REALLOC: HIERARCHICAL OPTIMIZATION BUDGET REALLOCATION FOR LLM POST-TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Post-training is central to improving the reasoning capabilities of large language models (LLMs). While existing post-training methods differ in optimization objectives, they all require aggregating these objectives across sampled training data to form the batch objective. Standard practices typically aggregate objectives through averaging, with recent research generally prioritizing sampled data at a single granularity level of training units and tailoring to specific algorithms. In this work, we revisit objective aggregation as a hierarchical budget allocation problem over three granularity levels, i.e., prompts, rollouts and tokens, and identify signals that can guide the allocation of optimization weight at each level. At the prompt level, we connect outcome variance to the gradient of expected accuracy and find that prompts with mixed outcomes exhibit greater learning potential. At the rollout level, we show that incorrect trajectories closer to the centroid of correct rollouts preserve more coherent reasoning and exhibit less redundant gradient structure. At the token level, we find that local entropy prominence identifies reasoning transitions more precisely than entropy magnitude alone. Based on these findings, we propose **ReAlloc**, a hierarchical optimization budget reallocation framework that leverages these level-specific signals into aggregation weights over three levels while leaving the underlying objective unchanged. This design allows ReAlloc to be applied to different post-training algorithms. Experiments with GRPO and OPD across three models and seven benchmarks demonstrate consistent improvements, with only minor extra computational overhead.

1 INTRODUCTION

Post-training plays an important role in advancing the reasoning capabilities of large language models (LLMs) (Guo et al., 2025; Ye et al., 2025; Fu et al., 2026). Although post-training methods define their optimization objectives differently, they have to aggregate token objectives into a batch objective for policy updates (Lu & Lab, 2025; Shao et al., 2024; Rafailov et al., 2023). In practice, standard aggregation often averages objectives, with the weights determined by the number of responses or sequence lengths (Liu et al., 2025; Zheng et al., 2025b; Hu et al., 2025). Recent work has explored prioritizing particular prompts (Xu et al., 2026a; Zhou et al., 2026b) or tokens during optimization (Wang et al., 2025; Xu et al., 2026b). However, these methods typically focus on a single granularity of sampled data and are developed for specific post-training algorithms.

Post-training naturally involves a hierarchy of sampled training units. A training batch contains multiple prompts, each prompt produces multiple rollouts and each rollout consists of a sequence of tokens. Training units at each level exhibit distinct characteristics, which may provide complementary signals for allocating optimization weight. As shown in Figure 1, the prompt on the left elicits highly consistent responses, whereas the prompt on the right produces more varied responses. Looking further within each prompt, even responses with the same incorrect outcome can differ substantially. One may preserve a coherent reasoning trajectory and fail at an intermediate decision, while another may collapse into degenerate generation. Within individual rollouts, some tokens mark transitions between reasoning steps, while others continue the current step.

Motivated by this, we explore whether and how to use these differences to guide objective aggregation. We first examine prompts through the lens of outcome variance, connecting it to the gradient of expected accuracy and empirically showing that prompts with mixed outcomes offer greater learn-

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

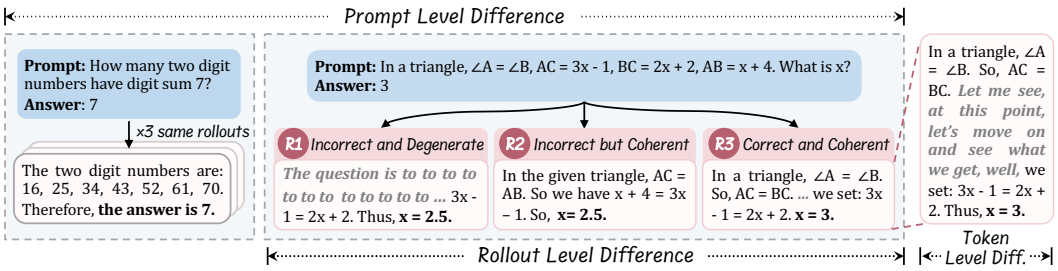


Figure 1: A motivating example: differences among training units at three levels. Prompts differ in rollout consistency, rollouts differ in trajectory structure, and tokens differ in roles within reasoning.

ing potential. We then turn to incorrect rollouts and study their distance from the centroid of correct rollouts, finding that nearby trajectories tend to preserve more coherent structures and exhibit higher gradient effective rank, while distant ones show more redundant gradient structure. Finally, at the token level, we find that local entropy prominence identifies reasoning transitions more precisely than entropy magnitude alone. These analyses provide observable signals for guiding aggregation at the prompt, rollout and token levels. Importantly, these patterns emerge across more than one post-training algorithm, suggesting that such signals are not tied to any specific objective.

Based on these findings, we propose **ReAlloc**, a hierarchical optimization budget reallocation framework for LLM post-training. ReAlloc maps the three level-specific signals to hierarchical aggregation weights over prompts, rollouts and tokens. Specifically, it prioritizes prompts with higher outcome variance, assigns greater weight to incorrect rollouts closer to the correct centroid, and emphasizes tokens with greater local entropy prominence. Importantly, ReAlloc changes only how objectives are aggregated while leaving the underlying objectives unchanged. This separation allows ReAlloc to be instantiated with different post-training algorithms. We apply ReAlloc to two representative post-training algorithms, Group Relative Policy Optimization (GRPO) (Shao et al., 2024) and On-Policy Distillation (OPD) (Agarwal et al., 2024; Lu & Lab, 2025). Further discussion of related work and the objectives of GRPO and OPD are in Appendices A.1 and A.2. Experiments across three models and seven benchmarks show consistent improvements under both algorithms. On Qwen3-4B-Base, ReAlloc improves Avg@8 by +8.44 on AMC23 under GRPO and by +5.42 on AIME25 under OPD. Ablation studies further show that reallocations at the three levels provide complementary improvements, while runtime measurements indicate only minor additional computational overhead. Our contributions are summarized as follows:

- We formulate objective aggregation as hierarchical budget allocation and identify three level-specific signals. Outcome variance at the prompt level, proximity to the correct centroid at the rollout level, and local entropy prominence at the token level. We analyze how these signals distinguish training units with different outcome patterns, trajectory structures and token roles.
- We propose ReAlloc, a hierarchical optimization budget reallocation framework that leverages these signals into aggregation weights over prompts, rollouts and tokens. ReAlloc changes only objective aggregation, making it applicable to different post-training algorithms.
- We evaluate ReAlloc with GRPO and OPD across three models and seven benchmarks. Results show that ReAlloc consistently improves the performance of both algorithms, with complementary gains across the three levels and only minor computational overhead.

2 Rethinking Objective Aggregation in LLM Post-Training

2.1 Notation

Let $x_i \sim \mathcal{D}$ denote the i-th prompt in a training batch of size B. For each prompt x_i, the current policy π_θ samples K responses, where the j-th response has length $T_{i,j}$,

$$y_{i,j} = (y_{i,j,t})_{t=1}^{T_{i,j}}, \qquad j = 1, \ldots, K.$$

For token $y_{i,j,t}$, the prefix context is denoted by $c_{i,j,t} = (x_i, y_{i,j,<t})$. We denote by $\ell_{i,j,t}(\theta)$ the token objective before aggregation.

2.2 Objective Aggregation in LLM Post-Training

Although different post-training algorithms construct $\ell_{i,j,t}(\theta)$ differently, they must aggregate local objectives across tokens, rollouts and prompts to form the batch objective. We write this aggregation in a general form as

$$\ell(\theta) = \sum_{i=1}^{B} \alpha_i \sum_{j=1}^{K} \beta_{i,j} \sum_{t=1}^{T_{i,j}} \gamma_{i,j,t} \ell_{i,j,t}(\theta), \tag{1}$$

where α_i, $\beta_{i,j}$, and $\gamma_{i,j,t}$ determine the contributions to optimization of prompt x_i, rollout $y_{i,j}$ and token $y_{i,j,t}$, respectively. Common aggregation rules assign these weights based on token counts and response lengths. For example, `seq-mean-token-mean` first averages token objectives within each rollout and then averages across rollouts, corresponding to

$$\alpha_i = \frac{1}{B}, \qquad \beta_{i,j} = \frac{1}{K}, \qquad \gamma_{i,j,t} = \frac{1}{T_{i,j}}.$$

In contrast, `token-mean` averages over all generated tokens, corresponding to

$$\alpha_i = 1, \qquad \beta_{i,j} = 1, \qquad \gamma_{i,j,t} = \frac{1}{\sum_{i=1}^{B}\sum_{j=1}^{K} T_{i,j}}.$$

Neither aggregation rule explicitly distinguishes prompts, rollouts or tokens according to their potential value for policy update. However, such aggregation can be suboptimal, motivating us to theoretically and empirically explore observable signals that can guide objective aggregation (further interpretations in Appendix A.3). Empirical analyses in this section use Qwen3-4B-Base with GRPO or OPD, DAPO-Math-17K for training, and AIME24 and AIME25 for evaluation.

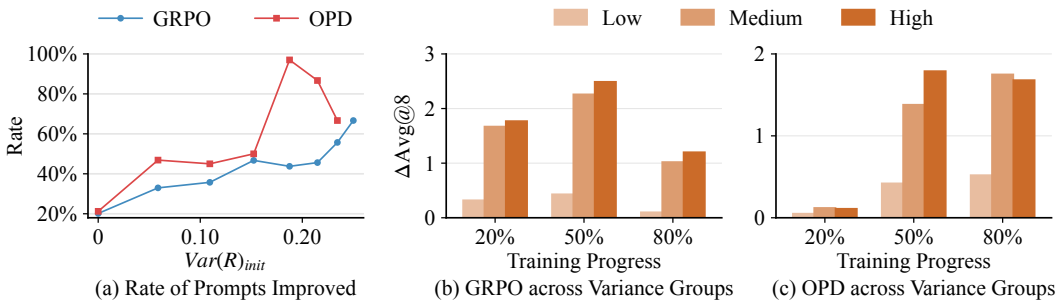
2.3 Prompts Exhibit Varying Learning Potential

Post-training improves LLMs by reshaping the output distribution toward desirable responses, but the opportunity for improvement varies across prompts. Prompts with both successful and unsuccessful rollouts expose the opportunity, because successful behavior is accessible, but the policy does not yet produce it reliably. In contrast, consistently successful prompts offer little room for further accuracy gains, while consistently unsuccessful prompts expose no successful continuation in the sampled group. We quantify this distinction using outcome variance.

For prompt x_i, let the current policy generate K rollouts with binary outcome rewards $R_{i,j} \in \{0, 1\}$. We define the empirical success rate $\hat{p}_i = \frac{1}{K}\sum_{j=1}^{K} R_{i,j}$ and outcome variance $V_i = \frac{1}{K}\sum_{j=1}^{K}(R_{i,j} - \hat{p}_i)^2 = \hat{p}_i(1 - \hat{p}_i)$, which directly captures the coexistence of successful and unsuccessful rollouts. Outcome variance also has a population-level connection to learning. Let $p_i(\theta)$ denote the expected outcome accuracy of prompt x_i under the current policy. Its gradient satisfies $\nabla_\theta p_i(\theta) = p_i(1 - p_i)(\mu_i^+ - \mu_i^-)$, where μ_i^+ and μ_i^- are the conditional mean policy scores of successful and unsuccessful responses. Thus, the population outcome variance $p_i(1 - p_i)$ appears as one factor in the gradient, and the empirical V_i estimates it up to a constant factor in expectation. While variance alone does not determine learning gain, this connection motivates it as a prompt-level signal. Detailed analyses are in Appendix A.4. We therefore examine empirically whether it reflects subsequent learning potential.

Mixed outcome prompts improve more frequently. We first examine whether outcome variance predicts subsequent learning progress. For each prompt, we measure its outcome variance at step t and evaluate the same prompt after 20 training steps. Figure 2(a) shows the fraction of prompts with improved Avg@8 as a function of initial outcome variance. Prompts that are correct in all rollouts are excluded, since their Avg@8 cannot increase further. Across both GRPO and OPD, improvement rates are generally higher at moderate to high variance, indicating that outcome variance captures differences in subsequent learning potential.

Training on medium and high variance prompts yields larger gains. We further compare learning gains across variance groups. At 20%, 50% and 80% of training, we split candidate prompts into variance tertiles and train for 20 additional steps on each group. As shown in Figure 2(b,c),



(a) Rate of Prompts Improved (b) GRPO across Variance Groups (c) OPD across Variance Groups

Figure 2: Outcome variance distinguishes prompt learning potential. (a) Fraction of prompts whose Avg@8 improves after additional training, grouped by initial outcome variance. (b, c) ΔAvg@8 after using the low, medium or high variance third of candidate prompts.

the medium and high variance groups yield substantially larger gains than the low variance group for both GRPO and OPD. These results show that outcome variance can be a criterion for prompt prioritization, and that training on prompts with mixed outcomes yields larger gains than training on low variance prompts.

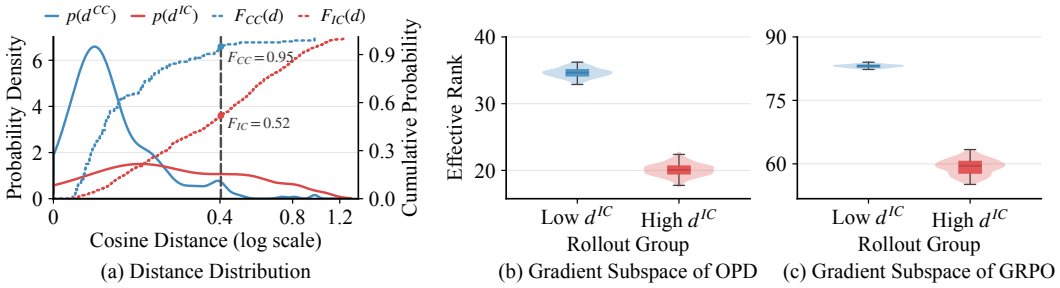
2.4 Incorrect Rollouts Differ in Trajectory Structure

Rollouts with the same outcome can exhibit substantially different trajectory structures, particularly when the outcome is incorrect. Some responses maintain a coherent reasoning process, make meaningful intermediate progress, and fail only after one or a few incorrect decisions. Others contain extensive repetition, malformed expressions or fragmented generations, with little consistent reasoning structure. Since correct rollouts provide a reference for trajectories that reach the desired outcome, we therefore focus on examining how incorrect rollouts behave in representation space.

Let $z(y)$ denote the representation of rollout y. We obtain $z(y)$ by pooling and normalizing the response hidden states. We construct a centroid from all correct rollouts $\mathcal{C}$ collected in a training batch $z_{\mathcal{C}} = \mathrm{Norm}\left(\frac{1}{|\mathcal{C}|}\sum_{y\in\mathcal{C}} z(y)\right)$, and measure the cosine distance of each rollout to this centroid with $d(y) = 1 - \cos\left(z(y), z_{\mathcal{C}}\right)$. We denote the resulting distances for correct and incorrect rollouts by d^{CC} and d^{IC}, respectively.

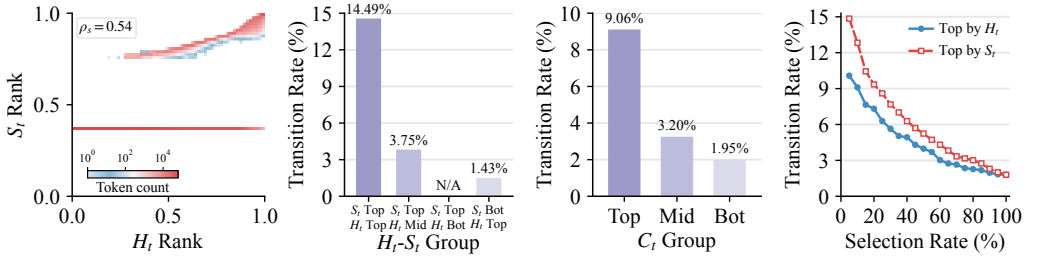
Distance reflects structural differences among incorrect rollouts. Figure 3(a) shows that incorrect rollouts overlap substantially with correct rollouts at low distances, but extend over a much broader range. At the 95th percentile of the correct rollout distance distribution, only 51.9% of incorrect rollouts remain within the same threshold. To interpret this variation, we rank incorrect rollouts by d^{IC} and qualitatively examine trajectories from the highest and lowest distance groups (examples in Appendix A.5). Low d^{IC} rollouts generally preserve recognizable reasoning structure despite producing incorrect answers, whereas high d^{IC} rollouts exhibit substantially more repetition, fragmented expressions and other degenerate generation patterns. We therefore refer to low d^{IC} trajectories as *Incorrect but Coherent* and high d^{IC} trajectories as *Incorrect and Degenerate*, suggesting that distance to the correct centroid can provide a signal of trajectory structure among incorrect responses.

Distant incorrect rollouts induce more redundant gradients. We further examine whether the trajectory differences reflected by d^{IC} are also manifested in the gradients induced during training. Using a fixed set of sampled prompts and rollouts, we compute the gradient induced by each incorrect rollout under the original OPD or GRPO and partition incorrect rollouts into nearby and distant groups by d^{IC}. We then compare the effective rank of their normalized gradient Gram matrices using repeated equal size subsampling. As shown in Figure 3(b,c), distant incorrect rollouts exhibit markedly lower effective rank under both OPD and GRPO, indicating that their gradient directions span a lower dimensional space and are more redundant than those of nearby incorrect rollouts.



(a) Distance Distribution (b) Gradient Subspace of OPD (c) Gradient Subspace of GRPO

Figure 3: Rollout distance captures structural differences among incorrect trajectories. (a) Distance distributions to the correct centroid, shown on a log scale for better visualization. (b, c) Gradient subspace differences across incorrect rollouts.



(a) H_t-S_t Rank Relation (b) H_t-S_t Transition Rate (c) C_t Transition Rate (d) Transition Rate by Selection

Figure 4: Local entropy peaks identify reasoning transition tokens. (a) Joint percentile ranks of token entropy H_t and entropy peak score S_t. (b) Reasoning transition rates in selected intersections of the H_t and S_t rankings. (c) Transition rates across local prominence groups C_t among top 20% H_t tokens. (d) Transition rates when selecting the same fraction of tokens according to S_t or H_t.

2.5 Token Entropy Peaks Distinguish Reasoning Transitions

Recent work by Wang et al. (2025) shows that high entropy tokens often mark reasoning forks and that updating only these tokens can match or exceed full token training. However, entropy magnitude alone does not distinguish an isolated uncertainty peak from a sustained high entropy region. We therefore examine whether *local entropy prominence* more precisely identifies reasoning transitions.

Let $H_t = -\sum_v \pi_\theta(v \mid c_{i,j,t}) \log \pi_\theta(v \mid c_{i,j,t})$ denote the predictive entropy at token position t. For a window of w tokens on each side, define

$$H_t^L = \frac{1}{w}\sum_{k=1}^{w} H_{t-k}, \qquad H_t^R = \frac{1}{w}\sum_{k=1}^{w} H_{t+k}.$$

We define the entropy peak score as

$$S_t = \sqrt{[H_t - H_t^L]_+ [H_t - H_t^R]_+}, \qquad [z]_+ = \max(z, 0). \tag{2}$$

Thus, $S_t > 0$ only when the center entropy exceeds both neighboring means, capturing bilateral local prominence rather than entropy magnitude alone. To separate absolute uncertainty from relative local prominence, for $H_t > 0$ we write

$$S_t = H_t C_t, \qquad C_t = \sqrt{\left[1 - \frac{H_t^L}{H_t}\right]_+ \left[1 - \frac{H_t^R}{H_t}\right]_+}.$$

Here, $C_t \in [0, 1]$ measures how strongly the center token stands out relative to its local neighborhood. We use S_t and C_t to test whether local prominence further distinguishes reasoning transitions among uncertain tokens. Further analysis is in Appendix A.6.

Entropy peaks distinguish reasoning transitions among uncertain tokens. In Figure 4(a), the rankings induced by H_t and S_t have a Spearman correlation of $\rho_s = 0.538$, indicating that the two rankings are related but substantially different. To characterize the selected positions, we use GPT-5.6 Sol annotations (see details in Appendix A.7) and define the reasoning transition rate as the fraction of tokens annotated as reasoning transitions. Within the top 20% by H_t, tokens with higher S_t exhibit substantially higher reasoning transition rates, while tokens with high S_t but only moderate H_t still outperform high entropy tokens with low peak scores (Figure 4(b)). These results show that local entropy prominence provides additional information for identifying reasoning transitions beyond entropy magnitude alone.

Local prominence sharpens reasoning transition identification. To further examine the role of relative local prominence, we group tokens in the top 20% by H_t according to the C_t component. As shown in Figure 4(c), reasoning transition rates increase with C_t among high entropy tokens, indicating that relative local prominence further distinguishes reasoning transitions. Figure 4(d) shows that, at the same selection fraction, ranking by S_t consistently yields higher transition rates than ranking by H_t, with larger gains at smaller selection budgets. Together, these results show that local prominence helps identify reasoning transitions more precisely than entropy magnitude alone.

3 Hierarchical Optimization Budget Reallocation

Existing post-training methods typically average optimization objectives across training units. However, this practice overlooks differences among prompts, rollouts and tokens in potential contributions to optimization. Based on the findings in Sec. 2, we use these level-specific observable signals to guide the allocation of optimization weight across prompts, rollouts and tokens. Building on the aggregation objective in Eq. 1, for each batch $\mathcal{B} \sim \mathcal{D}$, we define the maximization objective as:

$$J_{\mathrm{ReAlloc}}(\theta) = \mathbb{E}_{\mathcal{B}\sim\mathcal{D},\{y_{i,j}\}_{j=1}^{K}\sim\pi_{\theta_{\mathrm{old}}}(\cdot|x_i)}\left[\sum_{i=1}^{B}\alpha_i\sum_{j=1}^{K}\beta_{i,j}\sum_{t=1}^{T_{i,j}}\gamma_{i,j,t}\ell_{i,j,t}(\theta)\right], \tag{3}$$

where

$$\alpha_i = \frac{m_i}{\sum_{i'=1}^{B} m_{i'}}, \qquad \beta_{i,j} = \begin{cases} \dfrac{1}{K} & R_{i,j} = 1, \\ \dfrac{|\mathcal{I}_i^-|}{K}\dfrac{\exp(\frac{-r_{i,j}^{IC}}{\tau})}{\sum_{k\in\mathcal{I}_i^-}\exp(\frac{-r_{i,k}^{IC}}{\tau})} & R_{i,j} = 0. \end{cases} \qquad \gamma_{i,j,t} = \frac{S_{i,j,t}^{(w)}}{\sum_{t=1}^{T_{i,j}} S_{i,j,t}^{(w)}}.$$

Here, m_i is the prompt level selection indicator. We compute the outcome variance V_i of prompt x_i and set $m_i = 1$ for prompts in the top-ρ fraction of V_i within the batch. At the rollout level, $\mathcal{I}_i^- = \{j : R_{i,j} = 0\}$ denotes the set of incorrect rollouts. Each rollout is represented by concatenating mean pooled hidden states over relative position segments followed by L2 normalization. The correct centroid is the normalized mean representation of correct rollouts in the batch. We use $r_{i,j}^{IC}$ as the batch global rank of d^{IC} in ascending order. The softmax over $-r_{i,j}^{IC}$, where $\tau > 0$ controls the smoothness, assigns larger weights to incorrect rollouts closer to the correct centroid, while correct rollouts retain their original weights. If there is no correct rollout in the batch, we set original weights instead. At the token level, $S_{i,j,t}$ is the local entropy prominence in Eq. 2 with a window size w. The local objective $\ell_{i,j,t}(\theta)$ remains unchanged. Thus, our ReAlloc can be directly applied to different algorithms.

4 Experiment

4.1 Settings

Models and Methods. We evaluate our method with GRPO (Shao et al., 2024) and OPD (Lu & Lab, 2025) on three models: Qwen3-8B-Base, Qwen3-4B-Base (Yang et al., 2025) and DeepSeek-R1-Distill-Qwen-1.5B (denote as DeepSeek-R1-1.5B) (Guo et al., 2025). For GRPO, we compare with vanilla GRPO (Shao et al., 2024), DAPO (Yu et al., 2025) and the high entropy token selection method of Wang et al. (2025) (denote as FORK). For OPD, we compare ReAlloc with vanilla OPD

Table 1: Main results of GRPO and its variants on seven benchmarks. A@8 denotes Avg@8 and P@8 denotes Pass@8. **Bold** indicates best performance and underline indicates second-best.

Method	AIME24		AIME25		AIME26		AIMO		AMC23		HMMT25		MATH500		Avg.	
	A@8	P@8	A@8	P@8	A@8	P@8	A@8	P@8	A@8	P@8	A@8	P@8	A@8	P@8	A@8	P@8
Qwen3-8B-Base																
Baseline	10.42	40.00	7.08	20.00	7.92	26.67	7.29	20.00	43.13	80.00	2.08	13.33	60.90	90.40	19.83	41.49
GRPO	22.92	50.00	18.75	33.33	16.25	30.00	15.42	28.33	64.06	85.00	9.17	23.33	84.10	93.60	32.95	49.09
DAPO	24.17	46.67	17.91	30.00	16.67	33.33	16.25	30.00	65.63	90.00	8.75	20.00	84.75	93.80	33.45	49.11
FORK	25.00	53.33	19.17	36.67	19.92	36.67	17.71	31.67	69.69	87.50	10.00	23.33	85.18	94.00	35.24	51.88
ReAlloc (Ours)	**27.50**	**56.67**	**21.67**	**43.33**	**20.83**	**40.00**	**19.17**	**33.33**	**74.06**	**92.50**	**10.83**	**30.00**	**87.25**	**95.00**	**37.33**	**55.83**
Qwen3-4B-Base																
Baseline	6.67	20.00	5.42	30.00	5.42	16.67	5.21	15.00	34.69	72.50	0.83	6.67	41.48	85.40	14.25	35.18
GRPO	19.17	33.33	14.58	30.00	13.75	26.67	12.71	21.67	57.50	82.50	7.08	16.67	79.00	92.00	29.11	43.26
DAPO	21.67	36.67	17.08	40.00	12.92	30.00	13.33	23.33	60.94	80.00	7.50	16.67	79.25	91.80	30.38	45.49
FORK	22.08	43.33	17.50	36.67	14.58	33.33	15.83	25.00	61.25	85.00	7.08	20.00	82.13	92.40	31.49	47.96
ReAlloc (Ours)	**23.75**	**46.67**	**19.58**	**43.33**	**18.75**	**40.00**	**17.50**	**28.33**	**65.94**	**90.00**	**9.17**	**23.33**	**83.50**	**93.40**	**34.03**	**52.15**
DeepSeek-R1-1.5B																
Baseline	25.42	50.00	21.67	33.33	19.58	40.00	18.75	36.67	67.50	87.50	14.58	20.00	82.25	93.80	35.68	51.61
GRPO	50.42	73.33	31.25	40.00	33.33	46.67	38.13	53.33	82.81	90.00	17.92	36.67	82.50	93.60	48.05	61.94
DAPO	50.00	70.00	30.42	43.33	33.75	50.00	37.92	51.67	83.13	90.00	18.17	33.33	83.75	94.00	48.16	61.76
FORK	50.83	73.33	31.67	43.33	32.92	46.67	37.92	55.00	83.44	92.50	19.58	36.67	84.28	93.40	48.66	62.99
ReAlloc (Ours)	**51.25**	**76.67**	**32.50**	**46.67**	**34.17**	**53.33**	**38.33**	**56.67**	**83.75**	**95.00**	**20.00**	**40.00**	**85.30**	**94.40**	**49.33**	**66.10**

Table 2: Main results of OPD and its variants on seven benchmarks. A@8 denotes Avg@8 and P@8 denotes Pass@8. **Bold** indicates best performance and underline indicates second-best.

Method	AIME24		AIME25		AIME26		AIMO		AMC23		HMMT25		MATH500		Avg.	
	A@8	P@8	A@8	P@8	A@8	P@8	A@8	P@8	A@8	P@8	A@8	P@8	A@8	P@8	A@8	P@8
Qwen3-8B-Base																
Baseline	10.42	40.00	7.08	20.00	7.92	26.67	7.29	20.00	43.13	80.00	2.08	13.33	60.90	90.40	19.83	41.49
OPD	23.33	40.00	19.17	33.33	17.92	30.00	19.17	35.00	70.94	92.50	11.67	23.33	85.00	94.40	35.31	49.80
TIP	24.17	40.00	19.58	30.00	19.17	33.33	18.96	40.00	70.94	90.00	12.08	30.00	84.90	94.00	35.69	51.05
BRTS	25.00	43.33	20.00	33.33	19.58	36.67	19.79	38.33	71.56	92.50	12.50	23.33	85.25	94.20	36.24	51.67
ReAlloc (Ours)	**25.83**	**53.33**	**21.67**	**36.67**	**20.42**	**40.00**	**20.21**	**41.67**	**73.44**	**95.00**	**12.92**	**40.00**	**85.38**	**94.60**	**37.12**	**57.32**
Qwen3-4B-Base																
Baseline	6.67	20.00	5.42	30.00	5.42	16.67	5.21	15.00	34.69	72.50	0.83	6.67	41.48	85.40	14.25	35.18
OPD	22.08	40.00	15.83	23.33	15.42	33.33	13.96	26.67	60.94	90.00	9.17	20.00	81.38	93.00	31.25	46.62
TIP	22.92	36.67	17.08	30.00	15.00	30.00	14.58	28.33	61.56	87.50	10.42	16.67	81.90	92.80	31.92	46.00
BRTS	23.75	43.33	17.50	36.67	15.83	33.33	15.00	31.67	62.50	92.50	9.17	20.00	82.20	93.40	32.28	50.13
ReAlloc (Ours)	**25.42**	**46.67**	**21.25**	**46.67**	**18.33**	**36.67**	**16.46**	**35.00**	**63.75**	**95.00**	**10.83**	**23.33**	**82.63**	**93.60**	**34.10**	**53.85**
DeepSeek-R1-1.5B																
Baseline	25.42	50.00	21.67	33.33	19.58	40.00	18.75	36.67	67.50	87.50	14.58	20.00	82.25	93.80	35.68	51.61
OPD	44.58	76.67	30.83	43.33	23.75	50.00	32.92	56.67	81.88	**95.00**	19.58	**36.67**	84.78	94.20	45.47	64.65
TIP	44.17	73.33	30.83	40.00	24.17	46.67	33.75	55.00	81.56	87.50	19.17	30.00	85.00	95.80	45.52	61.19
BRTS	45.00	**80.00**	31.25	43.33	25.00	50.00	34.38	58.33	82.50	90.00	20.00	33.33	85.50	96.00	46.23	64.43
ReAlloc (Ours)	**47.50**	76.67	**31.67**	**46.67**	**27.08**	**53.33**	**35.63**	**61.67**	**82.81**	**95.00**	**21.67**	**36.67**	**87.50**	**97.00**	**47.69**	**66.72**

(Lu & Lab, 2025), BRTS (Zhang et al., 2026) and TIP (Xu et al., 2026b). We use Qwen3-8B-GRPO Kim et al. (2025), Qwen3-4B-Base-GRPO Li et al. (2026) and JustRL-DeepSeek-1.5B (He et al., 2025) as teacher models. We use `verl` (Sheng et al., 2025) and train on DAPO-Math-17K (Yu et al., 2025). During training, we sample eight rollouts per prompt and set the max response length as 8k, with disabled thinking mode. Implementation details are in Appendix A.8.

Benchmarks and Evaluation. We evaluate methods on seven benchmarks: AIME24 Zhang & Math-AI (2024), AIME25 Zhang & Math-AI (2025), AIME26 (Zhang & Math-AI, 2026), AMC23 (Mathematical Association of America, 2023), AIMO (only AIME22 and AIME23) (AI-MO, 2024), HMMT25 (Dekoninck et al., 2026) and MATH500 (Lightman et al., 2024). For each problem, we sample eight responses with temperature of 1.0 and set the max response length as 32K. We report Avg@8, the mean accuracy across the eight responses averaged over problems, and Pass@8, the fraction of problems with at least one correct response among the eight samples.

4.2 Main Results

Tables 1 and 2 present the main results under GRPO and OPD. Under GRPO, ReAlloc outperforms vanilla GRPO as well as DAPO and FORK. On Qwen3-4B-Base, it improves Avg@8 by

Table 3: Ablation of L-P, L-R and L-T under GRPO on Qwen3-4B-Base. A@8 denotes Avg@8 and P@8 denotes Pass@8. **Bold** indicates best performance and underline indicates second-best.

L-P	L-R	L-T	AIME24		AIME25		AIME26		AIMO		AMC23		HMMT25		MATH500		Avg.	
			A@8	P@8	A@8	P@8	A@8	P@8	A@8	P@8	A@8	P@8	A@8	P@8	A@8	P@8	A@8	P@8
–	–	–	19.17	33.33	14.58	30.00	13.75	26.67	12.71	21.67	57.50	82.50	7.08	16.67	79.00	92.00	29.11	43.26
✓	–	–	20.83	43.33	14.17	36.67	17.92	30.00	14.58	28.33	59.38	82.50	8.75	16.67	79.63	92.00	30.75	47.07
–	✓	–	18.75	40.00	13.33	33.33	15.42	30.00	15.83	23.33	62.50	85.00	7.50	**23.33**	82.30	92.00	30.80	46.71
–	–	✓	22.50	36.67	18.33	33.33	15.00	36.67	16.25	28.33	61.88	**90.00**	7.08	16.67	80.18	90.80	31.60	47.50
–	✓	✓	22.08	36.67	17.50	36.67	17.50	33.33	16.67	28.33	64.69	87.50	8.75	20.00	83.45	92.60	32.95	47.87
✓	–	✓	22.92	43.33	18.33	33.33	16.67	36.67	16.46	26.67	63.75	85.00	8.75	20.00	82.25	91.80	32.73	48.11
✓	✓	–	23.33	43.33	17.08	40.00	18.33	36.67	15.42	**30.00**	65.63	87.50	7.92	20.00	82.30	91.80	32.86	49.90
✓	✓	✓	**23.75**	**46.67**	**19.58**	**43.33**	**18.75**	**40.00**	**17.50**	28.33	**65.94**	**90.00**	**9.17**	**23.33**	**83.50**	**93.40**	**34.03**	**52.15**

Table 4: Ablation of L-P, L-R and L-T under OPD on Qwen3-4B-Base. A@8 denotes Avg@8 and P@8 denotes Pass@8. **Bold** indicates best performance and underline indicates second-best.

L-P	L-R	L-T	AIME24		AIME25		AIME26		AIMO		AMC23		HMMT25		MATH500		Avg.	
			A@8	P@8	A@8	P@8	A@8	P@8	A@8	P@8	A@8	P@8	A@8	P@8	A@8	P@8	A@8	P@8
–	–	–	22.08	40.00	15.83	23.33	15.42	33.33	13.96	26.67	60.94	90.00	9.17	20.00	81.38	93.00	31.25	46.62
✓	–	–	22.50	40.00	18.33	30.00	16.25	30.00	15.00	30.00	62.50	90.00	9.17	20.00	82.45	92.60	32.31	47.51
–	✓	–	24.58	43.33	17.92	26.67	17.08	33.33	13.96	30.00	62.50	90.00	10.42	16.67	82.33	93.00	32.68	47.57
–	–	✓	22.92	40.00	20.00	40.00	16.67	30.00	14.38	25.00	60.63	85.00	10.42	23.33	81.43	92.60	32.35	47.99
–	✓	✓	24.58	43.33	17.92	43.33	15.83	33.33	**16.46**	33.33	63.44	87.50	10.42	20.00	82.38	93.40	33.00	50.60
✓	–	✓	24.17	43.33	17.92	40.00	17.08	33.33	15.63	28.33	62.81	90.00	9.58	20.00	82.58	92.80	32.82	49.69
✓	✓	–	24.58	**46.67**	15.83	30.00	**18.33**	**36.67**	16.04	31.67	62.81	92.50	**11.25**	**26.67**	82.00	**93.60**	32.98	51.11
✓	✓	✓	**25.42**	**46.67**	**21.25**	**46.67**	**18.33**	**36.67**	**16.46**	**35.00**	**63.75**	**95.00**	10.83	23.33	**82.63**	**93.60**	**34.10**	**53.85**

8.44 points on AMC23 and 5.00 points on AIME26, while increasing Pass@8 on AIME26 by 13.33 points. The average Avg@8 improves by 4.92 points over vanilla GRPO. Under OPD, ReAlloc similarly improves over vanilla OPD, TIP, and BRTS. On Qwen3-4B-Base, it increases AIME25 Avg@8 from 15.83 to 21.25 (+5.42) and Pass@8 from 23.33 to 46.67 (+23.34), while improving the average Avg@8 by 2.85 points. Similar improvements are observed on Qwen3-8B-Base and DeepSeek-R1-1.5B under both algorithms. Taken together, the results demonstrate that ReAlloc provides consistent gains across optimization algorithms, models and benchmarks. It consistently benefits both Avg@8 and Pass@8, suggesting that hierarchical budget reallocation improves not only the average quality of sampled responses but also the coverage of correct solutions. These results suggest that the benefits of hierarchical budget reallocation are not tied to a particular model scale or post-training objective, but generalize consistently across different training settings.

4.3 Ablation Study

To examine the contribution of each level in ReAlloc, we conduct ablation studies on Qwen3-4B-Base under both GRPO and OPD. We separately enable prompt level (L-P), rollout level (L-R) and token level (L-T) reallocation, as well as their pairwise combinations. As shown in Tables 3 and 4, each individual level improves the overall average performance over the vanilla baseline. Under GRPO, L-T gives the largest standalone gain in Avg@8, while under OPD, L-R performs best among the single-level variants. This suggests that no single level consistently dominates across optimization algorithms. Combining multiple levels yields further improvements, indicating that the three components are complementary rather than redundant. All pairwise combinations outperform the corresponding single variants, while enabling all three levels achieves the best overall performance. These results validate the hierarchical design of ReAlloc. Reallocating the optimization budget at multiple levels jointly provides stronger gains than operating at any individual level alone.

4.4 Sensitivity of Hyperparameters

We study the sensitivity of the three level-specific hyperparameters: the prompt retention ratio ρ for L-P, the temperature τ controlling the smoothness of weighting for L-R, and the local window size w used to compute entropy peak scores for L-T. To isolate the effect of each hyperparameter, we enable only the corresponding reallocation level setting and disable the other two. Figure 5 reports Avg@8 under both GRPO and OPD. Across all three levels, performance peaks at moderate hyperparameter values, with similar trends under GRPO and OPD. For L-P, $\rho = 0.5$ achieves the best performance

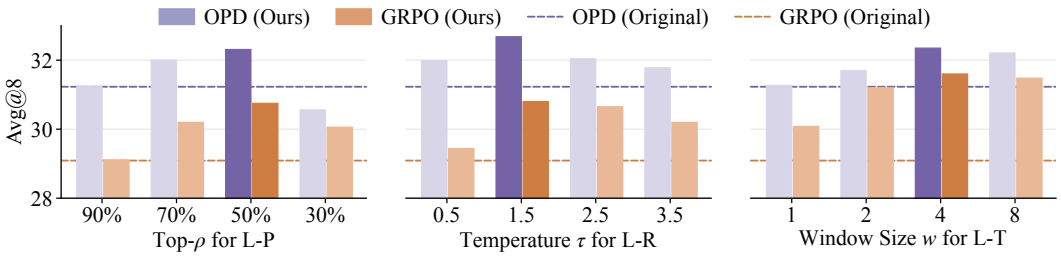


Figure 5: Hyperparameter sensitivity of OPD and GRPO for ρ, τ and w. Dark colored bars indicate the best setting, while dashed lines denote the performance of the original OPD or GRPO.

Table 5: Computational overhead comparison of three levels under GRPO and OPD.

Setting	GRPO				OPD			
	Step Time (s)	Δ (%)	Peak Mem. (GiB)	Δ (%)	Step Time (s)	Δ (%)	Peak Mem. (GiB)	Δ (%)
–	106.74 ± 3.79	–	85.48	–	168.82 ± 2.86	–	107.15	–
L-P	106.92 ± 3.45	+0.17	85.48	0.00	168.71 ± 2.82	-0.07	107.15	0.00
L-R	107.30 ± 3.46	+0.52	85.48	0.00	168.84 ± 2.83	+0.01	107.15	0.00
L-T	107.90 ± 3.39	+1.09	85.48	0.00	169.02 ± 2.86	+0.12	107.15	0.00
ReAlloc	108.18 ± 3.43	+1.35	85.48	0.00	169.30 ± 2.89	+0.28	107.15	0.00

for both OPD and GRPO. Retaining either too few or too many prompts reduces the benefit, with extreme settings yielding performance equal to or even worse than the original algorithm. For L-R, $\tau = 1.5$ performs best for both methods, while larger temperatures gradually weaken the benefit of rollout reweighting. Similarly, for L-T, $w = 4$ yields the best performance, whereas increasing the window size further provides no additional benefit. Overall, the consistent trends across GRPO and OPD indicate that the three reallocation mechanisms are not highly sensitive to a narrow hyperparameter choice, while each benefits from avoiding overly aggressive settings. The complete numerical results are provided in Appendix A.9.

4.5 Computational Overhead

To assess the computational overhead of the three proxies, we measure their additional cost under both GRPO and OPD. We use a fixed setup in which all variants process the same prompt-response batches, and run three independent trials for each variant, each averaging 20 measured steps after 10 warm-up steps. We report the mean ± standard deviation across trials. We report step time, including proxy computation and peak GPU memory, with relative changes (Δ) from the corresponding baseline. As shown in Table 5, across both GRPO and OPD, the three proxies incur only minor runtime overhead and no measurable increase in peak GPU memory. Overall, these results indicate that the three proxies are computationally lightweight, adding little runtime cost and no measurable memory overhead.

5 Conclusion

In this work, we revisit objective aggregation in LLM post-training from a hierarchical perspective. We identify distinct signals at the prompt, rollout and token levels. Outcome variance reflects prompt learning potential, proximity to the correct centroid captures structural and gradient differences among incorrect trajectories, and local entropy prominence more precisely identifies reasoning transitions. Based on these findings, we propose ReAlloc, a hierarchical optimization budget reallocation framework that maps these level-specific signals into aggregation weights while leaving the underlying post-training objective unchanged. Experiments with GRPO and OPD across three models and seven benchmarks demonstrate consistent improvements. Our current study is limited by its reliance on outcome rewards at the prompt level, its focus on mathematical reasoning, and its evaluation on two post-training algorithms. Extending ReAlloc to weaker reward signals, broader domains and other post-training methods is a promising direction for future work. Overall, our results highlight objective aggregation as an important design choice in LLM post-training.

AI USE STATEMENT

In this work, we used generative AI tools to assist with editing the research paper to improve readability. We also used generative AI tools for annotation in our empirical analysis, specifically to identify reasoning-transition tokens. We have not used generative AI tools for writing proofs, implementing methods, interpreting results, developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, supporting qualitative and thematic data analysis. Generating synthetic datasets, assisting with translation and cleaning and reformatting datasets are not applicable to this work. Additionally, we used generative AI tools for searching for information and formatting references. We have reviewed all AI-assisted work. We reviewed the reference formatting generated or revised by AI. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of our proposed approach, experimental settings, and implementation details in the main paper and appendix. An anonymous version of the source code and the full implementation are provided in the supplementary materials. The code is also available at `https://anonymous.4open.science/r/ReAlloc`.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://proceedings.iclr.cc/paper_files/paper/2024/hash/5be69a584901a26c521c2b51e40a4c20-Abstract-Conference.html`.

AI-MO. AIMO Validation AIME. Hugging Face dataset, 2024. URL `https://huggingface.co/datasets/AI-MO/aimo-validation-aime`. Accessed September 26, 2026.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: Matharena as an evaluation platform for mathematics with llms. 2026. URL `https://arxiv.org/abs/2605.00674`.

Jiayan Fu, Hang Xu, Yong Zhang, Zhaokai Luo, Yao Hu, Dongyan Zhao, and Mu Chuan. Pact: From credit assignment to critic alignment, 2026. URL `https://arxiv.org/abs/2609.26355`.

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

Bingxiang He, Zekai Qu, Zeyuan Liu, Yinghao Chen, Yuxin Zuo, Cheng Qian, Kaiyan Zhang, Weize Chen, Chaojun Xiao, Ganqu Cui, Ning Ding, and Zhiyuan Liu. JustRL: Scaling a 1.5B LLM with a Simple RL Recipe, 2025. URL `https://arxiv.org/abs/2512.16649`.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. REINFORCE++: Stabilizing critic-free policy optimization with global advantage normalization, 2025. URL `https://arxiv.org/abs/2501.03262`.

Woogyeol Jin, Taywon Min, Yongjin Yang, Dennis Wei, Yi Zhou, Swanand Ravindra Kadhe, Nathalie Baracaldo, and Kimin Lee. Entropy-aware on-policy distillation of language models, 2026. URL `https://arxiv.org/abs/2603.07079`.

Yoonjeon Kim, Doohyuk Jang, and Eunho Yang. Meta-awareness enhances reasoning models: Self-alignment reinforcement learning. *arXiv preprint arXiv:2510.03259*, 2025.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huanang Gao, Wenkai Yang, Zhiyuan Liu, and Ning Ding. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint arXiv:2604.13016*, 2026.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/aca97732e30bcf1303bc22ac3924fd16-Paper-Conference.pdf.

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, 2025. URL https://arxiv.org/abs/2503.20783.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

Mathematical Association of America. American mathematics competitions. https://maa.org/student-programs/amc/, 2023. Accessed September 16, 2026.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297. ACM, 2025. doi: 10.1145/3689031.3696075. URL https://doi.org/10.1145/3689031.3696075.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning, 2025. URL https://arxiv.org/abs/2506.01939.

Yuanda Xu, Hejian Sang, Zhengze Zhou, Ran He, and Zhipeng Wang. Paced: Distillation and on-policy self-distillation at the frontier of student competence, 2026a. URL https://arxiv.org/abs/2603.11178.

Yuanda Xu, Hejian Sang, Zhengze Zhou, Ran He, Zhipeng Wang, and Alborz Geramifard. Tip: Token importance in on-policy distillation, 2026b. URL https://arxiv.org/abs/2604.14084.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report, 2025. URL https://arxiv.org/abs/2505.09388.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning, 2025. URL https://arxiv.org/abs/2502.03387.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, juncai liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 113222–113244. Curran Associates, Inc., 2025. doi: 10.52202/085713-3775. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/a4277440d50f1f15d2cb4c14f7e0c0d2-Paper-Conference.pdf.

Ke Zhang, Yunjie Tian, DongDi Zhao, Yijiang Li, Yuanye Liu, Vishal M Patel, and Di Fu. On-policy distillation with best-of-n teacher rollout selection, 2026. URL https://arxiv.org/abs/2605.09725.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2026, 2026.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models, 2026. URL https://arxiv.org/abs/2601.18734.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025a. URL https://arxiv.org/abs/2507.18071.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025b. URL https://arxiv.org/abs/2507.18071.

Zhanke Zhou, Xiangyu Lu, Chentao Cao, Brando Miranda, Tongliang Liu, Bo Han, and Sanmi Koyejo. The easy, the hard, and the learnable: Confidence and difficulty-adaptive policy optimization for LLM reasoning. In *ICML*, 2026a.

Zhanke Zhou, Xiangyu Lu, Chentao Cao, Brando Miranda, Tongliang Liu, Bo Han, and Sanmi Koyejo. The easy, the hard, and the learnable: Confidence and difficulty-adaptive policy optimization for llm reasoning, 2026b. URL https://arxiv.org/abs/2606.07950.

A Appendix

A.1 Related Work

A.1.1 On-Policy Optimization and Distillation for LLMs

On-policy optimization has become an important paradigm for LLM post-training, especially for large reasoning models. GRPO eliminates the learned critic used in PPO and estimates advantages from groups of responses sampled for the same prompt (Shao et al., 2024), and has subsequently become a widely used RLVR baseline (Guo et al., 2025). Building on GRPO, DAPO introduces decoupled clipping, dynamic sampling, token-level loss aggregation, and overlong reward shaping to improve exploration and training stability (Yu et al., 2025), while Dr. GRPO identifies and removes biased normalization in GRPO to improve optimization and token efficiency (Liu et al.,

2025). GSPO instead moves importance weighting and clipping from the token level to the sequence level, highlighting the importance of optimization granularity in RL post-training (Zheng et al., 2025a).

Beyond reward-based RL, on-policy distillation trains on student-generated trajectories with dense teacher supervision at student-visited states, reducing the train inference distribution mismatch of off-policy distillation. Generalized Knowledge Distillation (GKD) applies teacher student divergence directly on student-generated sequences (Agarwal et al., 2024), while recent OPD formulations use token-level reverse-KL estimates as dense rewards within a policy-gradient-style update (Lu & Lab, 2025). Recent extensions include OPSD, which enables self-distillation by instantiating the teacher and student from the same model under asymmetric contexts (Zhao et al., 2026), and EOPD, which adapts the distillation objective according to teacher uncertainty to mitigate the mode seeking behavior of standard reverse-KL OPD (Jin et al., 2026).

A.1.2 Selective Optimization in LLM Post-Training

Recent studies have explored selectively allocating optimization effort to training units that are expected to provide more informative learning signals. CoDaPO adjusts problem sampling and weighting according to model confidence and empirical difficulty, concentrating training on examples that fall within the model's current learnable region (Zhou et al., 2026a). PACED similarly prioritizes problems near the student's competence frontier using pass-rate-based estimates (Xu et al., 2026a). Regarding the token selection, Wang et al. (2025) show that a small subset of high-entropy "forking tokens" accounts for a large fraction of the learning gains in RLVR, and that updating only these tokens can match or outperform training over all generated tokens. For on-policy distillation, TIP combines student uncertainty with teacher student disagreement to identify informative tokens for optimization (Xu et al., 2026b). These studies demonstrate that training units can differ substantially in their usefulness for post-training, and that prioritizing selected problems or tokens can improve optimization efficiency. However, existing methods typically make such decisions at a single granularity and often design the selection criterion around a particular training setting. In contrast, our work considers prompt, rollout and token levels jointly, and formulates objective aggregation as a hierarchical optimization budget allocation problem. This perspective allows level-specific signals to determine the relative contribution of training units while keeping the underlying local training objective unchanged.

A.2 LLM Post-Training Algorithms

Group Relative Policy Optimization. GRPO optimizes the policy using relative rewards among multiple responses sampled from the same prompt. Given rewards $\{r_{i,j}\}_{j=1}^{K}$ for the K responses to x_i, the group-relative advantage is

$$A_{i,j} = \frac{r_{i,j} - \mu_i}{\sigma_i},$$

where μ_i and σ_i are the mean and standard deviation of rewards within the group. The advantage $A_{i,j}$ is shared by all tokens in response $y_{i,j}$. Denote the token-level importance ratio as

$$\rho_{i,j,t}(\theta) = \frac{\pi_\theta(y_{i,j,t} \mid c_{i,j,t})}{\pi_{\text{old}}(y_{i,j,t} \mid c_{i,j,t})},$$

where π_{old} denotes the old rollout policy before update. The local GRPO objective is

$$\ell_{i,j,t}^{\text{GRPO}}(\theta) = \min\left(\rho_{i,j,t}(\theta)A_{i,j}, \text{clip}(\rho_{i,j,t}(\theta), 1-\epsilon, 1+\epsilon)A_{i,j}\right).$$

On-Policy Distillation. OPD (Lu & Lab, 2025) distills a teacher policy π_T on trajectories sampled from the current policy π_θ. At each prefix context $c_{i,j,t}$, OPD calculates the log-probability of token $y_{i,j,t}$ and expresses the reverse KL divergence as an expectation under the current policy,

$$\text{KL}(\pi_\theta(\cdot \mid c_{i,j,t}) \,\|\, \pi_T(\cdot \mid c_{i,j,t})) = \mathbb{E}_{y\sim\pi_\theta(\cdot|c_{i,j,t})}\left[\log \pi_\theta(y \mid c_{i,j,t}) - \log \pi_T(y \mid c_{i,j,t})\right].$$

It then uses a single-sample Monte Carlo estimate of the negative reverse KL divergence as the token-level reward

$$r_{i,j,t} = \log \pi_T(y_{i,j,t} \mid c_{i,j,t}) - \log \pi_\theta(y_{i,j,t} \mid c_{i,j,t}).$$

This reward is used by OPD to update the student policy with policy gradient on the states visited by its own rollouts. Thus, the local OPD objective is

$$\ell_{i,j,t}^{\text{OPD}}(\theta) = r_{i,j,t} \log \pi_\theta(y_{i,j,t} \mid c_{i,j,t}).$$

A.3 A First-Order Perspective on Objective Aggregation

To examine how aggregation affects an update, fix the current parameters θ and a batch of sampled responses, and let $\mathcal{U}$ denote the set of token indices $u = (i, j, t)$ in this batch. The effective weight of each token is $q_{i,j,t} = \alpha_i \beta_{i,j} \gamma_{i,j,t}$, with

$$q \in \Delta_{\mathcal{U}} := \left\{ q : q_u \geq 0,\ \sum_{u \in \mathcal{U}} q_u = 1 \right\}.$$

Let F be a target objective differentiable at θ, such as expected reward, and let $g_u = \nabla_\theta \ell_u(\theta)$ be the local gradient at this point. Holding these directions and the aggregation weights fixed during a gradient ascent step gives

$$\theta^+(q) = \theta + \eta G(q), \qquad G(q) = \sum_{u \in \mathcal{U}} q_u g_u,$$

where $\eta > 0$ is the step size. Define the first-order alignment coefficient of unit u as

$$a_u = \langle \nabla F(\theta), g_u \rangle .$$

For each fixed q, differentiability of F yields

$$F\big(\theta^+(q)\big) - F(\theta) = \eta \sum_{u \in \mathcal{U}} q_u a_u + o(\eta), \qquad \eta \to 0^+.$$

Thus, each unit's first-order contribution to the target objective depends on the alignment between its update direction and the target gradient, rather than solely on token counts or response lengths.

Proposition 1 (Local Improvement by Weight Reallocation) *For fixed directions $\{g_u\}_{u \in \mathcal{U}}$ and a target F differentiable at θ, let $q \in \Delta_{\mathcal{U}}$. If two units u, v satisfy $q_v > 0$ and $a_u > a_v$, then there exist $q' \in \Delta_{\mathcal{U}}$ and $\eta_0 > 0$ such that*

$$F\big(\theta^+(q')\big) > F\big(\theta^+(q)\big) \qquad \textit{for every } 0 < \eta < \eta_0.$$

Proof. Choose any fixed $\delta \in (0, q_v)$ and transfer mass δ from v to u, so that $q' \in \Delta_{\mathcal{U}}$ and

$$G(q') - G(q) = \delta(g_u - g_v).$$

Using the first-order expansion for the two fixed weight vectors,

$$\begin{aligned} F\big(\theta^+(q')\big) - F\big(\theta^+(q)\big) &= \eta \,\langle \nabla F(\theta),\, G(q') - G(q) \rangle + o(\eta) \\ &= \eta\delta(a_u - a_v) + o(\eta). \end{aligned}$$

Since $\delta(a_u - a_v) > 0$, the difference is positive for all sufficiently small $\eta > 0$.

For `seq-mean-token-mean` and `token-mean` aggregation, every token in a batch of nonempty responses receives positive weight. Thus, whenever the first-order coefficients differ across tokens, these aggregation rules are not generally locally optimal, since transferring weight between tokens can improve the first-order change while preserving the total weight. This observation serves only to motivate aggregation weights as a design degree of freedom. The level-specific signals in ReAlloc are motivated and examined independently in Secs. 2.3–2.5.

A.4 Analysis of Prompt Level Outcome Variance

For completeness, we provide a more detailed characterization of the prompt level outcome variance used in the main text. For prompt x_i, let the current policy generate $K \geq 2$ rollouts with binary outcome rewards $R_{i,j} \in \{0, 1\}$. Denote the numbers of successful and unsuccessful rollouts by

$$n_i^+ = \sum_{j=1}^{K} R_{i,j}, \qquad n_i^- = K - n_i^+,$$

and define the empirical success rate and outcome variance as

$$\hat{p}_i = \frac{1}{K} \sum_{j=1}^{K} R_{i,j}, \qquad V_i = \frac{1}{K} \sum_{j=1}^{K} (R_{i,j} - \hat{p}_i)^2. \tag{4}$$

We use the variance with denominator K. Since all prompts are sampled with the same number of rollouts, replacing K by $K - 1$ would only rescale all prompt scores by the same positive constant and would therefore leave their ranking unchanged.

Outcome variance captures success and failure coexistence. For binary rewards, Eq. 4 reduces to

$$V_i = \hat{p}_i(1 - \hat{p}_i) = \frac{n_i^+ n_i^-}{K^2}. \tag{5}$$

Equivalently,

$$V_i = \frac{1}{2K^2} \sum_{j=1}^{K} \sum_{k=1}^{K} (R_{i,j} - R_{i,k})^2.$$

These expressions make the interpretation explicit. For fixed K, V_i is proportional to the number $n_i^+ n_i^-$ of success and failure pairs in the sampled group. Hence, $V_i = 0$ when all sampled rollouts share the same outcome, and it is maximized when successes and failures are as balanced as possible. High variance prompts therefore exhibit more observed contrasts between successful and unsuccessful behavior under the current policy. This pairwise view is only an interpretation of the statistic. Our training objective does not explicitly construct pairwise losses or treat individual pairs as independent optimization signals.

Top variance selection under a fixed prompt budget. The same interpretation clarifies the prompt-level selection rule used in our method. Consider a batch of B prompts, and let M denote the number retained under a prescribed retention ratio ρ. With $m_i \in \{0, 1\}$ indicating whether prompt x_i is retained, Top-M variance selection can be written as

$$\max_{m \in \{0,1\}^B} \sum_{i=1}^{B} m_i V_i, \qquad \text{s.t.} \quad \sum_{i=1}^{B} m_i = M.$$

Under this fixed cardinality constraint, the solution is to retain the M prompts with the largest V_i. Moreover, because all prompts use the same rollout count K, Eq. 5 implies that the same selection also maximizes

$$\sum_{i=1}^{B} m_i n_i^+ n_i^-,$$

i.e., the total number of observed success–failure pairs up to the common factor K^{-2}. After selection, the prompt level aggregation weight is

$$\alpha_i = \frac{m_i}{M} = \frac{m_i}{\sum_{k=1}^{B} m_k}.$$

This characterization only describes what Top-M variance selection optimizes with respect to the observed batch statistics. It does not imply that the selected subset necessarily maximizes downstream accuracy or true optimization utility.

Connection to the gradient of expected outcome accuracy. We next relate the empirical statistic to a population quantity relevant to learning. For a fixed prompt x_i, define its expected outcome accuracy under the current policy as

$$p_i(\theta) = \mathbb{E}_{y \sim \pi_\theta(\cdot|x_i)} \left[R_i(y) \right],$$

where $R_i(y) \in \{0, 1\}$. Assume that the reward does not explicitly depend on θ, that the policy support is locally fixed, and that differentiation can be exchanged with expectation. Let

$$s_\theta(y \mid x_i) = \nabla_\theta \log \pi_\theta(y \mid x_i)$$

denote the score of a complete response. For $0 < p_i(\theta) < 1$, define

$$\mu_i^+ = \mathbb{E}[s_\theta(y \mid x_i) \mid R_i(y) = 1], \qquad \mu_i^- = \mathbb{E}[s_\theta(y \mid x_i) \mid R_i(y) = 0].$$

Using the score function identity,

$$\nabla_\theta p_i(\theta) = \mathbb{E}\left[R_i(y) s_\theta(y \mid x_i) \right] = p_i \mu_i^+.$$

Meanwhile, $\mathbb{E}[s_\theta(y \mid x_i)] = 0$, and conditioning on the binary outcome gives

$$p_i \mu_i^+ + (1 - p_i)\mu_i^- = 0.$$

Combining these identities yields

$$\nabla_\theta p_i(\theta) = p_i(1-p_i)\left(\mu_i^+ - \mu_i^-\right). \tag{6}$$

Thus, the population outcome variance $p_i(1-p_i)$ appears explicitly as one factor in the gradient of expected outcome accuracy.

The empirical score V_i is directly related to this population variance. If the K rollouts are sampled independently from the current policy, then

$$\mathbb{E}\left[V_i \mid x_i, \theta\right] = \frac{K-1}{K} p_i(1-p_i).$$

Therefore, V_i estimates the corresponding population variance up to the common multiplicative factor $(K-1)/K$, although finite sample fluctuations may still alter the ranking of individual prompts.

Eq. 6 also clarifies the limitation of this signal. The gradient depends not only on the outcome variance $p_i(1-p_i)$, but also on the separation $\mu_i^+ - \mu_i^-$ between successful and unsuccessful responses. Variance alone therefore does not determine the gradient magnitude or direction, nor the downstream learning gain. Whether higher variance prompts are more useful for continued training remains an empirical question.

A.5 Qualitative Analysis of Rollout Distance



Figure 6: Wordclouds of incorrect rollouts with high and low d^{IC}, which are constructed from the highest (left) and lowest (right) 20% of incorrect rollouts ranked by d^{IC}.

Examples of Degenerate Incorrect Rollouts with High d^{IC}

1. assistant\n

2. assistant\nassistansascus\n\nassistant\nassistant\n\n assistant\n\n assistant\n\n …

3. user\nAtPath Bank predicts its profit from bonds and mortgages according to the following formula: $P(x) = 4x^{2/3} – 300x^{1/3} + 2000$, where P represents the profit in thousands of dollars when x represents the number of bonds sold by Pathat Bank. What is the maximum profit, in thousands of dollars, that Pathat Bank could make from this prediction formula? thúcuser\n thúcuser\n thúcassistant\n\n thúcuser\n thúcuser\n\n thúcuser\n thúcuser\n thúcassistant\n\n thúcuser\n thúcuser\n\n thúcuser\n thúcuser\n\n thúcuser\n thúcuser\n thúcuser\n\n thúcuser\n thúcuser\n thúcassistant\n\n thúcuser\n thúcuser\n\n thúcuser\n thúcuser\n thúcuser\n thúcassistant\n\n thúcuser\n thúcuser\n\n thúcuser\n thúcuser\n thúcuser\n thúcuser\n thúcuser\n thúcuser\n thúcassistant\n\n thúcuser\n thúcuser\n\n thúcuser\n thúcuser\n thúcuser\n thúcuser\n\n thúcuser\n …

4. \n_blocksassistantg\n_blocksassistantc \n_blocksas s s s

Examples of Coherent Incorrect Rollouts with Low d^{IC}

1. Assistant: Let's break down the problem step by step to find the minimal \(N \) such that the 25 checkers can be placed in a row of 25 successive squares but in the reverse order.\n\n### Understanding the Problem:\n- **Initial Setup**: 25 checkers are placed on the 25 leftmost squares of a \(1 \times N \) board.\n- **Allowed Moves**:\n 1. A ...

2. We start by examining the given system of equations:\n\n1. \(x + \frac{1}{y} = 3 \)\n2. \(y + \frac{1}{x} = 4 \)\n\nFirst, let's manipulate the equations to find a suitable expression involving \(xy \). \n\n### Step 1: Express the sum of the equations\nAdd the two equations:\n\[\n\left(x + \frac{1}{y} \right) + \left(y + \frac{1}{x}...

3. Assistant: Alright, I have this problem about Gugu and his napkin. Let's try to understand what's going on step by step.\n\n**Problem Statement:**\n\nGugu has a napkin with ten distinct real numbers. He writes three lines of real numbers on the blackboard:\n\n1. **First Line:** Every number of the form \(a - b \), where \(a \) and \(b ...

4. To solve the problem, we need to find all prime numbers \(p \) such that both \(4p^2 + 1 \) and \(6p^2 + 1 \) are also prime numbers. Let's reason through this step by step:\n\n1. **Base Case Analysis:**\n - For \(p ...

5. Assistant: To determine how many pears Amila needs to give to get 7 strawberries, we'll follow these steps:\n\n### Step 1: Understand the Exchange Rates\nWe have the following exchange ...

A.6 Analysis of the Entropy Peak Score

By construction,

$$S_t > 0 \quad \Longleftrightarrow \quad H_t > H_t^L \text{ and } H_t > H_t^R.$$

Hence, any strict maximum within the local window,

$$H_t > \max_{1 \le k \le w} \{H_{t-k}, H_{t+k}\},$$

necessarily receives a positive score. When $w = 1$, this condition coincides with being a strict local maximum. For $w > 1$, the criterion is more permissive because a token may exceed both neighboring means even when one nearby token has higher entropy. Thus, S_t measures bilateral local prominence rather than strict local maximality.

The score is also invariant to a shared additive shift of entropy within the local neighborhood and equivariant to positive scaling. Specifically, under

$$H'_s = aH_s + b, \qquad a \ge 0,$$

we have

$$S_t(H') = aS_t(H).$$

Therefore, adding a constant offset leaves the score unchanged, while a flat neighborhood receives zero regardless of its absolute entropy level.

For $H_t > 0$, Eq. 2 can be decomposed as

$$S_t = H_t C_t,$$

where

$$C_t = \sqrt{\left[1 - \frac{H_t^L}{H_t}\right]_+ \left[1 - \frac{H_t^R}{H_t}\right]_+}.$$

Since predictive entropy is nonnegative, $0 \le C_t \le 1$. This decomposition separates the absolute uncertainty level H_t from its relative prominence C_t within the local neighborhood.

A.7 Detailed Prompt of Reasoning Transition Annotation

GPT-5.6 Sol Annotation Prompt

Please classify each token into one of the following three categories:
1. **Reasoning transition**:
A meaningful shift in the model's reasoning process, such as a correction, turning point, switch to a different reasoning path, introduction of a new hypothesis, or an attempt at verification or falsification.
2. **Drift / degeneration**:
A deterioration in the reasoning process, where the model becomes less coherent or less relevant, for example through garbled text, repetition, persistent confusion, circular reasoning, or deviation from the original problem.
3. **Ordinary continuation / other**:
A normal continuation of the current reasoning path without a notable transition or degeneration, or behavior that does not clearly fit either of the other categories.

A.8 Implementation Details

Table 6: Training hyperparameters for the ReAlloc GRPO experiment.

Configuration	Value
Model and Training	
Initial model	Qwen3-4B-Base
Training dataset	DAPO-Math-17K
Advantage estimator	grpo
Training batch size	64
PPO mini-batch size	64
PPO micro-batch size per GPU	1
Training epochs	1
Number of GPUs	8
Number of nodes	1
GPU	A100
Model dtype	bfloat16
Learning rate	1×10^{-6}
KL loss	True
KL loss coefficient	1×10^{-4}
KL loss type	low_var_kl
Entropy coefficient	0
Loss aggregation	seq-mean-token-mean
Gradient checkpointing	True
Rollout and Generation	
Inference engine	vLLM
Responses per prompt	8
Max. prompt length	1024
Max. response length	8192
Rollout temperature	1.0
Tensor model parallel size	1
GPU memory utilization	0.8
ReAlloc Configuration	
Prompt filtering enable	True
Prompt filtering Rate	0.5
Rollout weighting enable	True
Rollout weighting temperature	1.5
Token weighting enable	True
Token weighting window	4

Table 7: Training hyperparameters for the ReAlloc OPD experiment.

Configuration	Value
Model and Training	
Initial model	Qwen3-4B-Base
Teacher model	Qwen3-4B-Base-GRPO
Training dataset	DAPO-Math-17K
Advantage estimator	token_reward_direct
Training batch size	128
PPO mini-batch size	128
PPO micro-batch size per GPU	1
Training epochs	1
Number of GPUs	16
Number of nodes	1
GPU	PPU-810E
Model dtype	bfloat16
Learning rate	1×10^{-6}
KL loss	False
Loss aggregation	seq-mean-token-mean
Log-prob top-k	16
Top-k selection strategy	Student top-k
Reward weighting	student_p
Gradient checkpointing	True
Rollout and Generation	
Inference engine	vLLM
Responses per prompt	8
Maximum prompt length	1024
Maximum response length	8192
Rollout temperature	1.0
Teacher temperature	1.0
Repetition penalty	1.0
Tensor model parallel size	1
GPU memory utilization	0.8
ReAlloc Configuration	
Prompt filtering enable	True
Prompt filtering Rate	0.5
Rollout weighting enable	True
Rollout weighting temperature	1.5
Token weighting enable	True
Token weighting window	4

A.9 Detailed Hyperparameter Sensitivity Results

Table 8: Detailed hyperparameter sensitivity Avg. A@8 under OPD and GRPO on Qwen3-4B-Base.

	L-P Prompt Retention Ratio				**L-R Temperature**				**L-T Local Window Size**			
	90%	70%	50%	30%	0.5	1.5	2.5	3.5	1	2	4	8
OPD	31.25	32.00	32.31	30.56	31.99	32.68	32.04	31.78	31.27	31.70	32.35	32.21
GRPO	29.11	30.20	30.75	30.06	29.44	30.80	30.65	30.20	30.08	31.20	31.60	31.48