CoBA-RL: Capability-Oriented Budget Allocation for Reinforcement Learning in LLMs

Anonymous authors
Paper under double-blind review

Abstract

Reinforcement Learning with Verifiable Rewards (RLVR) has emerged as a key approach for enhancing LLM reasoning. However, standard frameworks like Group Relative Policy Optimization (GRPO) typically employ a uniform rollout budget, leading to resource inefficiency. Moreover, existing adaptive methods often rely on instance-level metrics, such as task pass rates, failing to capture the model's dynamic learning state. To address these limitations, we propose CoBA-RL, a reinforcement learning algorithm designed to adaptively allocate rollout budgets based on the model's evolving capability. Specifically, CoBA-RL utilizes a Capability-Oriented Value function to map tasks to their potential training gains and employs a heap-based greedy strategy to efficiently self-calibrate the distribution of computational resources to samples with high training value. Extensive experiments demonstrate that our approach effectively orchestrates the trade-off between exploration and exploitation, delivering consistent generalization improvements across multiple challenging benchmarks. These findings underscore that quantifying sample training value and optimizing budget allocation are pivotal for advancing LLM post-training efficiency.

1 Introduction

Reinforcement Learning with Verifiable Rewards (RLVR) (Lambert et al., 2024) has established itself as a cornerstone for elevating LLM reasoning in agentic, coding, and mathematical domains (Guo et al., 2025; Yang et al., 2025; Comanici et al., 2025; Bai et al., 2025). Within this landscape, Group Relative Policy Optimization (GRPO) (Shao et al., 2024) and its variants (Yu et al., 2025; Zheng et al., 2025a; Liu et al., 2026; Shrivastava et al., 2025) have risen to prominence. By assigning a uniform budget of G rollouts to every prompt to compute group-relative advantages, GRPO eliminates the need for a separate value network. However, ideally, the rollout budgets allocated to each instance should be commensurate with its training value. Intuitively, already mastered samples may offer limited benefit from additional rollouts, whereas samples the model solves only occasionally may benefit from further exploration.

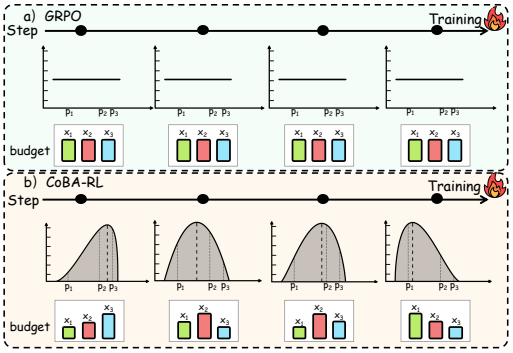

Figure 1: **Comparison between GRPO and CoBA-RL. (a)** GRPO employs a uniform rollout strategy. **(b)** CoBA-RL dynamically adjusts the allocation, directing larger rollout budgets toward instances with higher training value. p_i represents the pass rate of the instance x_i.

In practice, vanilla GRPO overlooks the critical impact of sample difficulty on training value and the corresponding rollout budget. While recent studies have begun to tailor budgets using instance-level metrics related to task difficulty, such as historical pass rates (Li et al., 2025b), these approaches typically rely on static value functions. They operate on the fixed assumption that harder samples inherently offer superior training value than simpler ones and that this relationship remains constant throughout the entire training process. This perspective neglects a crucial reality: the true training

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.

value of a sample is inextricably linked to the policy model's real-time capabilities (Wang et al., 2025b; Zhang et al., 2025b; Wang et al., 2025a).

As the model's capabilities evolve during training, the set of samples holding the highest training value constantly shifts (Hu et al., 2025). To accommodate these variations, the allocation strategy must continuously calibrate the trade-off between exploitation and exploration. Specifically, exploitation involves consolidating mastery over instances where the model already succeeds, whereas exploration requires allocating resources to sample diverse trajectories on challenging queries, thereby expanding the search space to discover potential solutions (Yang et al., 2026; Chen et al., 2025c; Hou et al., 2025; Cui et al., 2025). Consequently, it is imperative to quantify model capability to facilitate policy self-calibration, ensuring the rollout budget is continually re-aligned with the samples most suitable for the current training phase.

To tackle the aforementioned challenges, We propose **CoBA-RL**, a reinforcement learning algorithm that dynamically allocates the rollout budget in accordance with the model's evolving capability. Specifically, we introduce a Capability-Oriented Value function, modeled as a Beta distribution, to map individual instances to their potential training value. By continuously monitoring the global failure rate of the current training batch, we quantify the model's global capability and dynamically calibrate the shape of the value function, as shown in Figure 1. This mechanism autonomously orchestrates the exploration-exploitation trade-off: the value function shifts its high-density regions in real-time to prioritize either consolidating established knowledge or exploring high-uncertainty frontiers based on the model's current competence. Finally, to operationalize this theoretical distribution, we present an Efficient Allocation Optimization algorithm. By formulating the allocation as a constrained maximization problem, we employ a heap-based greedy strategy that iteratively assigns budget to samples offering the highest marginal gain, thereby maximizing the aggregate value of the training batch.

To validate the efficacy of our approach, we conducted extensive experiments using Qwen2.5-7B-Base, Qwen2.5-7B-Instruct, and Qwen3-1.7B/4B-Base models. Empirical results demonstrate that our method significantly outperforms strong baselines across multiple challenging mathematical benchmarks. These findings suggest that, empowered by the Capability-Oriented Value function, CoBA-RL effectively identifies samples holding the high training value at the current training step, thereby achieving a superior trade-off between exploration and exploitation.

Overall, our contributions are summarized as follows:

- We propose a reinforcement learning algorithm that optimizes the exploration-exploitation trade-off by autonomously allocating the rollout budget consistent with the model's evolving capability. By formulating the allocation as a constrained maximization problem, we employ a heap-based greedy strategy to efficiently direct computational resources toward tasks with the highest learning potential.
- We design a Capability-Oriented Value function as the core allocation criterion. This mechanism dynamically quantifies the training value of each task instance conditioned on the policy model's evolving capability.
- Extensive experiments across multiple benchmarks validate the effectiveness of **CoBA-RL**. Our results demonstrate that CoBA-RL significantly outperforms the standard GRPO baseline and alternative rollout allocation strategies, providing an efficient paradigm for Large Language Model post-training.

2 METHOD

In this section, we present CoBA-RL, as illustrated in Figure 2. The framework is composed of two pivotal components. First, the Capability-Oriented Value Function quantifies the policy model's capability via global failure rate statistics. This mechanism self-calibrates the function's shape to discern the model's evolving preferences for specific training samples. Second, the Heap-Based Greedy Budget Allocation is an efficient strategy that optimizes allocation to maximize the cumulative value of the current training batch.

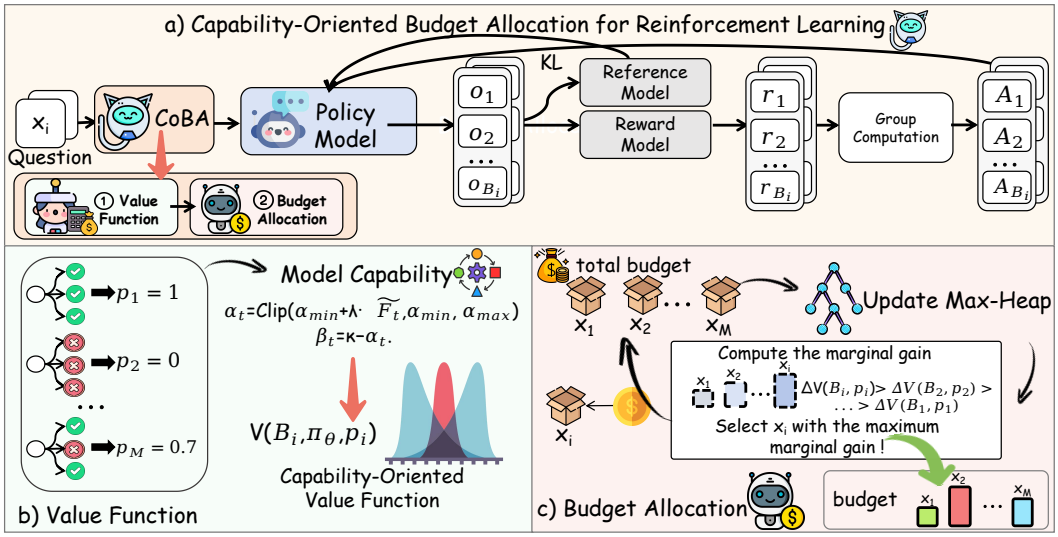

Figure 2: **Overview of CoBA-RL.** (a) Training pipeline: CoBA-RL dynamically allocates a rollout budget B_i per query. (b) Capability-Oriented Value Function: Beta parameters (α_t, β_t) adapt to the global failure rate to quantify training value. (c) Heap-Based Greedy Allocation: a max-heap efficiently allocates the total budget by prioritizing tasks with maximum marginal gain ΔV.

2.1 Preliminaries

The post-training process is formulated as a Reinforcement Learning problem utilizing Group Relative Policy Optimization (GRPO). At each training step t, considering a batch of tasks $\mathcal{X}_t = \{x_1, \ldots, x_M\}$, the policy π_{θ_t} samples a group of G outputs $\mathcal{O}_i = \{o_{i,1}, \ldots, o_{i,G}\}$ for each task x_i.

Given a binary outcome reward function $R(x, o) \in \{0, 1\}$, the task pass rate p_i within this group-based generation framework is naturally modeled as the expected probability of correctness:

$$p_i(x_i; \theta_t) = \mathbb{E}_{o \sim \pi_{\theta_t}(\cdot|x_i)} \left[\mathbb{I}(R(x_i, o) = 1) \right]. \tag{1}$$

The optimization objective is to maximize the following loss:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{x \sim \mathcal{X}_t} \left[\frac{1}{G} \sum_{k=1}^{G} \min \left(\rho_{i,k} A_{i,k}, \text{clip}(\rho_{i,k}, 1-\epsilon, 1+\epsilon) A_{i,k} \right) \right], \tag{2}$$

where $\rho_{i,k}$ denotes the probability ratio between the current and old policies, and $A_{i,k}$ represents the estimated advantage for each output.

2.2 Problem Formulation and Optimization

We now formally define the budget allocation problem. The system operates under a total rollout budget constraint B_{total}. For each task x_i, the policy generates a set of B_i trajectories $\mathcal{O}_i = \{o_{i,1}, \ldots, o_{i,B_i}\}$, subject to $\sum B_i = B_{\text{total}}$.

The objective is to determine the optimal set of budgets $\mathbf{B} = \{B_1, \ldots, B_M\}$ that maximizes the aggregate training gain defined by V. In particular, the function V maps each task x_i and its allocated budget B_i to a scalar value representing the expected learning potential. The optimization problem is formulated as:

$$\begin{aligned} \max_{B_1,\ldots,B_M} \quad & \sum_{i=1}^{M} V(B_i, \pi_\theta, p_i) \\ \text{subject to} \quad & \sum_{i=1}^{M} B_i = B_{\text{total}}, \quad B_{\text{low}} \le B_i \le B_{\text{up}}, \quad \forall i \in \{1,\ldots,M\}, \\ & B_i \in \mathbb{Z}^+. \end{aligned} \tag{3}$$

The primary challenge lies in accurately defining the value function V.This function must dynamically reflect the training potential of each task under the current policy π_θ. Intuitively, as the model masters an task, its value should diminish, thereby allocating the budget to other samples.

2.3 Capability-Oriented Value Function

To identify the training samples with high learning value during the training process, we propose a value function that integrates global capabilities awareness.

Definition 1 (Global Capability)**.** We formally define the model's global capability at step t. The **Global Success Rate** $\mathcal{S}_t$ is defined as the expected pass rate averaged over the current task batch:

$$\mathcal{S}_t = \frac{1}{M} \sum_{i=1}^{M} p_i(x_i; \theta_t). \tag{4}$$

Correspondingly, the **Global Failure Rate** $\mathcal{F}_t$ represents the complementary probability: $\mathcal{F}_t = 1 - \mathcal{S}_t$. These metrics serve as quantitative indicators of the policy model's capability variation; specifically, as the model's capability strengthens, the Global Success Rate $\mathcal{S}_t$ is expected to increase.

2.3.1 Capability-Induced Preference Density

To dynamically adjust the model's preference for samples of varying difficulty, we formalize the **Capability-Induced Preference Density**, which explicitly quantifies the varying degrees of preference that different policy models exhibit toward samples. We model this density as a Beta distribution whose parameters evolve with the training step t. We employ the Beta distribution for its flexible parameterization that adaptively shapes the preference density, facilitating a continuous, self-calibrating shift in sampling focus aligned with the model's evolving capability:

$$\text{Density}(p_i; \alpha_t, \beta_t) = \frac{p_i^{\alpha_t - 1}(1 - p_i)^{\beta_t - 1}}{\mathrm{B}(\alpha_t, \beta_t)} \tag{5}$$

where α_t and β_t are determined by the current global capability metric. Specifically, we use the global failure rate $\mathcal{F}_t$ defined above to characterize the model's training-stage capability. As RL training gradually converges, capability gains naturally plateau and $\mathcal{F}_t$ changes more slowly. Since the Beta shape parameters are driven by $\mathcal{F}_t$, we apply a non-linear transformation $\Psi(\cdot)$ to amplify small failure-rate changes in later stages:

$$\tilde{\mathcal{F}}_t = \Psi(\mathcal{F}_t). \tag{6}$$

The non-linear transformation is further discussed in Appendix F.8. Based on $\tilde{\mathcal{F}}_t$, we determine the shape parameters via a linear mapping, while keeping the sum $\kappa = \alpha_t + \beta_t$ constant:

$$\alpha_t = \text{clip}\left(\alpha_{\min} + \lambda \cdot \tilde{\mathcal{F}}_t,\ \alpha_{\min},\ \alpha_{\max}\right), \quad \beta_t = \kappa - \alpha_t. \tag{7}$$

Here, $\lambda = \alpha_{\max} - \alpha_{\min}$, with $\alpha_{\min} = 1$ and $\alpha_{\max} = \kappa - 1$. This dynamic mechanism provides an adaptive way to regulate the preference density during training. At a high level, in early training stages characterized by a high $\tilde{\mathcal{F}}_t$, the distribution skews toward high pass-rate samples to rapidly acquire training signals, and gradually shifts toward harder samples as capability improves. Importantly, this

mechanism responds to capability fluctuations by automatically recalibrating the preference density, ensuring that the budget is allocated to the sample interval that best consolidates the model's current capability, rather than blindly pursuing high-difficulty tasks.

2.3.2 BUDGET SATURATION FACTOR

While the Capability-Induced Preference Density identifies which tasks are theoretically pivotal, the actual training gain derived from a task instance x_i is intrinsically coupled with the allocated computational budget B_i. Intuitively, increasing B_i yields higher returns, but this gain follows the law of diminishing returns. To quantify this relationship, we design the **Budget Saturation Factor**:

$$\eta(B_i, p_i) = 1 - e^{-\frac{B_i}{\tau} p_i (1-p_i)}, \tag{8}$$

where τ is a temperature coefficient that controls the velocity at which the value reaches saturation.

By synthesizing the budget constraint with the Capability-Induced Preference Density, we formulate the final **Capability-Oriented Value Function** as:

$$V(B_i, \pi_\theta, p_i) = \left(1 - e^{-\frac{B_i}{\tau} p_i (1-p_i)}\right) \cdot \text{Density}(p_i; \alpha_t, \beta_t). \tag{9}$$

The *Budget Saturation Factor* acts as a realizability coefficient, forcing the optimization of B_i to conform to the shape of the *Capability-Induced Preference Density*. Consequently, as the policy model's capability evolves, thereby altering α_t and β_t, the objective function V dynamically shifts its topology. This ensures that the budget allocation strategy is not static, but continuously realigns itself with the model's changing proficiency throughout the training trajectory. Through this value function, we can adaptively map different task instances to a value based on the current model capability and the allocated rollout budget.

2.4 EFFICIENT ALLOCATION OPTIMIZATION VIA HEAP-BASED GREEDY STRATEGY

The optimization of the objective function defined in Eq. 3 relies on the mathematical property of the proposed value function.

Proposition 1. The marginal gain of the value function is strictly monotonically decreasing with respect to the allocated budget B_i. That is, defining the marginal gain as $\Delta V(B_i, p_i) = V(B_i + 1, p_i) - V(B_i, p_i)$, the following inequality holds for all $B_i \geq 0$:

$$\Delta V(B_i, p_i) > \Delta V(B_i + 1, p_i). \tag{10}$$

The proof is provided in the Appendix A. This diminishing marginal utility property guarantees that a greedy allocation strategy yields an optimal solution for the discrete resource allocation problem. Consequently, we design an efficient **Heap-Based Greedy Strategy**. Specifically, we maintain a max-heap of marginal gains for all tasks. In each iteration, we extract the task with the highest potential gain from the heap top, allocate a unit rollout budget, update its state, and re-insert it into the heap. The detailed procedure is outlined in Algorithm 1.

Notably, this strategy operates based on the relative magnitude of marginal gains within the current training batch. This facilitates a dual-adaptive allocation mechanism where resource distribution naturally shifts in response to the evolving shape of the value function—reflecting the shifting preferences of the current policy π_θ—while simultaneously adjusting to the specific variations of the samples in the batch. Such a design guarantees that the computational budget is dynamically channeled toward the instances offering the greatest learning potential.

3 EXPERIMENTS

3.1 EXPERIMENTAL SETUP

We initialize our policy models using Qwen2.5-7B-Instruct (Qwen et al., 2025), Qwen2.5-7B-Base (Qwen et al., 2025) and Qwen3-1.7B/4B-Base (Yang et al., 2025). For training, we utilize DAPO-Math-17K (Yu et al., 2025), a dataset widely adopted for mathematical reasoning tasks. We fix the total rollout budget to $\sum_{i=1}^{M} B_i = B_{\text{total}} = MG$, with $G = 16$.

Table 1: Performance comparison of GRPO, Knapsack-RL, and CoBA-RL across different models and benchmarks. All results are reported in percentage (%). The best results for each model are highlighted in **bold**. Superscripts denote the difference relative to the GRPO baseline (Blue for Knapsack-RL, Red for CoBA-RL).

Method	AIME24	AIME25	AMC23	MATH500	OLYMPIAD	Avg
Qwen2.5-7B-Instruct						
GRPO	14.17	12.71	69.84	76.78	37.68	42.24
Knapsack-RL	18.54$^{+4.37}$	15.21$^{+2.50}$	71.41$^{+1.57}$	**80.55**$^{+3.77}$	41.23$^{+3.55}$	45.39$^{+3.15}$
CoBA-RL	**19.16**$^{+4.99}$	**18.33**$^{+5.62}$	**75.16**$^{+5.32}$	**80.55**$^{+3.77}$	**43.27**$^{+5.59}$	**47.29**$^{+5.05}$
Qwen2.5-7B-Base						
GRPO	15.41	13.33	75.00	77.63	37.03	43.68
Knapsack-RL	19.58$^{+4.17}$	**16.67**$^{+3.34}$	74.37$^{-0.63}$	79.73$^{+2.10}$	41.33$^{+4.30}$	46.34$^{+2.66}$
CoBA-RL	**21.04**$^{+5.63}$	16.04$^{+2.71}$	**76.71**$^{+1.71}$	**80.23**$^{+2.60}$	**43.11**$^{+6.08}$	**47.43**$^{+3.75}$
Qwen3-4B-Base						
GRPO	18.54	15.62	65.62	81.19	42.61	44.72
Knapsack-RL	21.88$^{+3.34}$	20.01$^{+4.39}$	65.47$^{-0.15}$	80.51$^{-0.68}$	45.42$^{+2.81}$	46.66$^{+1.94}$
CoBA-RL	**22.71**$^{+4.17}$	**21.16**$^{+5.54}$	**72.34**$^{+6.72}$	**84.29**$^{+3.10}$	**46.78**$^{+4.17}$	**49.46**$^{+4.74}$
Qwen3-1.7B-Base						
GRPO	8.96	5.00	44.69	69.46	29.92	31.61
Knapsack-RL	12.50$^{+3.54}$	4.79$^{-0.21}$	45.16$^{+0.47}$	69.69$^{+0.23}$	30.68$^{+0.76}$	32.56$^{+0.95}$
CoBA-RL	**16.25**$^{+7.29}$	**7.29**$^{+2.29}$	**49.84**$^{+5.15}$	**72.85**$^{+3.39}$	**32.52**$^{+2.60}$	**35.75**$^{+4.14}$

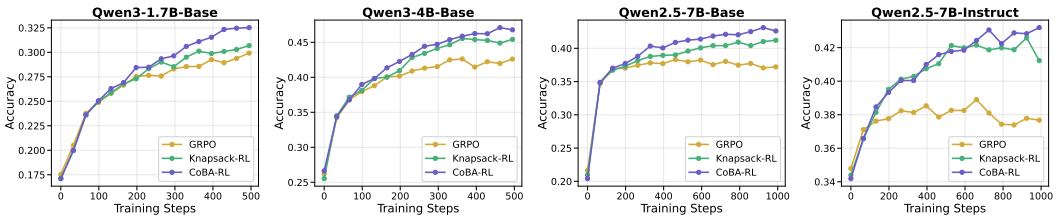

Figure 3: Performance comparison of different models and methods on the Olympiad benchmark (avg@16). The curves track the validation accuracy over training steps for GRPO, Knapsack, and CoBA-RL across varying model scales.

We evaluate performance on five challenging benchmarks: AIME24 (Zhang & Math-AI, 2024), AIME25 (Zhang & Math-AI, 2025), AMC23, MATH500 (Lightman et al., 2024), and OLYMPIAD Bench (He et al., 2024). To benchmark the efficacy of our dynamic budget allocation, we compare CoBA-RL against two primary baselines: GRPO (Shao et al., 2024) and Knapsack-RL (Li et al., 2025b). For fair comparison, all reported results are evaluated using the avg@16 metric. Comprehensive implementation details and additional experimental results are provided in Appendix E and F.

3.2 Main Results

As shown in Table 1, CoBA-RL consistently improves average accuracy across all four models. On Qwen2.5-7B-Instruct, it achieves 47.29%, surpassing GRPO and Knapsack-RL by 5.05 and 1.90 percentage points, respectively. On individual benchmarks, it reaches 18.33% on AIME25 and 75.16% on AMC23, improving over GRPO by 5.62 and 5.32 percentage points. CoBA-RL also achieves an average accuracy of 47.43% on Qwen2.5-7B-Base, including a 1.78-point gain over

Table 2: Performance comparison of Exploration-Exploitation strategies on Qwen2.5-7B-Instruct.

Strategy	**AIME24**	**AIME25**	**AMC23**	**MATH500**	**OLYMPIAD**	**Avg**
Explore → Exploit	16.87	10.41	73.43	79.93	41.84	44.50
Exploit → Explore (Ours)	**19.16**	**18.33**	**75.16**	**80.55**	**43.27**	**47.29**

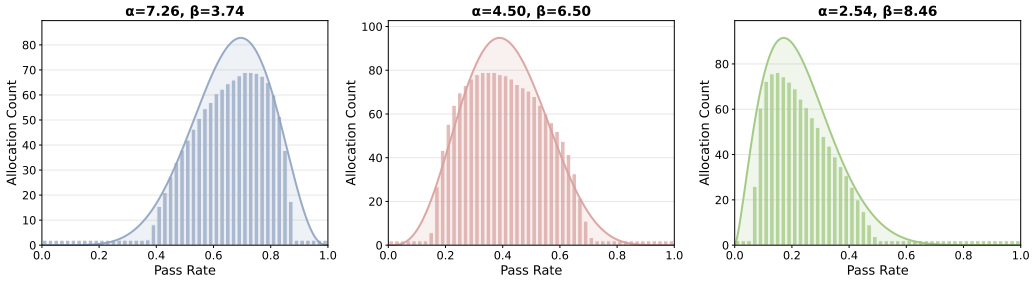

Figure 4: Visualization of different budget allocation across varying model capabilities.

Knapsack-RL on OlympiadBench. On Qwen3-4B-Base and Qwen3-1.7B-Base, the average gains over GRPO are 4.74 and 4.14 percentage points, respectively, demonstrating consistent benefits across model sizes. Figure 3 presents the OlympiadBench training curves across these models.

To intuitively validate our allocation mechanism, we visualize the resulting budget distributions under different value functions in Figure 4. As depicted, the allocated budget dynamically shifts in response to the model's evolving capability, conforming to the geometric shape of the specified value functions.

3.3 The Relationship between Exploration and Exploitation in Reinforcement Learning

We investigate how capability feedback shapes the balance between exploration and exploitation during reinforcement learning. Specifically, we conducted a comparative experiment on the Qwen2.5-7B-Instruct model to evaluate the performance differences between an "Explore → Exploit" strategy and our proposed "Exploit → Explore" strategy. The evolution of α_t for both methods is visualized in Figure 5. The overall Exploit-to-Explore trend emerges from capability feedback rather than a predefined training schedule. The allocation focus is recalibrated from observed failure rates and can reverse when capability regresses, as reflected in the local fluctuations of α_t.

As presented in Table 2, the results demonstrate that the "Exploit → Explore" strategy yields superior overall performance. Specifically, our method achieves a substantial improvement on the AIME25 benchmark, boosting accuracy from 10.41% to 18.33%, and attains the highest average score of 47.29%. This empirical evidence supports our hypothesis that prioritizing the exploitation of simple samples in the early stages facilitates the rapid consolidation of foundational capabilities, whereas allocating the budget to explore difficult instances in later stages effectively expands the model's solution space.

3.4 Comparative Analysis against Static and Linear Decay Baselines

Our approach adapts to evolving model capabilities during reinforcement learning (RL) by mapping capability fluctuations to an adaptive value function for resource allocation. To verify the efficacy of this dynamic mechanism, we benchmark it against three baselines: two static strategies with fixed value functions, $(\alpha, \beta) = (10.5, 1.5)$ for exploitation and $(1.5, 10.5)$ for exploration, and a linear step decay strategy where α_t decreases stepwise from 10 to 1 over training. Visualizations are provided in Appendix F.11.

The quantitative results are presented in Table 3. As observed, our dynamic budget allocation strategy consistently outperforms both the static strategies and the Linear Step Decay strategy. CoBA-RL achieves the highest average accuracy of 47.29%, surpassing the 45.39% performance

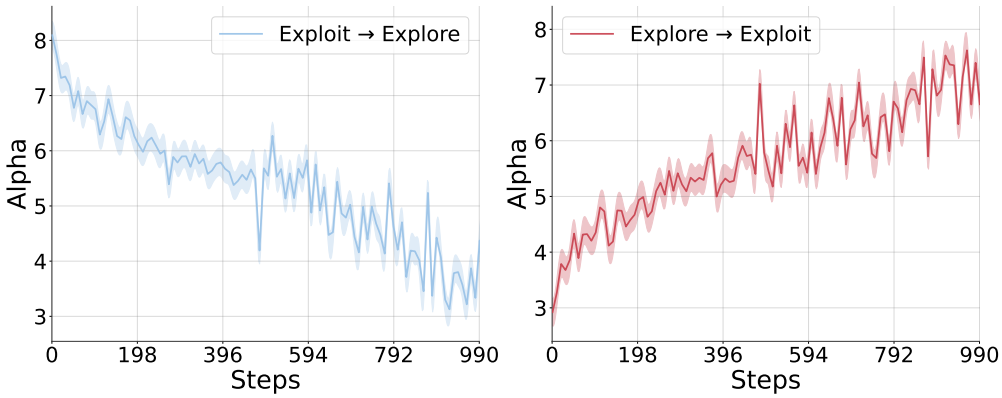

Figure 5: Evolution of α_t on Qwen2.5-7B-Instruct. **Left:** The "Exploit → Explore" strategy, where α_t exhibits a fluctuating downward trend. **Right:** The "Explore → Exploit" strategy, where α_t shows a fluctuating upward trend.

Table 3: Quantitative comparison against static strategies and linear step decay baselines. The best results are highlighted in **bold**.

Method	AIME24	AIME25	AMC23	MATH500	OLYMPIAD	Avg
$(\alpha = 10.5, \beta = 1.5)$	15.20	15.21	68.28	80.17	42.26	44.22
$(\alpha = 1.5, \beta = 10.5)$	18.12	16.45	70.31	79.92	41.25	45.21
Linear Step Decay	17.08	16.85	70.77	79.81	42.43	45.39
CoBA-RL (Ours)	**19.16**	**18.33**	**75.16**	**80.55**	**43.27**	**47.29**

of Linear Step Decay and the 45.21% accuracy of the best static baseline by margins of 1.90% and 2.08%, respectively. This validates that static or pre-determined schedules struggle to align with the unpredictable dynamics of real learning compared to our adaptive approach.

3.5 Comparison with Recent Adaptive Baselines

We include four additional recent baselines: SEC (Chen et al., 2025b), GVM-RAFT (Yao et al., 2025), DynaMO (Fang et al., 2026), and VIP (Nguyen et al., 2026). Together with GRPO and Knapsack-RL, these methods cover recent adaptive allocation and curriculum strategies. All methods use Qwen2.5-7B-Instruct with the same training data, optimization settings, number of update steps, and total budget of 4,096 rollout trajectories per step. As shown in Table 4, CoBA-RL achieves the highest Avg@16 of 47.29%, surpassing the strongest competing baseline, VIP, which achieves 46.67%, by 0.62 percentage points. CoBA-RL also outperforms SEC by 1.86 percentage points, compared with its Avg@16 of 45.43%, demonstrating its effectiveness relative to adaptive curriculum selection under the same rollout budget.

Table 4: Comparison with recent adaptive allocation and curriculum baselines on Qwen2.5-7B-Instruct.

Method	GRPO	Knapsack-RL	SEC	GVM-RAFT	DynaMO	VIP	**CoBA-RL**
Avg@16	42.24	45.39	45.43	45.68	45.95	46.67	**47.29**

3.6 Runtime Efficiency of Budget Allocation

To validate the computational efficiency of our allocation algorithm, we compare the Heap-Based Greedy strategy with a standard Dynamic Programming baseline in Table 5. Since both methods yield the same allocation under the diminishing marginal returns property proven in Appendix A, we focus solely on runtime.

Table 5: Runtime comparison of budget allocation methods.

	DP	Heap Greedy	Speedup
Time (s)	115.05	**0.124**	$\sim 928\times$

The Dynamic Programming baseline has pseudo-polynomial complexity $O(M \cdot B_{\text{total}} \cdot (B_{\text{up}} - B_{\text{low}}))$, making its cost scale with batch size, total budget, and allocation range. For a large setting with $M = 512$ and $B_{\text{total}} = 8192$, it requires 115.05 seconds, which is impractical for online RL training. In contrast, our Heap-Based Greedy strategy reduces the complexity to $O(B_{\text{total}} \log M)$ and completes allocation in only 0.124 seconds, enabling efficient integration into large-scale training pipelines.

4 Related Work

4.1 Reinforcement Learning for LLMs

Methods based on Reinforcement Learning with Verification and Reasoning (RLVR) have proven effective in enhancing the reasoning capabilities of LLMs (Dai et al., 2025; Zheng et al., 2025c;b; Jaech et al., 2024; Xie et al., 2025). Among them, GRPO (Shao et al., 2024) has been widely adopted due to its effectiveness and efficiency. Building on this, GSPO (Zheng et al., 2025a) defines sample importance based on sequence likelihood, while DAPO (Yu et al., 2025) introduces four distinct techniques to bolster reinforcement learning performance. However, these predominantly group-based mechanisms often overlook the inherent variability across different tasks, inevitably leading to a significant waste of rollout resources.

4.2 Progressive Training and Resource Allocation

Progressive strategies like curriculum learning enhance performance by organizing data into easy-to-hard stages through sample selection (Shi et al., 2025; Wu et al., 2025; Naïr et al., 2024; Deng et al., 2025; Li et al., 2025a; Zhang et al., 2025a) For example, SEC (Chen et al., 2025b) adjusts data distributions using policy gradients. However, while these methods focus on selecting which samples to train, our approach adaptively allocates varying budgets per sample.

We draw inspiration from budget allocation, a fundamental problem in operations research (Hussain et al., 2013) with broad applications in online advertising and marketing (Liu et al., 2020; Chen et al., 2025a; Cai et al., 2023), where bandit-based methods are often used to optimize resource distribution under uncertainty (Ge et al., 2025). In LLMs, ROI-Reasoning (Zhao et al., 2026) similarly formulates inference under limited token budgets as an ordered stochastic multiple-choice knapsack problem.

Despite these advancements, dynamically allocating rollout budgets in RL to balance exploration and exploitation remains challenging. While approaches like GVM-RAFT (Yao et al., 2025) minimize gradient variance via rejection sampling, and Knapsack-RL (Li et al., 2025b) maximizes batch value via a knapsack formulation, they are limited by their reliance on static or pre-defined value functions. As a result, they lack a mechanism to explicitly align rollout budget allocation with samples that hold high training value under the model's real-time capability.

5 Conclusion

We propose CoBA-RL, a reinforcement learning algorithm that dynamically adapts rollout budget allocation to evolving LLM capabilities by formulating the task as a constrained optimization problem. Importantly, this algorithm allows for seamless integration into existing RL pipelines. To precisely quantify the learning value of different samples during training, we introduce a Capability-Oriented Value function modeled via a dynamic Beta distribution, employing a heap-based greedy strategy to iteratively maximize marginal gains. Experiments demonstrate that CoBA-RL achieves a superior exploration-exploitation trade-off. Our analysis further reveals that effective training typically transitions from exploitation to exploration, highlighting that accurately assessing sample value and optimizing budget allocation are critical avenues for efficient LLM post-training.

AI USE STATEMENT

In preparing this manuscript, we used generative AI tools solely for language polishing, including correcting grammatical errors, refining wording, and improving the clarity and readability of text written by the authors. These tools were not used to develop research ideas or design methods or experiments. The authors reviewed all AI-assisted edits to ensure that they preserved the intended meaning and technical content. The authors take full responsibility for the accuracy, originality, and integrity of the final manuscript.

ETHICS STATEMENT

This work studies adaptive rollout budget allocation for reinforcement learning with verifiable rewards using publicly available models and benchmarks. Our study does not involve human participants or the collection of personal data. Applications of CoBA-RL should include appropriate human oversight. The underlying models and datasets should be used in accordance with their licenses and usage restrictions.

REFERENCES

Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, et al. Kimi K2: open agentic intelligence. *CoRR*, abs/2507.20534, 2025. doi: 10.48550/ARXIV.2507.20534. URL https://doi.org/10.48550/arXiv.2507.20534.

Tianchi Cai, Jiyan Jiang, Wenpeng Zhang, Shiji Zhou, Xierui Song, et al. Marketing budget allocation with offline constrained deep reinforcement learning. In Tat-Seng Chua, Hady W. Lauw, Luo Si, Evimaria Terzi, and Panayiotis Tsaparas (eds.), *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, WSDM 2023, Singapore, 27 February 2023 - 3 March 2023*, pp. 186–194. ACM, 2023. doi: 10.1145/3539597.3570486. URL https://doi.org/10.1145/3539597.3570486.

Huijun Chen, Ying-Ju Chen, Sung-Hyuk Park, and Dongwook Shin. Multichannel advertising: Budget allocation in the presence of spillover and carryover effects. *Manufacturing & Service Operations Management*, 27(3):862–880, 2025a.

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*, 2025b.

Zhipeng Chen, Xiaobo Qin, Youbin Wu, Yue Ling, Qinghao Ye, Wayne Xin Zhao, and Guang Shi. Pass@ k training for adaptively balancing exploration and exploitation of large reasoning models. *arXiv preprint arXiv:2508.10751*, 2025c.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

Runpeng Dai, Linfeng Song, Haolin Liu, Zhenwen Liang, Dian Yu, Haitao Mi, Zhaopeng Tu, Rui Liu, Tong Zheng, Hongtu Zhu, et al. Cde: Curiosity-driven exploration for efficient reinforcement learning in large language models. *arXiv preprint arXiv:2509.09675*, 2025.

Huilin Deng, Ding Zou, Rui Ma, Hongchen Luo, Yang Cao, and Yu Kang. Boosting the generalization and reasoning of vision language models with curriculum reinforcement learning. *arXiv preprint arXiv:2503.07065*, 2025.

Yangyi Fang, Jiaye Lin, Xiaoliang Fu, Cong Qin, Haolin Shi, Chaowen Hu, Lu Pan, Ke Zeng, and Xunliang Cai. How to allocate, how to learn? dynamic rollout allocation and advantage modulation for policy optimization, 2026. URL https://arxiv.org/abs/2602.19208.

Lin Ge, Yang Xu, Jianing Chu, David Cramer, Fuhong Li, Kelly Paulson, and Rui Song. Multi-task combinatorial bandits for budget allocation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 2247–2258, 2025.

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.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *ACL (1)*, pp. 3828–3850. Association for Computational Linguistics, 2024.

Zhenyu Hou, Xin Lv, Rui Lu, Jiajie Zhang, Yujiang Li, Zijun Yao, Juanzi Li, Jie Tang, and Yuxiao Dong. Advancing language model reasoning through reinforcement learning and inference scaling. *arXiv preprint arXiv:2501.11651*, 2025.

Zengjie Hu, Jiantao Qiu, Tianyi Bai, Haojin Yang, Binhang Yuan, Qi Jing, Conghui He, and Wentao Zhang. Vade: Variance-aware dynamic sampling via online sample-level difficulty estimation for multimodal rl. *arXiv preprint arXiv:2511.18902*, 2025.

Hameed Hussain, Saif Ur Rehman Malik, Abdul Hameed, Samee Ullah Khan, Gage Bickler, Nasro Min-Allah, Muhammad Bilal Qureshi, Limin Zhang, Wang Yongji, Nasir Ghani, et al. A survey on resource allocation in high performance distributed computing systems. *Parallel Computing*, 39(11):709–736, 2013.

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.

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.

Renda Li, Hailang Huang, Fei Wei, Feng Xiong, Yong Wang, and Xiangxiang Chu. Adacurl: Adaptive curriculum reinforcement learning with invalid sample mitigation and historical revisiting. *arXiv preprint arXiv:2511.09478*, 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 *ICLR*. OpenReview.net, 2024.

Mengjuan Liu, Wei Yue, Lizhou Qiu, and Jiaxing Li. An effective budget management framework for real-time bidding in online advertising. *IEEE access*, 8:131107–131118, 2020.

Shih-Yang Liu, Xin Dong, Ximing Lu, et al. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization. *arXiv preprint arXiv:2601.05242*, 2026.

Marwa Naïr, Kamel Yamani, Lynda Lhadj, and Riyadh Baghdadi. Curriculum learning for small code language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pp. 390–401, 2024.

Trung Hieu Nguyen, Bao Nguyen, Wenao Ma, Yuzhi Zhao, Ruifeng She, and Viet Anh Nguyen. Adaptive rollout allocation for online reinforcement learning with verifiable rewards. In *International Conference on Learning Representations*, volume 2026, pp. 136100–136132, 2026.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, et al. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

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.

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. *arXiv preprint arXiv: 2409.19256*, 2024.

Taiwei Shi, Yiyang Wu, Linxin Song, Tianyi Zhou, and Jieyu Zhao. Efficient reinforcement finetuning via adaptive curriculum learning. *arXiv preprint arXiv:2504.05520*, 2025.

Vaishnavi Shrivastava, Ahmed Awadallah, Vidhisha Balachandran, Shivam Garg, Harkirat Behl, and Dimitris Papailiopoulos. Sample more to think less: Group filtered policy optimization for concise reasoning. *arXiv preprint arXiv:2508.09726*, 2025.

Qinsi Wang, Jinghan Ke, Hancheng Ye, Yueqian Lin, Yuzhe Fu, Jianyi Zhang, Kurt Keutzer, Chenfeng Xu, and Yiran Chen. Angles don't lie: Unlocking training-efficient rl through the model's own signals, 2025a. URL https://arxiv.org/abs/2506.02281.

Zhenting Wang, Guofeng Cui, Yu-Jhe Li, Kun Wan, and Wentian Zhao. Dump: Automated distribution-level curriculum learning for rl-based llm post-training, 2025b. URL https://arxiv.org/abs/2504.09710.

Muling Wu, Qi Qian, Wenhao Liu, Xiaohua Wang, Zisu Huang, Di Liang, LI Miao, Shihan Dou, Changze Lv, Zhenghua Wang, et al. Progressive mastery: Customized curriculum learning with guided prompting for mathematical reasoning. *arXiv preprint arXiv:2506.04065*, 2025.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025. doi: 10.48550/ARXIV.2505.09388. URL https://doi.org/10.48550/arXiv.2505.09388.

Fengkai Yang, Zherui Chen, Xiaohan Wang, Xiaodong Lu, Jiajun Chai, Guojun Yin, Wei Lin, Shuai Ma, Fuzhen Zhuang, Deqing Wang, et al. Your group-relative advantage is biased. *arXiv preprint arXiv:2601.08521*, 2026.

Jiarui Yao, Yifan Hao, Hanning Zhang, Hanze Dong, Wei Xiong, Nan Jiang, and Tong Zhang. Optimizing chain-of-thought reasoners via gradient variance minimization in rejection sampling and rl. *arXiv preprint arXiv:2505.02391*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, et al. DAPO: an open-source LLM reinforcement learning system at scale. *CoRR*, abs/2503.14476, 2025. doi: 10.48550/ARXIV.2503.14476. URL https://doi.org/10.48550/arXiv.2503.14476.

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.

Xuemiao Zhang, Xu Liangyu, Feiyu Duan, Yongwei Zhou, Sirui Wang, Rongxiang Weng, Jingang Wang, and Xunliang Cai. Preference curriculum: Llms should always be pretrained on their preferred data. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 21181–21198, 2025b.

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.

Muyang Zhao, Qi Qi, and Hao Sun. Roi-reasoning: Rational optimization for inference via pre-computation meta-cognition. *arXiv preprint arXiv:2601.03822*, 2026.

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. *CoRR*, abs/2507.18071, 2025a. doi: 10.48550/ARXIV.2507.18071. URL https://doi.org/10.48550/arXiv.2507.18071.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37: 62557–62583, 2024.

Tong Zheng, Lichang Chen, Simeng Han, R Thomas McCoy, and Heng Huang. Learning to reason via mixture-of-thought for logical reasoning. *arXiv preprint arXiv:2505.15817*, 2025b.

Tong Zheng, Hongming Zhang, Wenhao Yu, Xiaoyang Wang, Runpeng Dai, Rui Liu, Huiwen Bao, Chengsong Huang, Heng Huang, and Dong Yu. Parallel-r1: Towards parallel thinking via reinforcement learning. *arXiv preprint arXiv:2509.07980*, 2025c.

A Proof of Proposition 1

Recall the definition of the value function given in Eq. 9:

$$V(B_i, \pi_\theta, p_i) = \left(1 - e^{-\frac{B_i}{\tau} p_i (1-p_i)}\right) \cdot \text{Density}(p_i; \alpha_t, \beta_t). \tag{11}$$

For simplicity, let us denote the terms independent of B_i as constants C and k. Let:

$$C = \text{Density}(p_i; \alpha_t, \beta_t) = \frac{p_i^{\alpha_t - 1}(1-p_i)^{\beta_t - 1}}{\mathrm{B}(\alpha_t, \beta_t)}, \tag{12}$$

$$k = \frac{p_i(1-p_i)}{\tau}. \tag{13}$$

We consider the non-trivial case where $p_i \in (0,1)$ and $\tau > 0$, implying that $C > 0$ and $k > 0$. The value function can be rewritten as:

$$V(B_i, \pi_\theta, p_i) = C\left(1 - e^{-kB_i}\right). \tag{14}$$

Step 1: Deriving the Marginal Gain. The marginal gain $\Delta V(B_i, p_i)$ represents the increment in value obtained by increasing the budget by one unit:

$$\begin{aligned}
\Delta V(B_i, p_i) &= V(B_i + 1, \pi_\theta, p_i) - V(B_i, \pi_\theta, p_i) \\
&= C\left(1 - e^{-k(B_i+1)}\right) - C\left(1 - e^{-kB_i}\right) \\
&= C\left(e^{-kB_i} - e^{-k(B_i+1)}\right) \\
&= Ce^{-kB_i}\left(1 - e^{-k}\right).
\end{aligned} \tag{15}$$

Let $A = C(1 - e^{-k})$. Since $k > 0$, we have $e^{-k} < 1$, thus $(1 - e^{-k}) > 0$. Combined with $C > 0$, it follows that $A > 0$. The marginal gain simplifies to:

$$\Delta V(B_i, p_i) = A \cdot e^{-kB_i}. \tag{16}$$

Step 2: Proving Monotonicity. To prove that the marginal gain is strictly decreasing, we compare $\Delta V(B_i, p_i)$ with $\Delta V(B_i + 1, p_i)$. We examine the ratio between consecutive marginal gains:

$$\frac{\Delta V(B_i + 1, p_i)}{\Delta V(B_i, p_i)} = \frac{A \cdot e^{-k(B_i+1)}}{A \cdot e^{-kB_i}} = \frac{e^{-k} \cdot e^{-kB_i}}{e^{-kB_i}} = e^{-k}. \tag{17}$$

Since $k > 0$, we have:

$$e^{-k} < 1. \tag{18}$$

Therefore:

$$\frac{\Delta V(B_i + 1, p_i)}{\Delta V(B_i, p_i)} < 1 \quad \Longrightarrow \quad \Delta V(B_i + 1, p_i) < \Delta V(B_i, p_i). \tag{19}$$

This confirms that the marginal gain $\Delta V(B_i, p_i)$ is a strictly decreasing geometric sequence with respect to the budget B_i.

B Case Study on the Necessity of Global Capability Signal

We provide a case study to illustrate why per-sample pass rate alone is insufficient for rollout budget allocation. During RL training, we observe a phenomenon of *fragile success*: a model may achieve a high pass rate on one instance while still failing on conceptually similar problems.

At one training step where the global failure rate is increasing, the model solves the following solid geometry problem with pass rate $p = 0.65$:

> Tetrahedron $ABCD$ has $AB = 5$, $AC = 3$, $BC = 4$, $BD = 4$, $AD = 3$, and $CD = \frac{12}{5}\sqrt{2}$. Find the volume.

Table 6: A case study of fragile success in solid geometry reasoning.

Similar Geometry Problem	Pass Rate
Regular tetrahedron with an interior point; sum of distances to faces	0.08
Rectangle $ABCD$ folded into a triangular pyramid; find volume	0.00
Convex quadrilateral $ABCD$ with diagonals intersecting at P; find area	0.00

However, at nearby steps, the model still fails on several similar geometry problems, as shown in Table 6.

This case suggests that the high pass rate on the original instance reflects fragile success rather than genuine mastery. A method relying only on per-sample pass rate may prematurely reduce the budget for this sample. In contrast, CoBA-RL uses the global failure rate $\mathcal{F}_t$ to capture the model's overall capability state, allowing the value function to maintain sufficient budget for such samples when the model's global capability is still weak or declining.

Therefore, $\mathcal{F}_t$ and p_i play complementary roles: $\mathcal{F}_t$ calibrates which difficulty region should be emphasized at the current training stage, while p_i determines the relative allocation among samples within the current batch.

C Implementation Details of Main Training Loop

Listing 1 illustrates how the CoBA-RL BudgetAllocator is integrated into the GRPO training loop. The core logic involves calculating the specific rollout count for each prompt in the current batch via a dictionary mapping and resampling the batch accordingly before the generation phase.

```
# Inside the GRPO training loop
for batch_dict in dataloader:
    # Extract Batch
    batch = process_batch(batch_dict)
    indices = batch["index"]

    # Execute Budget Allocation
    # Returns a dictionary: {index: count}
    total_budget = batch_size * default_group_size
    allocation_dict = budget_allocator.allocate(
        indices=indices,
        total_budget=total_budget
    )

    # Expand batch according to allocation counts
    repeat_indices = []
    for idx, count in allocation_dict.items():
        if count > 0:
            repeat_indices.extend([idx] * count)

    batch = batch[repeat_indices]

    outputs = actor.generate_sequences(batch)
    # ... GRPO loss calculation and update ...
```

Listing 1: Core logic for Budget Allocation integration in GRPO.

D Heap-Based Greedy Budget Allocation

The complete budget-allocation procedure is summarized in Algorithm 1. We first assign the minimum rollout budget B_{low} to every task and insert each task's next marginal gain into a max-heap. We then repeatedly allocate one additional rollout to the task with the largest marginal gain and update its heap entry until the total budget is exhausted or all tasks reach B_{up}.

Algorithm 1 Heap-Based Greedy Budget Allocation

Input: Task set $\{x_i\}_{i=1}^{M}$, pass rates $p = \{p_i\}$, total budget B_{total}, constraints $B_{\text{low}}, B_{\text{up}}$.
Output: Optimal budget allocation $\mathbf{B} = \{B_1, \ldots, B_M\}$.
Initialize $B_i \leftarrow B_{\text{low}}$ for all $i \in \{1, \ldots, M\}$.
$R \leftarrow B_{\text{total}} - \sum_{i=1}^{M} B_{\text{low}}$
Construct Max-Heap $\mathcal{H}$:
for $i = 1$ to M **do**
    if $B_i < B_{\text{up}}$ **then**
        Calculate marginal gain: $\Delta V_i \leftarrow V(B_i + 1, p_i) - V(B_i, p_i)$
        Push$(\mathcal{H}, (\Delta V_i, i))$
    end if
end for
while $R > 0$ **and** $\mathcal{H}$ is not empty **do**
    $(\Delta V_{i^*}, i^*) \leftarrow$ Pop$(\mathcal{H})$ ▷ Select task with max marginal gain
    $B_{i^*} \leftarrow B_{i^*} + 1$
    if $B_{i^*} < B_{\text{up}}$ **then**
        Update marginal gain: $\Delta V_{\text{new}} \leftarrow V(B_{i^*} + 1, p_{i^*}) - V(B_{i^*}, p_{i^*})$
        Push$(\mathcal{H}, (\Delta V_{\text{new}}, i^*))$
    end if
    $R \leftarrow R - 1$
end while
return $\mathbf{B}$

E Experiment Details

We implement our training pipeline using the **Verl** (Sheng et al., 2024) framework, utilizing SGLang (Zheng et al., 2024) as the inference engine. For optimization, we employ the AdamW optimizer with a learning rate of 1×10^{-6}. The hyperparameters are adjusted based on model scale: for models smaller than 7B, we set the global batch size $M = 512$ and train for approximately 500 steps; for the 7B model, we set $M = 256$ and extend training to nearly 1000 steps. To accommodate complex reasoning chains, we set the maximum response length to 4096 tokens. Consistent with recent trends in reasoning alignment, we do not incorporate the KL divergence penalty (i.e., $\beta_{\text{KL}} = 0$). Regarding the specific parameters for our **CoBA-RL** method, we constrain the rollout budget for each instance within the range $[B_{\text{low}}, B_{\text{up}}] = [2, 128]$. Detailed sensitivity analyses of key CoBA-RL hyperparameters are provided in Appendix F.6 and Appendix F.7. The inference hyperparameters used for evaluation are listed in Table 7. All experiments are conducted on 8 NVIDIA H800 GPUs.

We allocate budgets using each prompt's most recent empirical pass rate and update the cache after reward computation for its next occurrence. Cold start uses uniform allocation, with no extra probe rollouts. Rates are clipped to $[0.01, 0.99]$ only for value computation.

Table 7: Hyperparameters for evaluation rollout.

Hyperparameter	**Value**
Temperature	1.0
Top-p	0.9
Sample Count (n)	16
Do Sample	True

Regarding the specific implementation of budget allocation methods, following the recommendations of DAPO (Yu et al., 2025), we adopt the *Clip-higher* strategy for both Knapsack-RL and CoBA-RL.

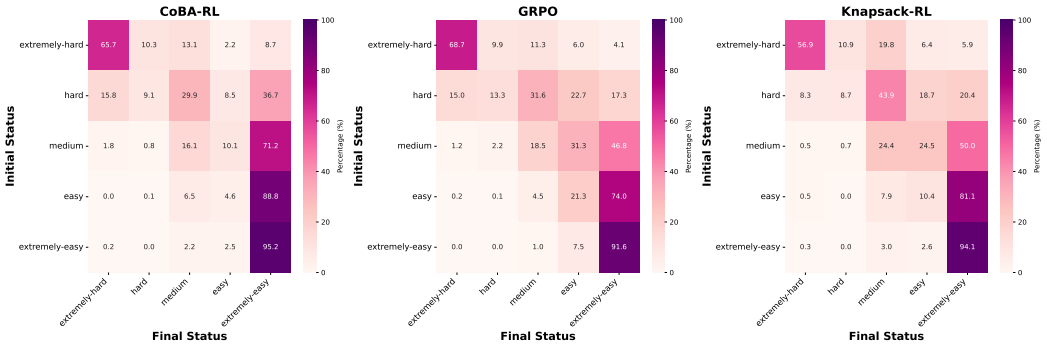

Figure 6: Task transition matrices for Qwen2.5-7B-Instruct during training. The cell (i, j) indicates the percentage of samples transitioning from the initial status i to the final status j.

F Additional Results

F.1 Analysis of Task Difficulty Transition

To evaluate performance across different learning stages, we categorize training prompts into five difficulty levels based on initial success rate (p_i): *extremely-hard* ($p_i = 0$), *hard* ($0 < p_i \leq 0.2$), *medium* ($0.2 < p_i < 0.8$), *easy* ($0.8 \leq p_i < 1.0$), and *extremely-easy* ($p_i = 1.0$). As shown in the final column of Figure 6, CoBA-RL consistently achieves the highest conversion rates. For *medium* tasks, it reaches 71.2%, significantly surpassing GRPO (46.8%) and Knapsack-RL (50.0%). For *hard* tasks, it achieves 36.7%, nearly doubling the GRPO baseline (17.3%) and outperforming Knapsack-RL (20.4%). Furthermore, it leads in converting *extremely-hard* tasks (8.7% vs. 4.1%) and *easy* tasks (88.8% vs. 74.0%), while maintaining the highest retention for *extremely-easy* instances (95.2%).

These results underscore the distinct superiority of CoBA-RL: By dynamically aligning resources with model capability, CoBA-RL maximizes learning potential across the entire difficulty spectrum, from solving novel problems to retaining established knowledge.

F.2 Effect of Total Rollout Budget

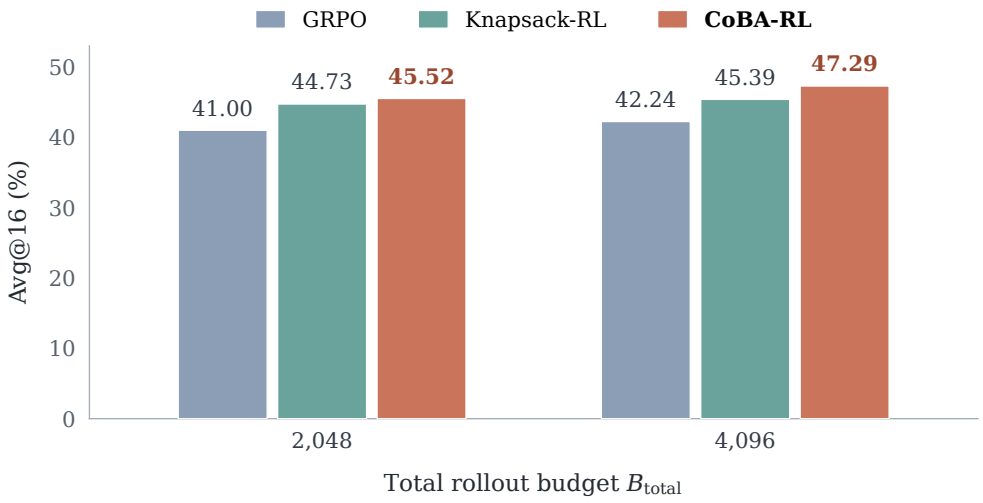

Figure 7: Performance comparison on Qwen2.5-7B-Instruct under different total rollout budgets.

We evaluate the performance of Qwen2.5-7B-Instruct under varying total rollout budgets B_{total}, as illustrated in Figure 7. CoBA-RL outperforms both GRPO and Knapsack-RL at both evaluated budget levels. In particular, CoBA-RL demonstrates strong rollout-budget efficiency: with $B_{\text{total}} = 2048$, it achieves an Avg@16 of 45.52%, exceeding both baselines at the same budget and surpassing GRPO's 42.24% with twice the rollout budget ($B_{\text{total}} = 4096$).

We attribute this improvement to the resource inefficiency of GRPO's uniform allocation strategy, which indiscriminately distributes the rollout budget across all samples and may waste computation on instances with limited training potential. In contrast, CoBA-RL uses the capability-oriented value function to guide rollout allocation toward high-value instances, thereby maximizing the aggregate training value of the current batch.

F.3 Generalization Beyond Math

To evaluate whether CoBA-RL generalizes beyond mathematical reasoning, we conduct additional experiments on Qwen2.5-7B-Instruct in two settings: code generation and general/logic reasoning. For code generation, we train on APPS and evaluate on the APPS test set, MBPP, and HumanEval. For general and logic reasoning, we evaluate math-trained checkpoints on GPQA Diamond, MMLU-Redux, and ZebraLogic.

Table 8: Generalization results beyond math reasoning on Qwen2.5-7B-Instruct. CoBA-RL consistently outperforms GRPO and Knapsack-RL across both code generation and general/logic reasoning domains.

Domain	**GRPO**	**Knapsack-RL**	**CoBA-RL**
Code Generation (Avg)	67.78	67.66	**69.29**
General & Logic Reasoning (Avg)	40.60	42.73	**43.24**

The results show that CoBA-RL is not limited to math reasoning tasks. Its core principle, namely that the training value of a sample depends on the model's evolving capability, is domain-agnostic. By dynamically calibrating rollout allocation according to the current model capability, CoBA-RL improves performance across both code generation and general reasoning evaluations.

F.4 End-to-End Generation Efficiency

We profile end-to-end rollout generation on Qwen2.5-7B-Instruct using 8 NVIDIA H800 GPUs. Table 9 reports the generation time, generated tokens, and mean SM utilization of each method. All methods request the same total budget of 4,096 trajectories per step. Because response lengths differ across methods, generated tokens and wall-clock generation time are measured outcomes rather than matching constraints.

Table 9: End-to-end generation efficiency on Qwen2.5-7B-Instruct with a matched budget of 4,096 trajectories per step.

Method	**Avg@16**	**Gen. time/step**	**Tokens/step**	**Mean SM util.**
GRPO, fixed 16	42.24	80.11 s	4.77M	51.26%
Knapsack-RL	45.39	116.76 s	7.34M	75.61%
CoBA-RL, $[B_{\text{low}}, B_{\text{up}}] = [2, 48]$	46.94	106.44 s	6.95M	74.45%
CoBA-RL, $[B_{\text{low}}, B_{\text{up}}] = [2, 128]$	**47.29**	110.60 s	7.22M	74.75%

Uniform GRPO produces fewer output tokens and has a shorter generation time, together with lower mean SM utilization. Under the measured configurations, CoBA-RL achieves higher Avg@16 with shorter generation time and similar mean SM utilization compared with Knapsack-RL. Tightening the upper budget bound from 128 to 48 reduces generation time by 3.76% while decreasing Avg@16 by 0.35 points, illustrating an accuracy–generation-time trade-off.

F.5 Robustness to Pass-Rate Estimation Noise

To evaluate robustness to pass-rate estimation noise, we perturb the pass-rate estimates used during budget allocation by adding Gaussian noise with standard deviation $\sigma = 0.1$. This experiment tests whether CoBA-RL depends on precise pass-rate estimates or can remain effective under noisy allocation signals.

Table 10: Evaluation under noisy pass-rate estimates.

Method	Avg
GRPO	42.24
Knapsack-RL	45.39
CoBA-RL + noise ($\sigma = 0.1$)	46.52
CoBA-RL	**47.29**

As shown in Table 10, adding Gaussian noise to the pass-rate estimates only causes a moderate performance drop, from 47.29 to 46.52. This result suggests that CoBA-RL is robust to noisy pass-rate estimates and mainly depends on the relative ordering of marginal gains among samples, rather than highly precise pass-rate values.

F.6 Effect of Beta Parameter κ

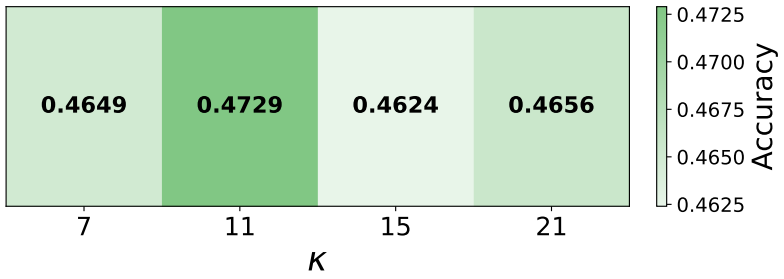

Figure 8: Ablation study on Qwen2.5-7B-Instruct illustrating the impact of different Beta parameter values $\kappa \in \{7, 11, 15, 21\}$ on model performance.

In this section, we examine the impact of the Beta parameter κ on Qwen2.5-7B-Instruct. The parameter κ is defined as the constant sum of the Beta distribution shape parameters, i.e., $\alpha + \beta$, and controls the overall shape of the Beta distribution. Therefore, it influences the sharpness of the capability-oriented value function.

We evaluate four distinct values: $\kappa \in \{7, 11, 15, 21\}$. As illustrated in Figure 8, our method exhibits strong robustness to variations in κ, with performance fluctuations remaining minimal across different settings. Specifically, the accuracy ranges from 46.24% at $\kappa = 15$ to a peak of 47.29% at $\kappa = 11$.

This stability validates the reliability of our capability-driven allocation criterion, suggesting that the improvement stems from the dynamic allocation mechanism itself rather than specific hyperparameter tuning. Based on these results, we adopt $\kappa = 11$ as the default configuration for our main experiments.

F.7 Sensitivity to τ

We further provide a sensitivity analysis on Qwen2.5-7B-Instruct for the temperature coefficient τ. In our default setting, CoBA-RL dynamically computes τ based on the current batch's pass-rate variance.

Intuitively, τ controls the budget gap between high-value and low-value samples. When the batch contains samples with diverse difficulty levels, a larger τ allows the allocator to create more meaningful budget gaps; when samples have similar difficulty, a smaller τ keeps the allocation closer to uniform.

Table 11: Sensitivity analysis of dynamic τ. CoBA-RL denotes the default setting with dynamically computed τ.

	CoBA-RL	fix $\tau = 4$	fix $\tau = 10$
Avg	**47.29**	46.15	46.12

As shown in Table 11, dynamic τ consistently outperforms fixed τ values, confirming the benefit of adapting allocation sharpness according to the current batch composition. Overall, these results indicate that the effectiveness of CoBA-RL does not rely on fragile hyperparameter tuning.

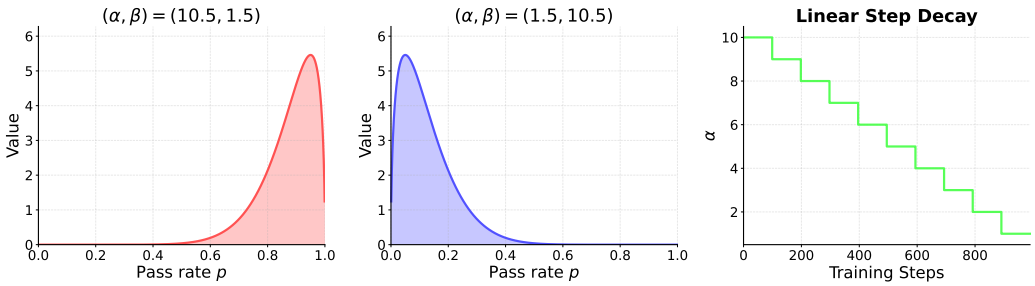

Figure 9: Visualization of the baseline strategies. **Left & Middle:** The probability density functions of the static Beta distributions used for exploitation ($(\alpha, \beta) = (10.5, 1.5)$) and exploration ($(\alpha, \beta) = (1.5, 10.5)$). **Right:** The pre-defined annealing schedule for α in the Linear Step Decay baseline.

F.8 Sensitivity to Nonlinear Mapping Forms

We analyze whether CoBA-RL depends on the specific form of the nonlinear mapping $\Psi(\cdot)$. Since RL training gradually converges, the global failure rate changes more slowly in later stages; a linear mapping may therefore become insensitive to small capability changes. We compare a linear mapping with two nonlinear alternatives, sigmoid and log-like mappings, as shown in Table 12.

Table 12: Sensitivity analysis of different mapping functions $\Psi(\cdot)$.

Mapping Function	**Avg**
Linear	46.12
Sigmoid	47.29
Log-like	46.98

Both nonlinear mappings outperform the linear mapping, indicating that the improvement mainly comes from nonlinear amplification rather than a specific parametric form of $\Psi(\cdot)$.

F.9 Multi-seed Evaluation

We conduct multi-seed evaluations with Qwen2.5-7B-Instruct to examine the stability of CoBA-RL. For AMC23, we report the mean and standard deviation over five random seeds. We additionally evaluate the same checkpoints on OlympiadBench using five evaluation seeds. The combined results are reported in Table 13.

CoBA-RL achieves the highest mean on both benchmarks and has the lowest variance on AMC23. On OlympiadBench, CoBA-RL outperforms Knapsack-RL by 1.33 points while maintaining low variation across evaluation seeds.

Table 13: Multi-seed evaluation on AMC23 and OlympiadBench with Qwen2.5-7B-Instruct. Results are reported as mean ± standard deviation over five seeds.

Method	AMC23	OlympiadBench
GRPO	70.31 ± 1.49	37.65 ± 0.28
Knapsack-RL	72.59 ± 1.41	41.65 ± 0.10
CoBA-RL	$\mathbf{75.19 \pm 0.79}$	$\mathbf{42.98 \pm 0.17}$

F.10 Allocation Dynamics

We further analyze the allocation dynamics of CoBA-RL to examine whether it assigns more rollouts to informative samples. We collect allocation statistics at representative training steps and report the averaged results in Table 14. We consider two metrics: learnable ratio and allocation entropy.

Table 14: Allocation dynamics across training, averaged over representative training steps.

Method	Learnable Ratio	Allocation Entropy
GRPO	44.26%	8.00
Knapsack-RL	67.69%	7.56
CoBA-RL	**72.87%**	**7.03**

The learnable ratio denotes the proportion of rollouts allocated to samples with intermediate pass rates, i.e., $0 < p_i < 1$. CoBA-RL achieves the highest learnable ratio, indicating that it allocates more budget to samples with effective training signals.

Allocation entropy measures how evenly the rollout budget is distributed. Higher entropy indicates more uniform allocation, while lower entropy reflects more targeted assignment. GRPO obtains an entropy of 8.00 due to uniform allocation. CoBA-RL achieves lower entropy than Knapsack-RL, suggesting more selective allocation toward high-value samples. These results show that CoBA-RL induces more effective and adaptive allocation behavior during training.

F.11 Visualization of Static and Linear Decay Baselines

To facilitate a deeper understanding of the comparative analysis in Section 3.4, we provide visual illustrations of the baseline strategies in Figure 9.

Figure 9 (Left and Middle) depicts the geometric shapes of the fixed value functions used in the static strategies. The configuration $(\alpha, \beta) = (10.5, 1.5)$ results in a distribution heavily skewed toward high success rates p, thereby enforcing an exploitation-prioritized allocation. Conversely, the setting $(\alpha, \beta) = (1.5, 10.5)$ yields a distribution peaked at low success rates, promoting an exploration-prioritized allocation.

Figure 9 (Right) illustrates the Linear Step Decay strategy, where α_t is annealed stepwise from 10 to 1 over the course of training. This baseline follows a rigid, pre-defined schedule and therefore lacks the adaptability of our capability-oriented approach.

G Limitation

Although CoBA-RL achieves consistent improvements across different model scales and domains, there remain several limitations. First, our experiments are still limited to models up to 7B due to computational constraints. Future work could evaluate whether the same allocation dynamics hold for larger-scale models. Second, our current value estimation relies on predefined signals such as pass rates and global capability statistics. A promising future direction is to train a dedicated value predictor to estimate sample utility from richer online training signals.