-aware competence estimate is the allocation problem that sGPO addresses.

Inference-Time Scaling Inference-time scaling (ITS; Brown et al., 2024; Snell et al., 2024) improves reasoning by allocating additional generation compute, often through multiple sampled candidates and verifier-based selection (Lightman et al., 2023). In verifiable tasks, such sampling can reveal the policy's existing capability without updating its parameters. We use an inference-only profiling pass before training to estimate how often the initial policy solves each query. Snell et al. (2024) and Damani et al. (2024) motivate input-adaptive allocation of inference compute. sGPO transfers this perspective to training-time allocation: inference-only profiling estimates policy-relative competence before training, and the resulting signal guides where subsequent RL compute is allocated.

3 METHOD

Motivation DELIBERATERL follows a simple principle from deliberate practice: training effort should be organized around the learner's competence rather than distributed uniformly (Fig. 3). A single offline profiling pass estimates the initial policy's empirical success rate $\hat{p}(q)$, and this common competence signal shapes the *practice distribution*: *what* the policy practices (query selection), *how much* rollout compute each query receives (rollout allocation), and *when* each query is presented (curriculum ordering).

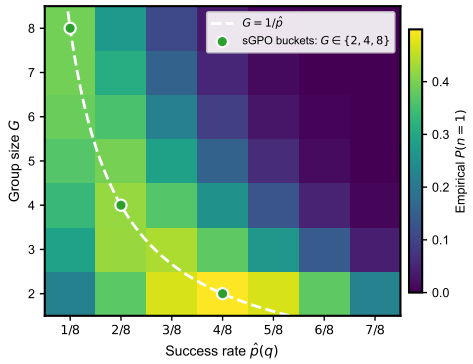


Figure 4: Empirical singleton-success probability $P(n{=}1)$ as a function of the profiled success rate $\hat{p}(q)$ and group size G. The dashed curve marks the inverse-success operating point $G = 1/\hat{p}$, which closely follows the high-probability region at low success rates. sGPO discretizes the allocation into $G \in \{2, 4, 8\}$ (green dots).

Why Offline Profiling? Profiling requires only verified generation, which is cheaper than acquiring the same signal during RL training; its estimates are available before the first update, so all three decisions are configured from the start rather than after a cold-start period; and one profile serves all three decisions, avoiding separate online estimators. The trade-off is that the estimates become stale as the policy changes. We therefore treat offline profiling as a short-horizon design choice over the two-epoch regime considered here and analyze staleness and re-profiling in Appendix J.

3.1 OFFLINE COMPETENCE PROFILING VIA INFERENCE-TIME SCALING

For each query q in the training set, we draw N samples from the initial policy p_{θ_0}, verify their rewards, and compute the empirical success rate

$$\hat{p}(q) = \frac{n_{\text{profiling}}(q)}{N}, \tag{5}$$

where $n_{\text{profiling}}(q)$ is the number of successful samples. Thus, $\hat{p}(q)$ in Eq. (5) measures difficulty relative to the initial policy rather than as an intrinsic property of the query, and is defined with respect to the profiling sampling distribution. We use $N = 8$ unless otherwise stated. The pass requires $N|\mathcal{D}|$ inference rollouts, whose cost is included in all reported compute totals.

3.2 COMPETENCE-AWARE COMPUTE ALLOCATION

Advantage Consider binary rewards $r_i \in \{0, 1\}$ and a group of G rollouts containing $n = \sum_{i=1}^{G} r_i$ successes. For a non-degenerate group, $0 < n < G$, the normalized group-relative advantage of a successful rollout (Eq. (1) with $\bar{r} = n/G$ and $\sigma_r = \sqrt{\bar{r}(1-\bar{r})}$) simplifies to

$$a_i\big|_{r_i=1} = \sqrt{\frac{G-n}{n}}. \tag{6}$$

For fixed G, the advantage in Eq. (6) decreases with n and is largest for the singleton-success configuration $n = 1$; at $n = 0$ and $n = G$, reward variance is zero and the group provides no learning signal. *This motivates singleton-success probability as an allocation surrogate.*

Group Size Suppose each rollout independently solves a query with probability p, so that $n \sim \text{Binomial}(G, p)$. The probability of exactly one success is

$$\Pr(n = 1 \mid G, p) = G\, p(1-p)^{G-1}. \tag{7}$$

Since group sizes are integers, we compare consecutive values using Eq. (7). For $p \in (0, 1)$,

$$\frac{\Pr(n = 1 \mid G+1, p)}{\Pr(n = 1 \mid G, p)} = \frac{G+1}{G}(1-p), \tag{8}$$

so, by Eq. (8), adding a rollout increases the singleton-success probability when $G < 1/p - 1$, decreases it when $G > 1/p - 1$, and leaves it unchanged at equality.[3] Subject to the constraint $G \geq 2$ (a single rollout admits no within-group comparison), one global maximizer is

$$G^{\star}_{\text{int}}(p) = \max\left\{2, \left\lfloor \frac{1}{p} \right\rfloor\right\}, \qquad p \in (0, 1). \tag{9}$$

The continuous relaxation has optimum $G^{\star}_{\text{cont}}(p) = -1/\log(1-p) \sim 1/p$ as $p \to 0$, in accordance with Eq. (9). Appendix B gives the full derivation, including tie cases and the continuous relaxation.

Intuition Since $\mathbb{E}[n] = Gp$, choosing $G \approx 1/p$ corresponds to roughly one expected success per group, without implying that every group contains exactly one (Figure 4). Because the true success probability is unknown, we use the offline estimate:

$$\hat{G}(q) \approx \frac{1}{\hat{p}(q)}, \qquad \hat{p}(q) > 0. \tag{10}$$

For batched training, sGPO restricts group sizes to $\{2, 4, 8\}$ (Section 3.3.2). With $N = 8$ and $t = 0.75$, these buckets maximize singleton-success probability over the feasible sizes for every positive retained estimate; Appendix B establishes this correspondence and its limits for finer profiling grids. Queries with zero observed successes are handled separately (Section 3.3.1).

Analysis This analysis establishes optimality for singleton-success probability under independent binary rewards with a fixed success probability, not for policy-gradient magnitude or learning progress per FLOP. Applied with $\hat{p}(q)$, finite-sample profiling error and policy changes can cause the estimate to differ from the current policy's success probability.

3.3 Training Strategies from $\hat{p}$

The profiled success rate determines three training decisions: query selection, rollout-group size, and curriculum ordering.

3.3.1 What to Practice: Query Selection

Given a filtering threshold t, we partition the training dataset $\mathcal{D}$ into

$$\mathcal{D}_{\text{high}} = \{q_j : \hat{p}_j > t\}, \quad \mathcal{D}_0 = \{q_j : \hat{p}_j = 0\}, \quad \mathcal{D}_{\text{learn}} = \{q_j : 0 < \hat{p}_j \leq t\}, \tag{11}$$

with $t = 0.75$ unless otherwise stated.

High Success Queries with $\hat{p}_j > t$ in Eq. (11) are excluded from the core training set, since additional rollouts on them are likely to produce repeated successes rather than diverse reward outcomes. The threshold is an empirical design choice; we evaluate its sensitivity in Appendix H.2.

Zero-observed Success For queries in $\mathcal{D}_0$, none of the N profiling rollouts succeeded. This does not imply that the true success probability is zero, only that no success was observed under the profiling budget. We exclude these queries from the core clusters but sample an α fraction once before training, denoted $\tilde{\mathcal{D}}_0$, and add it to every core rollout cluster $\mathcal{C}_g$ (defined in Eq. (14) below):

$$\bar{C}_g = C_g \cup \tilde{\mathcal{D}}_0, \qquad g \in \{2, 4, 8\}. \tag{12}$$

We use $\alpha = 10\%$ and ablate this choice in Section 4.3.

3.3.2 How Much to Practice: Rollout Allocation

Queries in $\mathcal{D}_{\text{learn}}$ form the core training set. We discretize Eq. (10) into three group sizes:

$$b(q) = \begin{cases} 2 & \text{if } \hat{p}(q) \in (1/4,\ t], \\ 4 & \text{if } \hat{p}(q) \in (1/8,\ 1/4], \\ 8 & \text{if } \hat{p}(q) \in (0,\ 1/8]. \end{cases} \qquad G_j = b(q_j). \tag{13}$$

With $N = 8$, $\hat{p}(q) \in \{0, 1/8, \ldots, 1\}$, so the bucket boundaries in Eq. (13) align with the profiling resolution. The resulting rollout clusters are

$$C_g = \{q_j \in \mathcal{D}_{\text{learn}} \mid G_j = g\}, \qquad g \in \{2, 4, 8\}. \tag{14}$$

[3] When $1/p = m$ is an integer, $G = m - 1$ and $G = m$ are tied; we select m, which corresponds to one expected success per group.

3.3.3 When to Practice: Curriculum Ordering

We traverse the clusters in Eq. (12) from higher to lower profiled success rate:

$$\theta^\star = \mathrm{SeqTrain}\left(\bar{C}_2 \to \bar{C}_4 \to \bar{C}_8\right), \tag{15}$$

so the curriculum in Eq. (15) begins with queries the initial policy solves most often and moves toward lower-success queries with larger rollout groups.

Spaced Re-exposure to Hard Queries The retained subset $\tilde{\mathcal{D}}_0$ is revisited in every phase rather than resampled. This follows spaced repetition, where revisiting the *same* difficult items at intervals improves acquisition over one-off exposure (Ebbinghaus, 1913; Cepeda et al., 2006). Each revisit follows further training on easier queries and receives a larger rollout budget ($2 \to 4 \to 8$), so the largest budget is spent when the policy is strongest.

Optimization sGPO changes the training-query distribution and rollout allocation while retaining the verifier reward and the DAPO policy-update rule (Yu et al., 2026); the phase-wise objective and update are given in Appendix C, and the full procedure in Algorithm 1. The profile and assignments are computed once and reused across the two-epoch horizon (Section 4.3, Appendix J).

4 Experiments

We evaluate sGPO on mathematical reasoning and scientific question answering. Our experiments ask whether offline competence profiling can preserve accuracy while reducing total training compute relative to uniform and online-adaptive baselines, and whether these savings extend across model scales and task domains.

4.1 Setup

Models We use pretrained and instruction-tuned models: the Qwen2.5-Math base series (Yang et al., 2024) for mathematical reasoning, and Qwen3-4B-Instruct-2507 (Yang et al., 2025) and Llama-3.2-3B-Instruct (Grattafiori et al., 2024) for scientific reasoning.

Datasets For mathematics, we train on DAPO-Math-17k (Yu et al., 2026), a curated set of 14,116 English problems with verifiable answers. For science, we use the L3 reasoning subset of SciKnowEval (Feng et al., 2024), which contains undergraduate-level multiple-choice questions in chemistry, physics, biology, and materials science, with a 90/10 train–test split.

Baselines We compare sGPO against three baselines. DAPO (Yu et al., 2026) 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) filters before rollout by probabilistically skipping prompts with zero-variance history, but retains uniform G and random ordering. Knapsack RL (Li et al., 2025b) allocates G online via knapsack optimization over per-query success rates from training history; 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 M.

4.2 Main Results

Mathematical Reasoning Table 2 shows that sGPO substantially reduces compute while preserving competitive accuracy across both model scales. At 7B, sGPO reaches 13.8% average accuracy using 8.9 EF total compute, compared with 12.9% at 22.3 EF for DAPO and 13.1% at 20.1 EF for Knapsack RL. The sGPO total includes 1.6 EF of one-time profiling compute; its RL training loop itself uses 7.3 EF. At 1.5B, sGPO reaches 7.0% average accuracy using 1.5 EF total compute, compared with 6.0% at 4.4 EF for DAPO and 7.2% at 4.7 EF for Knapsack RL. Thus, the reduction in compute persists across model scales without a substantial loss in average accuracy. The average-accuracy differences between sGPO and the strongest baseline at each scale ($+0.7$ points at 7B, -0.2 points at 1.5B) are within about one propagated standard error. We therefore read Table 2 as showing that accuracy is preserved at lower compute, not that it improves.

Table 2: Main results on mathematical reasoning benchmarks across model scales. Efficiency columns report estimated ExaFLOPs ($\times 10^{18}$) with H100 GPU-hours in parentheses (measured wall-clock on 8×H100); Profiling denotes the one-time inference cost and Training the RL training-loop cost. Evaluation scores are avg@16 accuracy (%). Avg is the unweighted mean over the five benchmarks; w Avg weights each benchmark by its number of problems. sGPO achieves comparable average accuracy to the strongest baseline while requiring substantially less total compute and GPU time, including its upfront profiling cost. See Appendix I for GPU-hour and token breakdowns.

	Efficiency: FLOPs (GPU-h) ↓			**Performance (Accuracy)** ↑						
Method	Profiling	Training	Total	AIME24	AIME25	AIME26	HMMT25	HMMT26	Avg	w Avg
Qwen2.5-Math-1.5B										
Base model	—	—	—	$9.4_{\pm 1.1}$	$4.0_{\pm 0.7}$	$6.7_{\pm 0.9}$	$0.6_{\pm 0.3}$	$1.5_{\pm 0.5}$	$4.4_{\pm 0.3}$	$4.4_{\pm 0.3}$
GRESO	0.0	4.3 (47.5)	4.3 (47.5)	$11.5_{\pm 1.2}$	$5.8_{\pm 0.9}$	$8.5_{\pm 1.1}$	$0.2_{\pm 0.2}$	$\mathbf{3.4}_{\pm 0.7}$	$5.9_{\pm 0.4}$	$5.8_{\pm 0.4}$
DAPO	0.0	4.4 (30.7)	4.4 (30.7)	$15.0_{\pm 1.3}$	$4.6_{\pm 0.8}$	$8.1_{\pm 1.0}$	$0.8_{\pm 0.3}$	$1.7_{\pm 0.5}$	$6.0_{\pm 0.4}$	$6.0_{\pm 0.4}$
Knapsack RL	0.0	4.7 (38.4)	4.7 (38.4)	$16.5_{\pm 1.4}$	$\mathbf{6.8}_{\pm 0.9}$	$\mathbf{9.0}_{\pm 1.1}$	$0.8_{\pm 0.3}$	$3.0_{\pm 0.6}$	$\mathbf{7.2}_{\pm 0.4}$	$\mathbf{7.1}_{\pm 0.4}$
sGPO (ours)	0.4 (2.3)	**1.1** (6.1)	**1.5** (8.4)	$\mathbf{16.7}_{\pm 1.4}$	$5.8_{\pm 0.9}$	$8.8_{\pm 1.1}$	$\mathbf{1.0}_{\pm 0.4}$	$2.7_{\pm 0.6}$	$7.0_{\pm 0.4}$	$6.9_{\pm 0.4}$
Qwen2.5-Math-7B										
Base model	—	—	—	$16.9_{\pm 1.4}$	$5.4_{\pm 0.9}$	$7.1_{\pm 1.0}$	$1.0_{\pm 0.4}$	$2.7_{\pm 0.6}$	$6.6_{\pm 0.4}$	$6.5_{\pm 0.4}$
GRESO	0.0	21.9 (66.8)	21.9 (66.8)	$26.7_{\pm 1.7}$	$12.9_{\pm 1.3}$	$11.5_{\pm 1.2}$	$2.1_{\pm 0.5}$	$6.1_{\pm 0.9}$	$11.9_{\pm 0.5}$	$11.7_{\pm 0.5}$
DAPO	0.0	22.3 (80.7)	22.3 (80.7)	$29.8_{\pm 1.7}$	$12.5_{\pm 1.2}$	$14.0_{\pm 1.3}$	$2.9_{\pm 0.6}$	$5.4_{\pm 0.9}$	$12.9_{\pm 0.5}$	$12.8_{\pm 0.5}$
Knapsack RL	0.0	20.1 (90.8)	20.1 (90.8)	$28.9_{\pm 1.7}$	$\mathbf{14.8}_{\pm 1.3}$	$11.9_{\pm 1.2}$	$\mathbf{3.8}_{\pm 0.7}$	$6.1_{\pm 0.9}$	$13.1_{\pm 0.5}$	$13.0_{\pm 0.5}$
sGPO (ours)	1.6 (4.8)	**7.3** (21.9)	**8.9** (26.7)	$\mathbf{31.0}_{\pm 1.7}$	$13.5_{\pm 1.3}$	$\mathbf{15.4}_{\pm 1.4}$	$2.5_{\pm 0.6}$	$\mathbf{6.6}_{\pm 0.9}$	$\mathbf{13.8}_{\pm 0.6}$	$\mathbf{13.7}_{\pm 0.6}$

Scientific Reasoning To test whether the allocation strategy extends beyond mathematical reasoning and beyond the Qwen family, we apply sGPO to SciKnowEval using Qwen3-4B-Instruct and Llama-3.2-3B-Instruct (Table 3). On Qwen, sGPO uses 1.9 EF total estimated compute, including 0.4 EF of profiling, compared with 5.2 EF for DAPO—a 2.7× reduction—while average accuracy is essentially unchanged (58.0% versus 57.9%; weighted 62.5% versus 62.2%). On Llama, sGPO uses 1.5 EF compared with 4.2 EF for DAPO (2.8× fewer), with slightly higher average accuracy (29.2% versus 28.9%; weighted 30.1% versus 29.7%). These results indicate that the compute savings extend to a scientific question-answering setting and to a second model family. The appendix additionally compares against GRESO and Knapsack RL on Llama.

Table 3: Cross-domain results on SciKnowEval-L3 using two generalist models from different families, Qwen3-4B-Instruct and Llama-3.2-3B-Instruct. FLOPs are estimated ExaFLOPs ($\times 10^{18}$); Profiling denotes the one-time inference cost and Training denotes the RL training-loop cost. Scores are avg@16 accuracy (%). Avg is the unweighted mean over the four domains; w Avg weights each domain by its number of test problems. sGPO achieves comparable or higher average accuracy using 2.7× (Qwen) and 2.8× (Llama) less total estimated compute, including its upfront profiling cost.

	Efficiency (FLOPs) ↓			**Performance (Accuracy)** ↑					
Method	Profiling	Training	Total	Chemistry	Physics	Biology	Materials	Avg	w Avg
Qwen3-4B-Instruct									
Base model	—	—	—	36.8	53.0	25.5	64.9	45.1	44.6
DAPO	0.0	5.2	5.2	63.3	**66.6**	26.8	**74.8**	57.9	62.2
sGPO (ours)	0.4	**1.5**	**1.9**	**64.3**	65.7	**27.8**	74.0	**58.0**	**62.5**
Llama-3.2-3B-Instruct									
Base model	—	—	—	16.4	27.6	23.6	24.1	22.9	21.0
DAPO	0.0	4.2	4.2	30.6	30.1	25.4	29.5	28.9	29.7
sGPO (ours)	0.3	**1.2**	**1.5**	**31.1**	**30.3**	**25.6**	**29.9**	**29.2**	**30.1**

Key Takeaways Across both domains, sGPO matches the average accuracy of the baselines while using substantially less compute: 2.5–2.9× fewer total FLOPs than DAPO on mathematics and 2.7–2.8× fewer on science, with the one-time profiling cost included. A single offline profiling pass thus yields large compute savings in both mathematical reasoning and scientific question answering.[4]

[4]Our goal is compute efficiency at comparable accuracy; these results do not establish accuracy gains over every baseline.

4.3 Ablations and Analysis

We next examine how the three allocation decisions in sGPO contribute to its accuracy–compute trade-off, how much exposure to zero-observed-success queries is useful, and how static profiling behaves across the evaluated two-epoch horizon.

Table 4: Component ablation of sGPO on Qwen2.5-Math-7B. Rows cumulatively remove curriculum ordering (*when*), adaptive rollout allocation (*how much*), and competence-aware query selection (*what*).

Method	FLOPs (EF)	AIME24$_{\text{avg@16}}$
sGPO	**8.9**	**31.0**
– Curriculum	8.9	29.6
– Adaptive G	10.7	27.5
– Query Selection	22.3	29.8

Table 5: Effect of the zero-observed-success retention rate α on Qwen2.5-Math-7B. α controls the fraction of $\mathcal{D}_0$ retained once and reused across curriculum phases.

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

Component Ablation Table 4 mirrors the three decisions used by DeliberateRL to shape the practice distribution. Removing the easy-to-hard curriculum (*when*) lowers AIME24 accuracy from 31.0% to 29.6% at unchanged compute, suggesting that ordering contributes beyond query selection and rollout allocation. Removing adaptive rollout allocation (*how much*) as well, and reverting to uniform $G = 8$, increases compute from 8.9 to 10.7 EF while lowering accuracy to 27.5%. Finally, removing competence-aware query selection (*what*) recovers the DAPO setting and raises total compute to 22.3 EF.

These results suggest complementary roles for the three decisions: query selection determines which prompts enter the core training set, rollout allocation determines how much generation compute retained prompts receive, and curriculum ordering determines when they are presented. On its own, query selection reduces compute (10.7 vs. 22.3 EF) but lowers AIME24 accuracy (27.5% vs. 29.8%); the accuracy gain appears only when it is combined with adaptive G and curriculum ordering.

Zero-Observed Success Retention Table 5 varies α, the fraction of $\mathcal{D}_0 = \{q : \hat{p}(q) = 0\}$ retained for training. The retained subset is sampled once and reused across all phases and both epochs. Among the tested values, $\alpha = 10\%$ achieves the highest AIME24 accuracy. Excluding these queries entirely reduces compute but also reduces accuracy, whereas larger retention rates increase the amount of training devoted to queries with no observed profiling success without improving accuracy. We therefore use $\alpha = 10\%$ as an empirical trade-off between exploratory coverage and concentration on queries with demonstrated initial-policy success.

Static Profiling Across Epochs All main experiments reuse the initial-policy profile and assignments for both epochs. Appendix J compares this with re-profiling after epoch 1: naive re-profiling performs substantially worse ($\sim$25% versus $\sim$32% AIME24 accuracy), motivating static assignments over the evaluated horizon. We leave dynamic reassessment over longer horizons to future work.

5 Conclusion

We introduced DeliberateRL, a framework for shaping the *practice distribution* in RLVR through competence-aware query selection, rollout allocation, and curriculum ordering. Its sGPO instantiation uses a single offline profile to match the accuracy of uniform and online-adaptive baselines while using at least $2.5\times$ less total compute than uniform allocation over the evaluated two-epoch horizon. These results highlight compute allocation as an important design axis for efficient RLVR.

Limitations First, sGPO targets RLVR with objective, binary rewards, which makes it difficult to apply to subjective preference alignment (RLHF). Second, it relies on competence estimates from a single initial-policy profile, which become stale as the policy changes. Our experiments support reusing the original assignments over two epochs, but not over substantially longer runs, and naive epoch-boundary re-profiling underperforms stale replay. Designing adaptive reassessment strategies that preserve useful difficulty diversity, including adaptive schedules for revisiting hard queries, remains an open problem.

AI USE STATEMENT

We used generative AI tools to assist with editing draft text for clarity, formatting LaTeX tables, and suggesting boilerplate code for data-processing scripts. We did not use generative AI to design the sGPO allocation rule, or produce reported experimental results.

ETHICS STATEMENT

Reducing estimated training FLOPs can lower computational resource requirements, although actual energy use and carbon emissions depend on hardware, utilization, and datacenter conditions. Lower compute requirements may also make RLVR experiments more accessible to groups with limited hardware budgets. 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

Hadi Amiri, Timothy A Miller, and Guergana K Savova. Repeat before forgetting: Spaced repetition for efficient and effective training of neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2401–2410, 2017.

Reyhane Askari-Hemmat, Mohammad Pezeshki, Elvis Dohmatob, Florian Bordes, Pietro Astolfi, Melissa Hall, Jakob Verbeek, Michal Drozdzal, and Adriana Romero-Soriano. Improving the scaling laws of synthetic data with deliberate practice. *arXiv preprint arXiv:2502.15588*, 2025.

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

Udbhav Bamba, Minghao Fang, Yifan Yu, Haizhong Zheng, and Fan Lai. Xrpo: Pushing the limits of grpo with targeted exploration and exploitation. *arXiv preprint arXiv:2510.06672*, 2025.

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

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

Guillermo Campitelli and Fernand Gobet. Deliberate practice: Necessary but not sufficient. *Current directions in psychological science*, 20(5):280–285, 2011.

Nicholas J Cepeda, Harold Pashler, Edward Vul, John T Wixted, and Doug Rohrer. Distributed practice in verbal recall tasks: A review and quantitative synthesis. *Psychological bulletin*, 132(3): 354, 2006.

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

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

Hermann Ebbinghaus. A contribution to experimental psychology. *New York, NY: Teachers College, Columbia University*, 1913.

K Anders Ericsson. Deliberate practice and the acquisition and maintenance of expert performance in medicine and related domains. *Academic medicine*, 79(Supplement_2):S70–S81, 2004.

K Anders Ericsson, Ralf T Krampe, and Clemens Tesch-Römer. The role of deliberate practice in the acquisition of expert performance. *Psychological review*, 100(3):363, 1993.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, et al. Skywork open reasoner 1 technical report. *arXiv preprint arXiv:2505.22312*, 2025.

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

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

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

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

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

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.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Limr: Less is more for rl scaling. *arXiv preprint arXiv:2502.11886*, 2025a.

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

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

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

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

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

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

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

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

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.

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

Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Tanglifu Tanglifu, Xiaowei Lv, et al. 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)*, pp. 318–327, 2025.

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

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

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

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

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Enci Zhang, Xingang Yan, Wei Lin, Tianxiang Zhang, and Lu Qianchun. 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*, pp. 6630–6644, 2025a.

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

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

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

Yiming Zong, Yige Wang, and Jiashuo Jiang. Cross-epoch adaptive rollout optimization for rl post-training. *arXiv preprint arXiv:2606.05606*, 2026.

CONTENTS

1 Introduction **1**

2 Background **3**

3 Method **5**

3.1 Offline Competence Profiling via Inference-Time Scaling 5

3.2 Competence-Aware Compute Allocation 5

3.3 Training Strategies from $\hat{p}$ 6

3.3.1 What to Practice: Query Selection 6

3.3.2 How Much to Practice: Rollout Allocation 6

3.3.3 When to Practice: Curriculum Ordering 7

4 Experiments **7**

4.1 Setup 7

4.2 Main Results 7

4.3 Ablations and Analysis 9

5 Conclusion **9**

A Related Work **15**

B Derivation of the Inverse-Success Rate Heuristic **16**

B.1 Discrete Allocation Rule 16

B.2 Continuous Relaxation and Inverse-Success Approximation 17

B.3 Expected-Success Interpretation and Practical Allocation 17

B.4 Learning Signal versus Rollout Cost 18

C Optimization Objective **20**

D Algorithm **21**

E Additional Experimental Analysis **22**

E.1 Inference–Training Compute Asymmetry 22

E.2 Offline versus Online Allocation 22

E.3 Efficiency Decomposition 22

F Learning Dynamics **23**

G Uniform Group Scaling **24**

H Robustness and Generalization Experiments **25**

H.1 Sensitivity to Profiling Budget 25

H.2 Sensitivity to the High-Success Threshold 25
H.3 Cross-Family Evaluation with Llama 25
H.4 Post-Cold-Start Training Cost 26

I Wall-Clock Time and Token Counts **27**

J Multi-Epoch Training **28**

K Self-Consistency vs. Verified Profiling **30**

L Training Dynamics **31**

M Experimental Details **32**
M.1 Datasets and Models 32
M.2 Prompt 32
M.3 Training and Evaluation Setup 33
M.4 Optimizer Settings Across Curriculum Phases 33
M.5 FLOP Accounting 33
M.6 Hyperparameters 34

A Related Work

Rollout Allocation Methods for non-uniform compute allocation in RLVR differ mainly in where their estimate of prompt utility comes from and which allocation decision it controls (Table 1). Most rely on *training-time reward history*. GRESO (Zheng et al., 2026) skips prompts whose recent groups had zero reward variance, and DEPO (Zhao et al., 2026) combines difficulty-based selection with diversity sampling; both decide *what* to train on. Knapsack RL (Li et al., 2025b) instead decides *how much*, solving a knapsack problem over per-prompt success statistics, and CERO (Zong et al., 2026) maintains Beta posteriors over prompt success and couples allocations across epochs under a global rollout budget. AR3PO (Zhang et al., 2025b) adapts rollout counts and reuses earlier responses, while GDRO (Panaganti et al., 2026) casts allocation as group distributionally robust optimization. A second line *predicts* competence: VIP (Nguyen et al., 2026) fits a lightweight model of per-prompt success to minimize expected gradient variance under a fixed budget, and CoBA-RL (Yao et al., 2026) uses a probabilistic capability model. XRPO (Bamba et al., 2025) treats allocation as an exploration–exploitation problem, combining targeted allocation with in-context seeding and novelty-aware advantage sharpening.

These online methods can update their allocations as competence changes, and predictive models may generalize beyond prompts already observed. sGPO studies a complementary trade-off: it computes a verified competence profile once before optimization and uses it to fix query selection, group sizing, and curriculum ordering over a short training horizon. All three decisions are therefore available from the first RL step without maintaining an online allocator.

Offline Difficulty Filtering Filtering training data by the pass rate of a reference model before RL is common in recent reasoning pipelines, typically to remove prompts that are always or never solved (Team et al., 2025; He et al., 2025), and LIMR (Li et al., 2025a) selects a compact training subset offline. These works use the offline estimate for *selection* only. sGPO reuses the same profile to also set per-query rollout budgets and the curriculum order, and treats the profiling cost as part of the total compute budget.

Curriculum Learning for Reasoning Curriculum learning (Bengio et al., 2009) schedules training 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) builds self-evolving curricula from advantage proxies, VCRL (Jiang et al., 2025) prioritizes samples by reward variance, TACLer (Lai &Nissim, 2026) and ADCL (Zhang et al., 2025a) re-estimate difficulty during training, and Light-R1 (Wen et al., 2025) stages a curriculum across SFT, DPO, and RL. These methods mainly design the ordering itself; in sGPO, the ordering follows directly from $\hat{p}(q)$, the same signal that sets filtering and group sizes.

Deliberate Practice and Spaced Repetition Deliberate practice structures training around the learner's current ability, with feedback guiding repeated attempts (Ericsson et al., 1993); spaced repetition revisits difficult items at intervals rather than all at once (Ebbinghaus, 1913; Cepeda et al., 2006), and has been used to schedule training examples for neural networks (Amiri et al., 2017). Deliberate practice has also motivated efficient synthetic-data generation for diffusion models (Askari-Hemmat et al., 2025). DeliberateRL borrows these ideas only at the level of allocating training effort: prompt-level success estimates and verifier feedback replace the individualized diagnosis, corrective instruction, and continual reassessment of human deliberate practice. We use them as design motivation, not as a claim to reproduce the full instructional process.

B Derivation of the Inverse-Success Rate Heuristic

We motivate the inverse-success-rate allocation rule by considering the probability that a rollout group contains exactly one successful response. Suppose each rollout independently succeeds with probability $p \in (0, 1)$ and a group contains G rollouts. Then

$$n \sim \text{Binomial}(G, p), \tag{16}$$

and the probability of observing exactly one success is

$$f_p(G) := \Pr(n = 1 \mid G, p) = Gp(1-p)^{G-1}. \tag{17}$$

We first maximize this probability over integer group sizes, then relate the resulting allocation to its continuous relaxation. Throughout, practical rollout groups satisfy $G \geq 2$, since $G = 1$ cannot provide a within-group reward comparison.

B.1 Discrete Allocation Rule

For successive integer group sizes, the ratio of singleton-success probabilities is

$$\frac{f_p(G+1)}{f_p(G)} = \frac{(G+1)p(1-p)^G}{Gp(1-p)^{G-1}} = \frac{G+1}{G}(1-p). \tag{18}$$

Consequently,

$$\frac{f_p(G+1)}{f_p(G)} \begin{cases} > 1, & G < 1/p - 1, \\ = 1, & G = 1/p - 1, \\ < 1, & G > 1/p - 1. \end{cases} \tag{19}$$

Thus, $f_p(G)$ increases up to its maximizing integer group size or sizes and decreases thereafter.

Over positive integers, if $1/p$ is not an integer, the unique maximizer is $\lfloor 1/p \rfloor$. If $1/p = m$ is an integer, then $m \geq 2$ and the two maximizers are $m - 1$ and m. Imposing the practical constraint $G \geq 2$, one global maximizer is therefore

$$G^\star_{\text{int}}(p) = \max\left\{2, \left\lfloor \frac{1}{p} \right\rfloor\right\}. \tag{20}$$

The constrained maximizer is unique unless $1/p = m$ is an integer with $m \geq 3$, in which case both $m - 1$ and m remain feasible and attain the maximum. At $p = 1/2$, only $G = 2$ remains feasible among the two unconstrained maximizers.

Equation (20) establishes an exact integer optimum for singleton-success probability. It does not require a low-p approximation. The endpoints $p = 0$ and $p = 1$ are excluded from the derivation: at either endpoint, every group with $G \geq 2$ has identical binary rewards and provides no group-relative reward signal.

Table 6 illustrates the transition identified by the ratio test. For $p = 0.30$, $R_p(2) > 1$ and $R_p(3) < 1$, giving the unique maximizer $G = 3 = \lfloor 1/p \rfloor$. For $p = 0.25$, $R_p(3) = 1$, so $G = 3$ and $G = 4$ tie; the rule selects $G = \lfloor 1/p \rfloor = 4$.

Table 6: Numerical illustration of discrete singleton-success maximization. Here $f_p(G) = Gp(1-p)^{G-1}$ and $R_p(G) = f_p(G+1)/f_p(G)$ measures the effect of increasing group size from G to $G+1$. Ratios above one indicate an increase, equality indicates a tie, and ratios below one indicate a decrease. Bold entries are global maxima over integer $G \geq 2$, before imposing the batching restriction $\{2, 4, 8\}$. Values are rounded to four decimals.

	$p = 0.30,\ 1/p \approx 3.33$		$p = 0.25,\ 1/p = 4$	
G	$f_p(G)$	$R_p(G)$	$f_p(G)$	$R_p(G)$
2	0.4200	1.0500	0.3750	1.1250
3	**0.4410**	0.9333	**0.4219**	1.0000
4	0.4116	0.8750	**0.4219**	0.9375
5	0.3602	0.8400	0.3955	0.9000

B.2 Continuous Relaxation and Inverse-Success Approximation

For comparison, extend the expression $f_p(G)$ to real $G > 0$. Because the logarithm is monotone, maximizing this extension is equivalent to maximizing

$$\log f_p(G) = \log G + \log p + (G-1)\log(1-p). \tag{21}$$

Differentiating with respect to G gives

$$\frac{\partial}{\partial G}\log f_p(G) = \frac{1}{G} + \log(1-p). \tag{22}$$

Setting the derivative to zero yields

$$G^\star_{\text{cont}}(p) = -\frac{1}{\log(1-p)} = \frac{1}{-\log(1-p)}. \tag{23}$$

Since

$$\frac{\partial^2}{\partial G^2}\log f_p(G) = -\frac{1}{G^2} < 0, \tag{24}$$

Equation (23) is the unique maximizer of the extension over real $G > 0$. If the continuous relaxation also imposes $G \geq 2$, its maximizer is

$$G^\star_{\text{cont},\,\geq 2}(p) = \max\left\{2, \frac{1}{-\log(1-p)}\right\}. \tag{25}$$

For low-success queries,

$$-\log(1-p) = p + \frac{p^2}{2} + \frac{p^3}{3} + \mathcal{O}(p^4). \tag{26}$$

Substituting this expansion into Equation (23) gives

$$G^\star_{\text{cont}}(p) = \frac{1}{p} - \frac{1}{2} - \frac{p}{12} + \mathcal{O}(p^2), \tag{27}$$

and hence

$$G^\star_{\text{cont}}(p) \sim \frac{1}{p} \qquad \text{as } p \to 0. \tag{28}$$

The inverse-success operating point therefore agrees asymptotically with the continuous optimum. For integer group sizes, however, Equation (20) already gives an exact result, including the minimum group-size constraint and the tie cases.

B.3 Expected-Success Interpretation and Practical Allocation

Under the Binomial model, the expected number of successful rollouts is

$$\mathbb{E}[n] = Gp. \tag{29}$$

Thus, the inverse-success operating point $G \approx 1/p$ corresponds to roughly one expected success per group. More precisely, for $0 < p \leq 1/2$, the particular integer maximizer in Equation (20) satisfies

$$1 - p < p\, G^\star_{\text{int}}(p) \leq 1. \tag{30}$$

Its expected number of successes therefore approaches one as $p \to 0$. This is an interpretation of the singleton-success optimum, not a guarantee that every sampled group contains exactly one success.

Allocation from an Estimated Success Probability Because the true success probability is unknown, sGPO uses the offline estimate $\hat{p}(q)$. For a query with $\hat{p}(q) > 0$, the corresponding integer operating point is

$$\widehat{G}_{\text{int}}(q) = \max\left\{2, \left\lfloor \frac{1}{\hat{p}(q)} \right\rfloor\right\}. \tag{31}$$

Equation (31) provides an analytical reference for allocation. The implementation directly applies the bucketed rule $b(q)$ in Section 3.3.2, restricting group sizes to $\mathcal{G} = \{2, 4, 8\}$ for efficient batched training. The following analysis establishes when this implemented rule agrees with singleton-success optimization.

Exact Correspondence on the Default Profiling Grid With the default profiling budget $N = 8$ and filtering threshold $t = 0.75$, the positive retained estimates are

$$\hat{p}(q) \in \left\{\frac{1}{8}, \frac{2}{8}, \frac{3}{8}, \frac{4}{8}, \frac{5}{8}, \frac{6}{8}\right\}. \tag{32}$$

Table 7: Singleton-success optima at the positive retained values of the default $N = 8$ profiling grid. The continuous column reports the optimum over real $G > 0$, before imposing $G \geq 2$. Integer maximizers respect $G \geq 2$. At every listed value, the implemented group size is a global integer maximizer and therefore also maximizes singleton-success probability over $\mathcal{G} = \{2, 4, 8\}$. For practical allocation, p is replaced by the offline estimate $\hat{p}(q)$.

p	$G^{\star}_{\mathrm{cont}}(p)$	Integer maximizers ($G \geq 2$)	sGPO group size
$1/8$	7.49	$\{7, 8\}$	8
$1/4$	3.48	$\{3, 4\}$	4
$3/8$	2.13	$\{2\}$	2
$1/2$	1.44	$\{2\}$	2
$5/8$	1.02	$\{2\}$	2
$3/4$	0.72	$\{2\}$	2

At each of these values, the implemented allocation maximizes singleton-success probability over the feasible set $\mathcal{G}$:

$$b(q) \in \arg\max_{G \in \mathcal{G}} G\, \hat{p}(q)\big(1 - \hat{p}(q)\big)^{G-1}. \tag{33}$$

Table 7 compares the continuous optimum, the integer maximizers, and the implemented group sizes on this grid.

Restricted-set Optima Beyond the Default Grid For arbitrary p, the exact maximizing choice within $\mathcal{G}$ can be obtained by comparing adjacent feasible sizes. For $g \in \{2, 4\}$,

$$\frac{f_p(2g)}{f_p(g)} = 2(1-p)^g. \tag{34}$$

Define the two transition probabilities

$$p_{8,4} = 1 - 2^{-1/4} \approx 0.1591, \qquad p_{4,2} = 1 - 2^{-1/2} \approx 0.2929. \tag{35}$$

One maximizing allocation over $\mathcal{G}$ is

$$G^{\star}_{\mathcal{G}}(p) = \begin{cases} 8, & 0 < p \leq p_{8,4}, \\ 4, & p_{8,4} < p \leq p_{4,2}, \\ 2, & p_{4,2} < p < 1. \end{cases} \tag{36}$$

At $p = p_{8,4}$, sizes 8 and 4 tie; at $p = p_{4,2}$, sizes 4 and 2 tie.

The implemented bucket boundaries, $1/8$ and $1/4$, agree with Equation (36) on the default retained profiling grid. They need not agree on finer grids. For example, at $\hat{p} = 5/32$, the implemented rule assigns $G = 4$, whereas the singleton-success optimum over $\mathcal{G}$ is $G = 8$. Accordingly, the fixed buckets used in the higher-budget profiling experiments remain a coarse allocation heuristic; the exact correspondence in Equation (33) is specific to the default grid.

Queries with $\hat{p}(q) = 0$ are handled separately by the zero-observed-success retention rule. Their allocation is not derived by substituting zero into an inverse-success expression. Moreover, zero observed successes in a finite profile do not imply that the true success probability is zero.

B.4 Learning Signal versus Rollout Cost

The singleton-success criterion selects a finite group size with an interpretable reward configuration. It is distinct from maximizing the probability of obtaining any non-degenerate group:

$$\Pr(0 < n < G \mid G, p) = 1 - (1-p)^G - p^G. \tag{37}$$

For $p \in (0, 1)$, this mixed-group probability is strictly increasing in G. Maximizing it without a budget constraint therefore does not yield the finite operating point derived above.

Larger groups also do not make a low-success query's empirical success fraction approach one half. Under the Binomial model,

$$\mathbb{E}\left[\frac{n}{G}\right] = p, \qquad \mathrm{Var}\left(\frac{n}{G}\right) = \frac{p(1-p)}{G}. \tag{38}$$

As G increases, the empirical success fraction concentrates around p. Thus, for small p, additional rollouts increase the chance of obtaining a mixed group without making its success fraction typically balanced.

The singleton-success objective is also not a cost-normalized objective. Under the simplifying assumption of equal cost per rollout, singleton-success probability per rollout is

$$\frac{f_p(G)}{G} = p(1-p)^{G-1}, \tag{39}$$

which decreases with G for $p \in (0, 1)$. Consequently, maximizing $f_p(G)$ should not be interpreted as maximizing singleton-success probability per unit compute. Actual learning progress additionally depends on the sampled trajectories, their policy gradients, response lengths, and the optimization procedure.

Finally, the derivation assumes independent binary outcomes with a fixed success probability. In practice, $\hat{p}(q)$ is estimated from finitely many samples under the initial policy and its profiling generation settings. Its relationship to training-time success can change as the policy or sampling distribution changes. The exact optimization statements therefore apply to the stated probability model and, when using $\hat{p}(q)$, to the corresponding plug-in surrogate.

Accordingly, this analysis supplies a principled motivation for competence-aware rollout allocation, but does not establish optimality for aggregate policy-gradient magnitude, expected learning progress, or learning progress per FLOP.

C Optimization Objective

sGPO retains the verifiable reward function and the underlying DAPO policy-update rule while changing the allocation of training queries and rollout compute. Following the background formulation, we distinguish the phase-wise expected-reward objective from the group-relative surrogate used to construct practical policy updates.

Phase-wise Expected Reward For curriculum phase $g \in \{2, 4, 8\}$, let $\bar{C}_g = C_g \cup \widetilde{D}_0$ denote the augmented training cluster. The cluster assignments are determined by the offline profile and remain fixed during optimization. Every query processed in phase g receives a rollout group of size g.

Under a uniform distribution over queries in $\bar{C}_g$, the phase-wise expected-reward objective is

$$J_g(\theta) = \frac{1}{|\bar{C}_g|} \sum\nolimits_{q_j \in \bar{C}_g} \sum\nolimits_{\tau} p_\theta(\tau \mid q_j)\, r(\tau, q_j). \tag{40}$$

This has the same reward-maximization form as the dataset-level objective in the background, with the query distribution restricted to the current cluster. Thus, query selection changes which examples contribute to the objective, while the reward assigned to a given query–response pair remains unchanged.

Because the verifier reward does not depend explicitly on θ, applying the score-function identity gives

$$\nabla_\theta J_g(\theta) = \frac{1}{|\bar{C}_g|} \sum\nolimits_{q_j \in \bar{C}_g} \sum\nolimits_{\tau} p_\theta(\tau \mid q_j)\, r(\tau, q_j) \nabla_\theta \log p_\theta(\tau \mid q_j). \tag{41}$$

This is the exact gradient of the phase-wise expected-reward objective. Group size determines the sampling budget used to construct an update; it does not appear in the expected-reward objective itself.

Group-relative Policy Update In practice, a frozen behavior policy $p_{\theta_{\text{old}}}$ generates g rollouts for each sampled query:

$$\{\tau_{j,i}\}_{i=1}^{g} \sim p_{\theta_{\text{old}}}(\cdot \mid q_j), \qquad r_{j,i} = r(\tau_{j,i}, q_j). \tag{42}$$

As in Equation (1), rewards are normalized within each query's rollout group. Writing

$$\bar{r}_j = \frac{1}{g} \sum_{i=1}^{g} r_{j,i}, \qquad \sigma_j^2 = \frac{1}{g} \sum_{i=1}^{g} (r_{j,i} - \bar{r}_j)^2, \tag{43}$$

the group-relative advantage is $a_{j,i} = (r_{j,i} - \bar{r}_j)/\sigma_j$ for $\sigma_j > 0$. For the schematic update below, we set $a_{j,i} = 0$ when $\sigma_j = 0$: such groups provide no relative reward signal, although their generation cost has already been incurred.

Treating the sampled advantages as fixed during differentiation, the unclipped, sequence-level update evaluated at the behavior policy has the form

$$\hat{\mathcal{F}}_g(\theta_{\text{old}}) = \frac{1}{|\bar{C}_g|\, g} \sum\nolimits_{q_j \in \bar{C}_g} \sum\nolimits_{i=1}^{g} a_{j,i}\, \nabla_\theta \log p_\theta(\tau_{j,i} \mid q_j)|_{\theta=\theta_{\text{old}}}. \tag{44}$$

The full-cluster notation follows the background; practical updates use minibatches of sampled queries. The group size is written as g throughout because it is shared by all queries within a phase.

Equation (44) describes a group-relative surrogate update. Since its advantages use a mean and standard deviation estimated from the same sampled group, it is not generally an unbiased Monte Carlo estimator of Equation (41).

Connection to the DAPO Implementation We instantiate sGPO on top of DAPO (Yu et al., 2026). Equation (44) expresses the sequence-level structure used in the background; the implementation retains DAPO's token-level importance weighting, asymmetric clipping, and loss normalization. The sampled advantages and behavior-policy probabilities are held fixed while the policy parameters are updated.

The three allocation decisions affect how this update is constructed: query selection determines the available training clusters, rollout allocation determines the number of responses sampled per query, and curriculum ordering determines the sequence of phase-wise updates. The singleton-success criterion in Section 3.2 guides rollout allocation; it does not replace the verifier reward or introduce an additional term into the DAPO loss. Accordingly, sGPO changes the training distribution and sampling procedure while preserving the underlying reward and policy-update rule.

D ALGORITHM

Algorithm 1 sorted-Group Policy Optimization (sGPO)

Require: Training dataset $\mathcal{D}$; initial policy p_{θ_0}; profiling budget $N = 8$; group sizes $\mathcal{G} = \{2, 4, 8\}$; filtering threshold $t = 0.75$; zero-observed-success retention rate $\alpha = 0.1$

Offline competence profiling

for each query $q \in \mathcal{D}$ **do**
  $\{\tau_i\}_{i=1}^{N} \sim p_{\theta_0}(\tau \mid q)$ // Generate N profiling rollouts
  $r_i \leftarrow \text{Verify}(\tau_i, q) \in \{0, 1\}, i = 1, \ldots, N$
  $\hat{p}(q) \leftarrow \frac{1}{N} \sum_{i=1}^{N} r_i$ // Empirical success rate
end for

Query selection and rollout allocation

$\mathcal{D}_{\text{high}} \leftarrow \{q \in \mathcal{D} : \hat{p}(q) > t\}$ // Exclude from core training
$\mathcal{D}_0 \leftarrow \{q \in \mathcal{D} : \hat{p}(q) = 0\}$
$\mathcal{D}_{\text{learn}} \leftarrow \{q \in \mathcal{D} : 0 < \hat{p}(q) \leq t\}$
Partition $\mathcal{D}_{\text{learn}}$ into C_2, C_4, C_8 using $b(q)$ from Eq. (13)

Zero-observed-success retention

$\tilde{\mathcal{D}}_0 \leftarrow \text{Subsample}(\mathcal{D}_0, \alpha)$ // Sample once and reuse across phases

Easy-to-hard curriculum training

$\theta \leftarrow \theta_0$
for $G \in \mathcal{G}$ in ascending order **do**
  $\bar{C}_G \leftarrow C_G \cup \tilde{\mathcal{D}}_0$
  Train p_θ on $\bar{C}_G$ using rollout-group size G
end for
return optimized policy p_θ

E Additional Experimental Analysis

This section analyzes where the compute savings of sGPO arise and how offline profiling differs from online allocation. All numbers refer to the Qwen2.5-Math-7B experiments in Table 2 and Table 4; the behavior of the fixed profile as the policy improves is studied in Appendix J.

E.1 Inference–Training Compute Asymmetry

Under our FLOP accounting (Appendix M.5), a profiling token costs about $2P$ FLOPs and an RL training token about $12P$, so profiling is roughly $6\times$ cheaper per token. At Qwen2.5-Math-7B scale (Table 2; Figure 2a), sGPO spends 1.6 EF on its one-time profiling pass and 7.3 EF on RL training, for 8.9 EF total, compared with 22.3 EF for DAPO. Training compute alone falls by $3.1\times$, and total compute by $2.5\times$ after including profiling.

The benefit of profiling is not only its cost but also its timing. The competence estimate $\hat{p}(q)$ (Eq. (5)) is available before the first policy update and determines all three allocation decisions used by sGPO: which queries enter the core training set, how much rollout compute they receive, and when they are presented. sGPO therefore spends inference compute upfront to reduce the more expensive training-time computation performed later.

E.2 Offline versus Online Allocation

GRESO and Knapsack RL obtain competence information from training-time rewards rather than from a pre-training profile. GRESO uses reward history to skip queries likely to produce zero-variance groups, but retains a uniform group size $G = 8$ and random ordering. Knapsack RL adapts the rollout budget using per-query success statistics accumulated during training; without a pre-training estimate, its allocation is uniform until that history becomes available.

At Qwen2.5-Math-7B scale (Table 2), GRESO and Knapsack RL use 21.9 EF and 20.1 EF total compute, respectively, compared with 8.9 EF for sGPO, with average accuracies of 11.9%, 13.1%, and 13.8%. The comparison highlights two practical differences of offline profiling. First, allocation decisions are available from the first RL update rather than only after training-time observations have accumulated. Second, a single competence estimate jointly controls query selection, rollout allocation, and curriculum ordering (Table 1). Together, these let sGPO save compute from the start of training rather than only once an online allocator has collected enough statistics.

E.3 Efficiency Decomposition

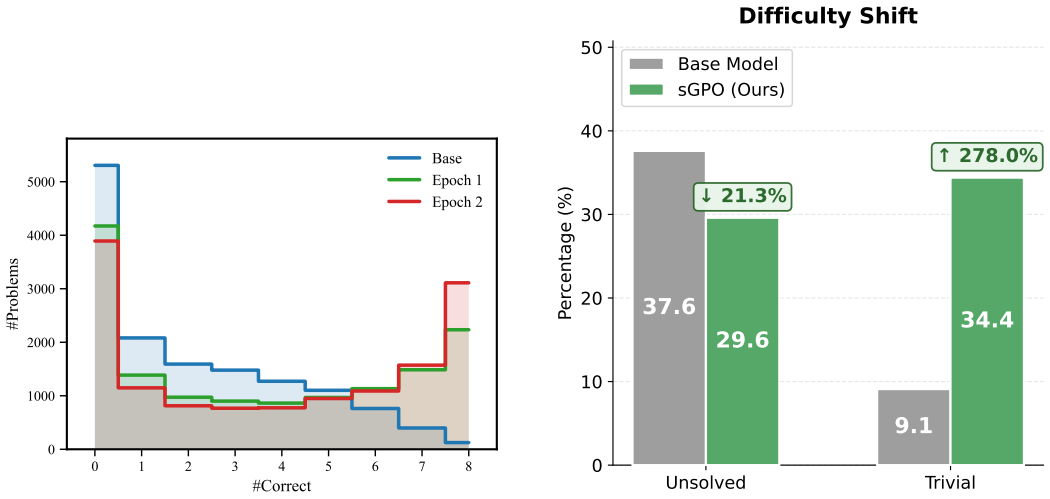
At Qwen2.5-Math-7B scale (Table 2), sGPO uses 7.3 EF for RL training and 1.6 EF for profiling, compared with 22.3 EF for DAPO. The training-only reduction is therefore 15.0 EF, and the net reduction after profiling is 13.4 EF.

The cumulative ablation in Table 4 traces the same reduction along the evaluated configuration path, on a total-compute basis that includes the 1.6 EF profiling cost. Adding competence-aware query selection with uniform $G = 8$ reduces compute from 22.3 EF (DAPO) to 10.7 EF, a difference of 11.6 EF. Adding adaptive rollout allocation reduces it further to 8.9 EF, a difference of 1.8 EF. Curriculum ordering leaves compute unchanged. These two differences sum to the 13.4 EF net reduction.

These sequential differences should not be interpreted as independent causal contributions of the components, because changing the configuration also changes training dynamics and generated-token counts.

F Learning Dynamics

To measure how policy competence changes during training, we re-profile the full training set with the epoch-1 checkpoint of sGPO on Qwen2.5-Math-7B, using the same protocol as the initial profile (N=8; Eq. (5)). Because the profiled queries are training queries, this measures how competence on the training distribution shifts, not generalization.



(a) Full per-bin distribution. **(b)** Zero-observed-success and high-success fractions.

Figure 5: Difficulty-distribution shift during training for sGPO using Qwen2.5-Math-7B. (a) Profiling distributions (number of correct samples out of N=8) for the base model and the epoch-1 checkpoint. (b) The two extreme categories: zero observed success ($\hat{p} = 0$) and high success ($\hat{p} > 0.75$). The zero-observed-success fraction decreases from 37.6% to 29.6% (-8.0 percentage points), while the high-success fraction increases from 9.1% to 34.4% ($+25.3$ percentage points).

Figure 5a shows the full per-bin distribution. The base model concentrates at the two extremes that yield no group-relative signal: 37.6% of queries have zero observed success ($\hat{p} = 0$) and 9.1% have high success ($\hat{p} > 0.75$). After one epoch, probability mass moves out of the zero-success bin toward intermediate and high-success bins. Figure 5b summarizes the endpoints: the high-success fraction grows by 25.3 percentage points (9.1% $\rightarrow$ 34.4%), and the zero-observed-success fraction shrinks by 8.0 percentage points (37.6% $\rightarrow$ 29.6%).

Many learnable queries become high-success The 25.3-point increase in high-success queries indicates substantial improvement on queries that were initially in $\mathcal{D}_{\text{learn}}$. Because sGPO keeps their original group sizes in epoch 2, these queries become more likely to produce all-correct groups.

Some unsolved queries become solvable The zero-observed-success fraction falls by 8.0 points, a net reduction of about 21% relative to its initial size. Two mechanisms may contribute: direct exposure through the retained subset $\tilde{\mathcal{D}}_0$, which is revisited in every phase, and transfer from capability built on easier queries. Consistent with the first mechanism, removing the retained subset (α=0%) lowers AIME24 accuracy by 2.7 points (Table 5), although this ablation does not isolate the two mechanisms.

Implication These shifts show that the initial profile becomes stale as training proceeds. Fixed rollout assignments become conservative as queries get easier, while some queries with no observed initial success become viable training examples. Appendix J tests whether refreshing the profile after epoch 1 helps. Naive re-profiling does not, which is why we reuse the original assignments over the two-epoch horizon.

G Uniform Group Scaling

Does increasing the uniform group size close the gap with adaptive allocation ? Figure 6 compares sGPO ($G \in \{2, 4, 8\}$, max group size 8) against DAPO with G=16 on Qwen2.5-Math-7B.

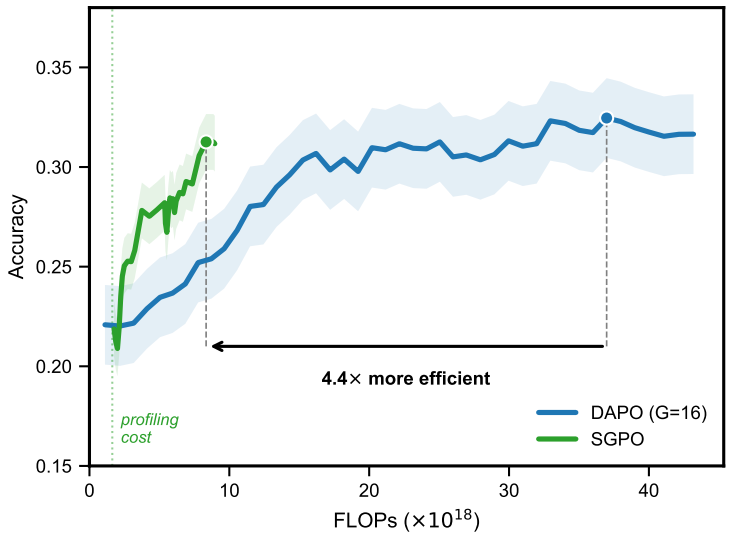


Figure 6: Accuracy vs. cumulative FLOPs for sGPO and DAPO (G=16) on Qwen2.5-Math-7B. sGPO reaches comparable peak accuracy 4.4× faster.

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.

H Robustness and Generalization Experiments

We conduct additional experiments to evaluate the robustness of sGPO to design choice variations, its transfer to a different model family, and its training cost after the initial information-acquisition stage. These experiments complement the main two-epoch results and use the same FLOP accounting described in Appendix M.5.

H.1 Sensitivity to Profiling Budget

The default profiling budget $N = 8$ trades estimation resolution against the cost of collecting the profile. To test sensitivity to this choice, we evaluate $N \in \{8, 16, 32\}$ on SciKnowEval with Qwen3-4B-Instruct.

As shown in Table 8, increasing N raises the upfront profiling cost but does not improve downstream performance in this setting. Among the tested values, $N = 8$ gives both the lowest total compute and the highest average and weighted-average accuracy. We therefore use $N = 8$ as an empirical default rather than claiming it is universally optimal.

Table 8: Sensitivity to profiling budget N on SciKnowEval using Qwen3-4B-Instruct. Accuracy values are avg@16 (%). Among the tested budgets, $N = 8$ gives the strongest accuracy–compute trade-off.

N	**Profile EF**	**Total EF**	**Avg**	**w Avg**
8	**0.4**	**1.9**	**58.0**	**62.5**
16	0.8	2.5	52.5	54.6
32	1.6	3.5	51.8	53.5

A larger profiling budget improves estimation precision but also changes the training distribution induced by the fixed selection and allocation rules. One possible mechanism is that larger budgets detect rare successes on very difficult queries, admitting them into the positive-success core even though the maximum rollout group remains $G = 8$.

For example, a query with true success probability $p = 0.02$ produces at least one profiling success with probability $1 - (1 - p)^N$, approximately 14.9% at $N = 8$ and 47.6% at $N = 32$. Yet its probability of producing a mixed training group at $G = 8$ is only approximately 14.9%. Thus, more precise profiling need not improve the allocation induced by an unchanged rollout cap.

This is a possible explanation rather than an established mechanism. The experiment does not isolate estimation precision from changes in query inclusion, bucket assignments, or training exposure.

H.2 Sensitivity to the High-Success Threshold

We also vary the high-success filtering threshold $t \in \{0.625, 0.75, 0.875\}$ while keeping the remaining sGPO configuration fixed. Table 9 shows that $t = 0.75$ performs best among the tested values on SciKnowEval. Both more aggressive filtering ($t = 0.625$) and more permissive filtering ($t = 0.875$) reduce accuracy.

These results support $t = 0.75$ as an empirical design choice for our experiments; the threshold is not derived from a theoretical optimum.

Table 9: Sensitivity to the high-success filtering threshold t on SciKnowEval using Qwen3-4B-Instruct. Scores are avg@16 accuracy (%). The default $t = 0.75$ is selected empirically from the tested values.

t	**Total EF**	**Avg**	**w Avg**
0.625	1.8	49.2	49.8
0.750	1.9	**58.0**	**62.5**
0.875	2.0	49.8	51.2

H.3 Cross-Family Evaluation with Llama

In this section we expand on the experiments for Llama-3.2-3B-Instruct using SciKnowEval.

Table 10 shows that sGPO reaches an average accuracy of 29.2% and weighted average of 30.1% using 1.5 EF in total, including 0.3 EF for profiling. This is comparable to the strongest online baselines in accuracy while requiring substantially less estimated compute. We view this result as an initial replication beyond the Qwen family rather than evidence of broad model-family generality.

Table 10: Cross-family evaluation on SciKnowEval using Llama-3.2-3B-Instruct. Scores are avg@16 accuracy (%). sGPO's 1.5 EF total includes 0.3 EF of one-time profiling compute. This experiment provides an initial non-Qwen replication rather than a claim of broad model-family generality.

Method	Total EF	Chemistry	Physics	Biology	Materials	Avg	w Avg
Base	—	16.4	27.6	23.6	24.1	22.9	21.0
DAPO	4.2	30.6	30.1	25.4	29.5	28.9	29.7
GRESO	3.6	30.8	**30.5**	**25.8**	29.7	**29.2**	29.9
Knapsack RL	3.8	30.3	29.7	25.2	29.2	28.6	29.4
sGPO	**1.5**	**31.1**	30.3	25.6	**29.9**	**29.2**	**30.1**

H.4 Post-Cold-Start Training Cost

A potential advantage of online allocation methods is that their difficulty estimates improve after training has already collected reward statistics. To separate this cold-start cost from later training cost, we compare the estimated FLOPs consumed during epoch 2 of Qwen2.5-Math-7B training.

Table 11 reports 11.15 EF for DAPO, 8.95 EF for Knapsack RL, and 3.65 EF for sGPO's epoch 2 training loop. This comparison excludes sGPO's one-time profiling cost and therefore isolates the cost of training after the initial profile has already been acquired.

The comparison is not fully matched in terms of information acquisition: sGPO obtains its competence estimates before training, whereas Knapsack RL acquires its allocation statistics from the preceding training history. We therefore use this experiment only to characterize post-cold-start training cost, not as a standalone end-to-end efficiency comparison.

Table 11: Estimated epoch-2 training FLOPs on Qwen2.5-Math-7B. For sGPO, this is training-only compute and excludes the one-time offline profiling pass. Because the methods acquire allocation information differently, this experiment isolates post-cold-start training cost rather than providing a fully matched end-to-end comparison.

Method	Epoch-2 Training EF
DAPO	11.15
Knapsack RL	8.95
sGPO	**3.65**

I Wall-Clock Time and Token Counts

Table 12 reports measured wall-clock time on a single 8×H100 node, split into one-time profiling and RL training, together with H100 GPU-hours (wall-clock minutes $\times\, 8/60$). Table 13 reports the number of processed tokens logged during profiling and training. The estimated FLOPs in Table 2 follow directly from these token counts as $2P\,T_{\text{profile}} + 12P\,T_{\text{train}}$, with P the nominal parameter count (1.5B or 7B; Appendix M).

Table 12: Wall-clock cost in H100 GPU-hours (measured on 8×H100; GPU-hours = wall-clock minutes $\times\, 8/60$). P, T, and Total denote profiling, RL training, and their sum. T/T_{DAPO} compares training GPU time with DAPO at the same model scale; $\text{Total}/\text{Total}_{\text{DAPO}}$ additionally includes profiling. Values below 1 indicate less GPU time than DAPO.

	Qwen2.5-Math-1.5B					*Qwen2.5-Math-7B*				
Method	P	T	Total	$\frac{T}{T_{\text{DAPO}}}$	$\frac{\text{Total}}{\text{Total}_{\text{DAPO}}}$	P	T	Total	$\frac{T}{T_{\text{DAPO}}}$	$\frac{\text{Total}}{\text{Total}_{\text{DAPO}}}$
DAPO	—	30.7	30.7	1.00	1.00	—	80.7	80.7	1.00	1.00
GRESO	—	47.5	47.5	1.55	1.55	—	66.8	66.8	0.83	0.83
Knapsack RL	—	38.4	38.4	1.25	1.25	—	90.8	90.8	1.13	1.13
sGPO	2.3	**6.1**	**8.4**	**0.20**	**0.27**	4.8	**21.9**	**26.7**	**0.27**	**0.33**

Table 13: Processed tokens (millions) during profiling (P) and RL training (T). T/T_{DAPO} compares training tokens with DAPO at the same model scale; $\text{Total}/\text{Total}_{\text{DAPO}}$ additionally includes profiling tokens. Values below 1 indicate fewer tokens than DAPO. Token counts are not directly comparable in cost: profiling tokens require only generation and are substantially cheaper per token than training tokens (Appendix M).

	Qwen2.5-Math-1.5B					*Qwen2.5-Math-7B*				
Method	P	T	Total	$\frac{T}{T_{\text{DAPO}}}$	$\frac{\text{Total}}{\text{Total}_{\text{DAPO}}}$	P	T	Total	$\frac{T}{T_{\text{DAPO}}}$	$\frac{\text{Total}}{\text{Total}_{\text{DAPO}}}$
DAPO	—	244	244	1.00	1.00	—	265	265	1.00	1.00
GRESO	—	239	239	0.98	0.98	—	261	261	0.98	0.98
Knapsack RL	—	261	261	1.07	1.07	—	239	239	0.90	0.90
sGPO	133	**61**	**194**	**0.25**	**0.80**	114	**87**	**201**	**0.33**	**0.76**

Profiling Tokens sGPO's profiling pass processes more tokens than its RL training loop (133M versus 61M at 1.5B; 114M versus 87M at 7B), yet accounts for only about 18% of total wall-clock time at 7B (36 of 200 minutes) and a similar share of estimated FLOPs (1.6 of 8.9 EF). This reflects the inference–training asymmetry motivating offline profiling: profiling tokens require only generation, whereas training tokens additionally incur policy-update computation. As a result, sGPO's RL training loop processes 3–4× fewer tokens than DAPO, and even including profiling it processes fewer tokens in total than every baseline (0.80 and 0.76 of DAPO's total at 1.5B and 7B). Because profiling tokens are cheaper, these token ratios understate the compute savings, which are better reflected by the FLOP and GPU-hour comparisons.

FLOPs versus Wall-Clock Wall-clock time is not strictly proportional to estimated FLOPs, and the ratio between them varies across methods. Several factors contribute. First, autoregressive generation is memory-bandwidth bound and each batch waits for its longest response, so long-tail response lengths add time without proportionally adding FLOPs. Second, methods incur scheduling overhead not captured by the FLOP estimates, such as resampling to fill batches under dynamic sampling, per-prompt skip bookkeeping in GRESO, and per-step allocation in Knapsack RL. Third, the $12P$-per-token estimate ignores the growth of attention cost with sequence length, which disproportionately affects methods that allocate more rollouts to hard queries with long responses. Finally, fixed per-step costs (e.g., weight synchronization between training and generation, reward verification) weigh more heavily at the 1.5B scale.

sGPO's measured wall-clock reduction relative to DAPO (3.7× at 1.5B and 3.0× at 7B) exceeds its estimated FLOP reduction (2.9× and 2.5×). One likely reason is that each curriculum phase groups queries of similar profiled difficulty, making response lengths within a batch more homogeneous and reducing idle time spent waiting for the longest response.

J Multi-Epoch Training

All main results in this paper use two epochs of sGPO training. This section describes the epoch 2 continuation strategies that motivated this choice.

We evaluate 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 the same query assignments and rollout-group sizes.
2. **Re-profiled:** Re-profile the dataset using the epoch 1 checkpoint to obtain updated $\hat{p}$ values, then recompute query selection, rollout-group assignments, and curriculum ordering.
3. **Re-profiled + fresh optimizer:** Same as (2), but reset the optimizer state before epoch 2.

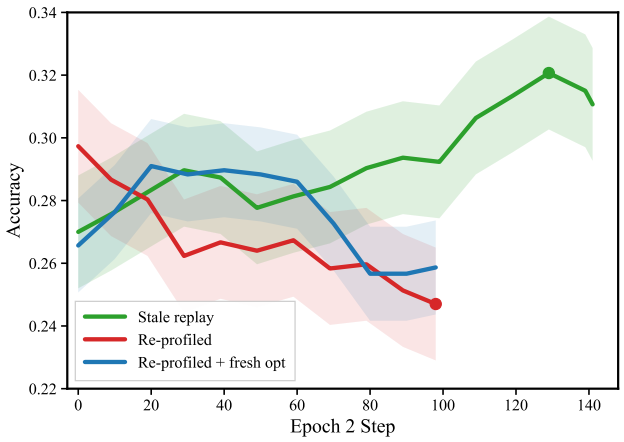


Figure 7: Epoch 2 accuracy on Qwen2.5-Math-7B under three continuation strategies. Stale replay (green) improves to $\sim$32%, whereas both re-profiled variants (red, blue) steadily degrade during epoch 2 to $\sim$25%.

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

The comparison therefore favors stale replay over the two re-profiled strategies in this setting. Resetting the optimizer does not recover the stale-replay trajectory: the two re-profiled variants behave similarly despite differing in optimizer state. This suggests that optimizer state alone is insufficient to explain the observed difference and is consistent with the change in training assignments playing the larger role.

Why does naive re-profiling underperform? Re-profiling changes query selection, rollout-group assignments, and curriculum ordering simultaneously. In particular, after epoch 1 many initially learnable queries have moved into the high-success regime (Appendix F) and are consequently removed by the renewed filtering rule. The re-profiled second epoch therefore places relatively more training effort on lower-success queries, whereas stale replay continues to expose the policy to a broader range of the initial competence distribution.

Importantly, this effect should not be interpreted as a failure of the expected-success allocation heuristic. Re-profiling does not imply that the expected number of successes becomes approximately zero for all retained queries. For positive-success queries, the refreshed allocation still targets roughly one expected success per group, subject to profiling error and the feasible rollout-group sizes. Rather, re-profiling changes *which* queries remain in the training distribution and how they are partitioned across curriculum phases.

Thus, the advantage of stale replay in our two-epoch setting is better understood as a practice-distribution effect. Renewed filtering and re-bucketing remove or reassign examples that stale replay continues to visit, substantially changing the composition of the second-epoch training distribution. The present experiment does not isolate the individual contributions of filtering, rollout allocation,

and curriculum ordering, and therefore does not establish which of these changes causes the observed performance degradation.

We consequently interpret this experiment as evidence about the specific re-profiling procedure evaluated here, rather than as evidence that stale assignments are generally optimal. Naively recomputing the same selection, allocation, and curriculum rules after each epoch can shift the practice distribution in a way that is detrimental over the evaluated two-epoch horizon. Designing dynamic reassessment strategies that adapt to changing policy competence while preserving useful training coverage remains an open direction.

K 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.

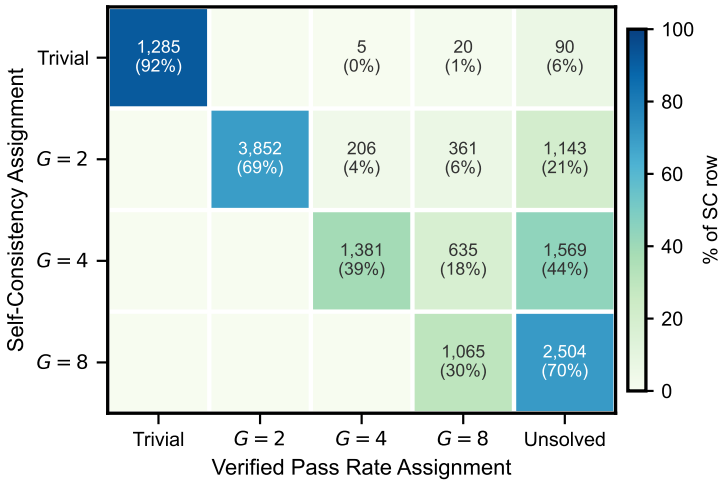


Figure 8: Confusion matrix between self-consistency (SC) and verified pass-rate difficulty assignments on DAPO-Math-17k (Qwen2.5-Math-7B, N=8). Each cell shows the count and row-normalized percentage. SC never classifies queries as unsolved (no "Unsolved" row exists) because it cannot distinguish consistent 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 substantial training compute to queries with no observed verified success. Such queries are more likely to produce all-incorrect groups and therefore no group-relative reward signal, although a finite zero-success profile does not imply that their true success probability is exactly zero. Verified profiling identifies these queries and either filters them or subsamples them at $\alpha = 10\%$.

L TRAINING DYNAMICS

Figure 9 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).

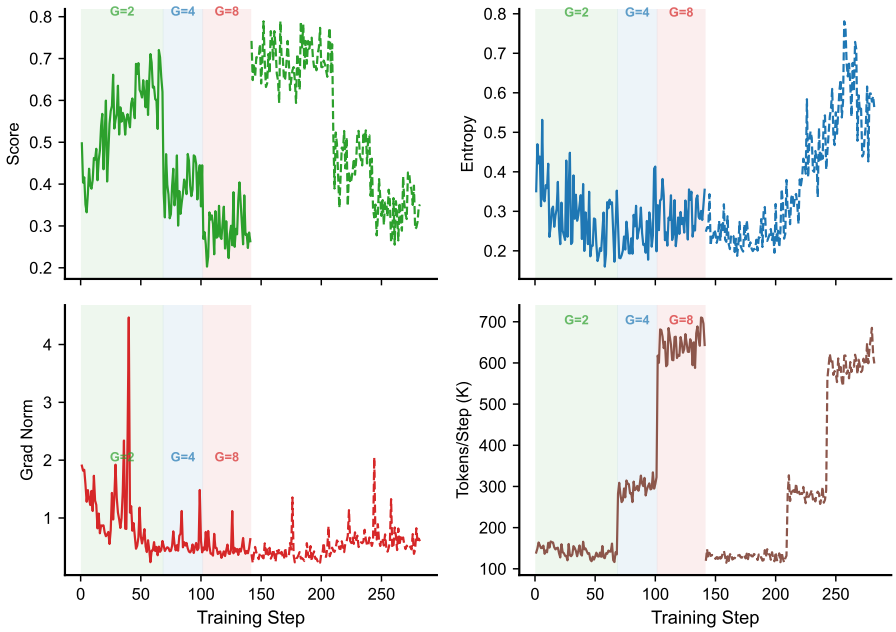


Figure 9: Training dynamics across curriculum phases for SGPO on Qwen2.5-Math-7B. Top-left: mean reward score, top-right: policy entropy, bottom-left: gradient norm, bottom-right: tokens per step. Solid: epoch 1, dashed: epoch 2. Phase bands indicate G=2 (green), G=4 (blue), G=8 (red).

Score Mean reward 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 the $G = 4$ phase at 0.45. The $G = 8$ phase, which contains the $\hat{p} = 1/8$ core cluster together with the retained $\widetilde{\mathcal{D}}_0$ subset, peaks at 0.40 but ends at 0.26, close to its starting value of 0.27. This flat trajectory suggests limited within-phase improvement at this scale. The current component ablations do not isolate the causal contribution of the $G = 8$ phase. In epoch 2, the $G = 2$ phase starts at 0.74, compared with 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 ($0.32 \rightarrow 0.36$ in epoch 1 and $0.45 \rightarrow 0.56$ in epoch 2), coinciding with the hardest phase, which combines positive-success core queries with the retained zero-observed-success subset.

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.

M Experimental Details

M.1 Datasets and Models

Tables 14, 15, and 16 summarize the datasets, benchmarks, and models used in this work.

Table 14: 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.

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

Table 15: Evaluation benchmarks. SciKnowEval counts reflect the 10% test split.

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

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`
Llama-3.2-3B-Instruct (Grattafiori et al., 2024)	3B	`meta-llama/Llama-3.2-3B-Instruct`

M.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.

M.3 Training and Evaluation Setup

All experiments use DAPO (Yu et al., 2026) as the underlying RL optimizer and are run on $8\times$ H100 GPUs. sGPO modifies only the allocation of training data and rollout compute by adding offline profiling, query selection, adaptive rollout-group sizing, and curriculum ordering. All other optimizer settings, including the learning rate, KL regularization, and clipping, follow the DAPO setup.

Unless otherwise stated, we use a profiling budget of $N = 8$ verified samples per query from the initial policy. The resulting empirical success rate $\hat{p}(q)$ partitions the dataset into high-success queries with $\hat{p}(q) > 0.75$, zero-observed-success queries with $\hat{p}(q) = 0$, and a core training set with $0 < \hat{p}(q) \leq 0.75$. Queries in the core set are assigned rollout group sizes $G \in \{2, 4, 8\}$ according to the discretized allocation rule in Eq. 13. For each curriculum phase, we additionally mix an $\alpha = 10\%$ subsample of the zero-observed-success set and train sequentially over $\bar{C}_2 \to \bar{C}_4 \to \bar{C}_8$.

The profiling estimates and resulting assignments are computed once before training and reused across both epochs in the main experiments. We ablate the profiling budget, filtering threshold, mixing ratio, and re-profiling strategy separately.

Evaluation uses **avg@16** accuracy. For each evaluation query, we generate 16 independent responses with temperature 1.0 and top-$p = 0.7$, verify each response independently, and report the mean fraction of correct responses across queries.

M.4 Optimizer Settings Across Curriculum Phases

The learning rate, KL coefficient, clipping parameters, optimizer state, gradient-accumulation schedule, and loss-normalization rule are held fixed across the $g \in \{2, 4, 8\}$ curriculum phases. We do not rescale the learning rate as a function of rollout-group size. Consequently, the number of processed tokens per optimization step varies across phases.

M.5 FLOP Accounting

We report estimated compute in ExaFLOPs (EF; 10^{18} FLOPs). Let P denote the number of model parameters. Following standard transformer compute approximations (Kaplan et al., 2020; Hoffmann et al., 2022), we approximate a forward pass as $\sim 2P$ FLOPs per processed token. All reported compute values use the same accounting assumptions across methods.

Profiling cost The offline profiling pass performs autoregressive inference only and does not require gradient computation. We therefore estimate its cost as

$$\mathrm{FLOPs}_{\mathrm{profile}} = 2P\, T_{\mathrm{profile}},$$

where T_{profile} is the total number of tokens processed during the $N = 8$ profiling samples over the training set. Profiling cost is included in all reported sGPO compute totals.

Training cost A GRPO-style training step includes rollout generation together with the forward and backward computations required for the policy update. For the DAPO implementation used in our experiments, a simplified per-token decomposition is

1. **Rollout generation** (actor forward pass): $2P$;
2. **Reference log-probabilities** (reference-model forward pass): $2P$;
3. **Actor update** (forward + backward): approximately $6P$.

These components account for approximately $10P$ FLOPs per training token. For consistency with prior compute accounting and to conservatively include auxiliary training operations not represented in this simplified decomposition, we use an effective estimate of $12P$ per training token:

$$\mathrm{FLOPs}_{\mathrm{train}} = 12P\, T_{\mathrm{train}},$$

where T_{train} denotes the total number of tokens processed by the RL training loop.

Operation-level compute accounting We distinguish profiling, rollout generation, additional forward-only evaluations, and policy optimization:

$$C_{\mathrm{total}} \approx 2P\,(T_{\mathrm{profile}} + T_{\mathrm{rollout}} + T_{\mathrm{aux}}) + 6PT_{\mathrm{update}}.$$

Here, each token count includes the corresponding operations actually executed, and T_{update} counts token exposures across all optimization passes. Discarded rollout groups incur generation cost but are not charged for policy updates that were not performed.

Both offline profiling and online allocation obtain competence information from verified generations. Offline profiling makes this information available before the first policy update, allowing selection, rollout allocation, and curriculum ordering to be configured from the start. Its benefit depends on whether the subsequent compute savings exceed the upfront profiling cost. The magnitude of the resulting compute savings depends on this inference–training cost asymmetry: RL optimizers with lower per-token training cost would reduce the relative advantage of upfront profiling. Accordingly, our FLOP results should be interpreted under the common accounting assumptions above rather than as hardware-independent runtime or energy measurements.

M.6 HYPERPARAMETERS

Table 17: Hyperparameters for profiling, RL training, and evaluation on the mathematical reasoning experiments. Profiling and training rollouts use the same sampling settings.

Stage	Hyperparameter	Value
Profiling	Samples per query (N)	8
	Sampling temperature	1.0
	Top-p / top-k	1.0 / -1 (disabled)
Training	Optimizer	AdamW
	Learning rate	1×10^{-6}
	Advantage estimator	GRPO
	Clip ratio (low / high)	0.2 / 0.28
	Train batch size (prompts)	64
	Mini-batch size	8
	Max prompt length	2048
	Max response length	8192 (7B) / 2048 (1.5B)
	Rollout temperature	1.0
	Rollout top-p / top-k	1.0 / -1 (disabled)
	Group size G (baselines)	8
	Group size G (sGPO)	adaptive, $\{2, 4, 8\}$
	GPU memory utilization	0.8
Evaluation	Metric	avg@16 accuracy
	Samples per query	16
	Temperature	1.0
	Top-p / top-k	0.7 / -1 (disabled)
	Max response length	8192 (7B) / 2048 (1.5B)