BEYOND PROMPT SELECTION: LEARNING TO ALLOCATE ROLLOUTS FOR EFFICIENT GRPO TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

GRPO, a widely used RL method for reasoning models, allocates the same number of rollouts to every prompt, despite differences in the value of additional rollouts. This is particularly wasteful for *degenerated* prompts, for which sampled responses receive identical rewards and therefore provide no policy gradient. We introduce ADAROLL, an adaptive rollout allocation framework that dynamically concentrates computation on prompts where additional rollouts are most likely to be useful. In the first training epoch, ADAROLL performs *pilot rollout initialization*, using one rollout per prompt and its entropy and reward to predict prompt degeneracy and the distribution of future correct and incorrect rollouts. It then progressively allocates additional rollouts over multiple rounds according to a cost-benefit tradeoff between their predicted learning value and generation cost. In subsequent epochs, the predictor continuously adapts to the evolving policy using historical and newly generated rollout information, without requiring additional pilot rollouts. Across model scales and mathematical reasoning tasks, ADAROLL's predictions are 31% more accurate than the baselines, saving up to 47% of computation and improving the LLM's final performance by up to 2.5%.

1 INTRODUCTION

Reinforcement learning (RL) has become a powerful paradigm for improving the reasoning capabilities of large language models (LLMs), particularly on complex tasks such as mathematical problem solving, with minimal supervision. Among recent RL algorithms, Group Relative Policy Optimization (GRPO) (Shao et al., 2024a) has gained widespread adoption for its simplicity and effectiveness. Given a prompt, GRPO samples a group of responses from the current policy, evaluates their rewards, and uses relative rewards to construct the policy advantage, eliminating the need for a separate value model. However, GRPO allocates the same number of rollouts to every prompt, despite substantial differences in how informative additional rollouts are. Since reasoning rollouts can be long and expensive to generate, this uniform allocation can result in considerable wasted computation.

A key observation is that the learning value of a prompt can vary dramatically. If all sampled responses to a prompt receive the same reward—for example, if they are all correct or all incorrect—their relative advantages are zero, and the prompt produces no policy gradient. We refer to such prompts as *degenerate*. For these prompts, generating additional rollouts provides no learning signal, yet GRPO continues to spend the same rollout budget on them. This suggests a natural question: *can we dynamically allocate rollout computation to the prompts for which additional rollouts are most likely to be informative?*

Recent approaches have begun to address this inefficiency by identifying non-degenerate prompts and allocating more computation to them (Qu et al., 2026; Baroian & Berger, 2026; Zheng et al., 2025) However, predicting whether a prompt will be informative is challenging because its rollout behavior depends not only on the prompt itself, but also on the current policy. Methods based solely on prompt-level properties may miss this interaction, while methods relying on historical rollout outcomes can become stale as the policy evolves. More fundamentally, a prompt's value is not binary: even among non-degenerate prompts, the expected benefit of additional rollouts can vary substantially. Effective rollout allocation therefore requires both adapting to the evolving policy and estimating how much additional computation is likely to be worthwhile.

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.

We introduce **ADAROLL**, an **ada**ptive **roll**out allocation framework that treats rollout generation as a sequential cost-benefit problem. Rather than assigning a fixed rollout budget to every prompt, it invests computation where the predicted learning value is highest. In the first training epoch, when no rollout history is available, ADAROLL performs *pilot rollout initialization*, generating one rollout per prompt and using its entropy and reward to initialize a lightweight predictive model. The model estimates each prompt's degeneracy probability and a Beta distribution over future correct and incorrect rollouts. Based on these predictions, ADAROLL selects promising prompts and progressively allocates additional rollouts over multiple rounds, stopping when the predicted benefit no longer justifies the cost.

In subsequent epochs, ADAROLL continuously updates its predictions using information from both historical and newly generated rollouts, eliminating the need for additional pilot rollouts. This allows the allocation strategy to track the evolving policy and progressively improve its estimates of which prompts are likely to benefit from further sampling. Experiments across different model scales and mathematical reasoning tasks demonstrate that ADAROLL's predictions are 31% more accurate than the baselines, saving up to 47% of computation and improving the LLM's final performance by up to 2.5%, substantially improving rollout efficiency while maintaining or improving training performance over existing approaches.

This perspective reframes GRPO efficiency as a problem of adaptive rollout computation, where both the choice of prompts and the number of rollouts are dynamically determined by their predicted training value. Across multiple model scales and mathematical reasoning benchmarks, ADAROLL substantially reduces rollout computation while maintaining comparable or improving performance over standard GRPO and existing rollout-selection methods.

2 RELATED WORKS

Several recent works aim to improve RL training efficiency by identifying prompts whose rollouts are likely to provide informative learning signals. Qu et al. (2026) predicts the probability of non-degenerate rollouts using Bayesian learning from historical rollout outcomes, while Baroian & Berger (2026); Zheng et al. (2025) similarly leverage rollout history, with Zheng et al. (2025) further distinguishing between overly easy and overly difficult problems through dynamically learned thresholds. Rather than explicitly predicting non-degeneracy, Kim et al. (2026); Jiang et al. (2026) allocate more computation to prompts expected to yield higher reward variance. These approaches, however, primarily use rollout rewards as the selection signal, overlooking other readily available signals such as rollout entropy that can provide additional information about prompt-level uncertainty. More importantly, they generally use a predefined number of rollouts per selected prompt, rather than jointly deciding *which prompts to sample and how much computation to allocate* based on the predicted value of additional rollouts.

Other approaches incorporate richer signals for prompt selection, including prompt similarity obtained through additional attention mechanisms (Sun et al., 2026), evaluations from external LLMs (Wu et al., 2026; Liang et al., 2026), or direct prediction of prompt-level mean rewards using an LLM (Gao et al., 2025). While these methods exploit information beyond rollout rewards, they often rely on auxiliary models or prompt-level representations that are decoupled from the evolving RL policy. In addition, they primarily rely on information collected from previous iterations and do not incorporate rollout signals from the current RL iteration as they become available. In contrast, ADAROLL combines entropy and reward signals from both historical and newly generated rollouts to continuously track the evolving policy, while jointly determining which prompts to sample and how many additional rollouts to generate based on their predicted learning value.

3 COST-EFFICIENT ROLLOUT GENERATION FOR GRPO

GRPO learns from the relative rewards of multiple sampled rollouts, making the usefulness of a prompt inherently dependent on the diversity of outcomes within its rollout group. For prompt p_i, GRPO samples G rollouts $R_i^{(1)}, \ldots, R_i^{(G)}$, each receiving a binary reward $r_i^j \in \{0, 1\}$. The GRPO

objective for p_i is:

$$\mathcal{L}_{\mathrm{GRPO}}(p_i) = -\frac{1}{G}\sum_{j=1}^{G}\left[\min\left(\frac{\pi_\theta}{\pi_{\mathrm{old}}}A_i^{(j)}, \mathrm{clip}\left(\frac{\pi_\theta}{\pi_{\mathrm{old}}}, 1-\epsilon_{\mathrm{clip}}, 1+\epsilon_{\mathrm{clip}}\right)A_i^{(j)}\right) - \beta D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{ref}})\right]. \tag{1}$$

where the relative advantage is:

$$A_i^{(j)} = \frac{r_i^{(j)} - \bar{r}_i}{\sigma_i + \epsilon_{\mathrm{adv}}}, \qquad \bar{r}_i = \frac{1}{G}\sum_{j=1}^{G} r_i^{(j)}, \qquad \sigma_i = \mathrm{std}\left(r_i^{(1)}, \ldots, r_i^{(G)}\right). \tag{2}$$

Here, ϵ_{clip} denotes the clipping range, $\epsilon_{\mathrm{adv}} > 0$ stabilizes advantage normalization, and β weights KL regularization.

Making GRPO efficient. When all G rollouts for a prompt receive the same reward, every $A_i^{(j)} = 0$, and the prompt provides no reward-dependent policy-gradient signal despite the cost of generating and evaluating its rollouts. We call such prompts *degenerate*. This motivates a central question: **can we identify informative prompts early and allocate additional rollouts only where they are likely to be useful?**

Our key insight is to use each generated rollout as evidence for this decision, progressively refining prompt-level predictions and allocating computation accordingly. This leads to three principles underlying our ADAROLL method:

- **Refine predictions with every rollout.** In Section 3.1, we show that an individual rollout's entropy and correctness predict whether its prompt will ultimately be non-degenerate, motivating progressive rather than one-shot prompt selection.
- **Allocate computation by marginal cost-benefit.** In Section 3.2, we show how ADAROLL prioritizes prompts with the highest predicted benefit from an additional rollout relative to its cost.
- **Progressively generate rollouts under uncertainty.** In Section 3.3, ADAROLL generates more rollouts when predictions are confident and fewer when they are uncertain, gathering information before committing additional computation while still reaching the desired group size.

We next describe these design choices in detail.

3.1 PREDICTING NON-DEGENERACY FROM ROLLOUT OBSERVATIONS

Given n_i observed rollouts for prompt p_i, we estimate whether the remaining $G - n_i$ rollouts will make the prompt non-degenerate. We model the probability that p_i produces a correct response with a Beta distribution parameterized by α_i and β_i, interpreted as pseudo-counts of correct and incorrect outcomes. The resulting Beta-Binomial probability of observing k_1 additional correct and k_0 additional incorrect rollouts is:

$$P(k_1, k_0 \mid \alpha_i, \beta_i) = \binom{k_0 + k_1}{k_0} \prod_{j=0}^{k_1-1}(\alpha_i + j) \prod_{j=0}^{k_0-1}(\beta_i + j) \Big/ \prod_{j=0}^{k_0+k_1-1}(\alpha_i + \beta_i + j). \tag{3}$$

The probability that the full group is non-degenerate is therefore:

$$\begin{aligned} t_i(\alpha_i, \beta_i, n_i) &= P\left(\text{non-degeneracy} \mid \alpha_i, \beta_i, R_i^{(1)}, \ldots, R_i^{(n_i)}\right) \\ &= 1 - \underbrace{\mathbf{1}\left\{\sum_{j=1}^{n_i} r_i^{(j)} = 0\right\}}_{\text{observed all zeros}} P(0, G-n_i \mid \alpha_i, \beta_i) - \underbrace{\mathbf{1}\left\{\sum_{j=1}^{n_i} r_i^{(j)} = n_i\right\}}_{\text{observed all ones}} P(G-n_i, 0 \mid \alpha_i, \beta_i). \end{aligned} \tag{4}$$

If the observed rollouts contain both correct and incorrect responses, the group is already non-degenerate; otherwise, t_i is the probability that the remaining rollouts contain the opposite outcome.

We parameterize the Beta distribution equivalently by its mean and concentration,

$$\mu_i = \frac{\alpha_i}{\alpha_i + \beta_i}, \qquad \nu_i = \alpha_i + \beta_i, \tag{5}$$

where μ_i is the predicted accuracy and ν_i controls predictive uncertainty. Next, we learn a lightweight predictor of (μ_i, ν_i) using features available during rollout generation: *reward* provides information about expected correctness, while *entropy* provides complementary information about uncertainty.

Entropy and reward from a single rollout predicts problem non-degeneracy. At the beginning of the first RL iteration, no historical rollout information is available. We therefore generate one pilot rollout per prompt to initialize the predictor; in subsequent iterations, historical observations provide this initial information. A rollout's binary reward provides a direct signal about the prompt's expected accuracy μ_i, while its token-level entropy captures uncertainty in the model's generation. Specifically, for a response $R_i^{(j)}$ of length L, let $h_{i,j,t}$ denote the entropy of the policy's distribution over the full vocabulary $\mathcal{V}$ at token position t:

$$h_{i,j,t} = -\sum_{v \in \mathcal{V}} \pi_\theta(v \mid p_i, R_{i,<t}^{(j)}) \log \pi_\theta(v \mid p_i, R_{i,<t}^{(j)}). \quad (6)$$

We sort the token entropies in descending order, $h_{i,j,(1)} \geq \cdots \geq h_{i,j,(L)}$, and define the rollout entropy feature as the sum of the top k values:

$$H_i^{(j)} = \sum_{\ell=1}^{k} h_{i,j,(\ell)}, \quad (7)$$

where $k = L/2$. We find this choice effective empirically, while focusing on high-entropy positions is also consistent with prior findings that they often correspond to reasoning forks and carry strong policy-gradient signals (Wang et al., 2025; Li et al., 2026).

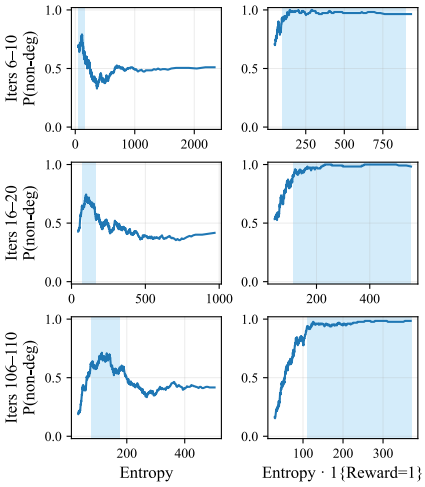


Figure 1: Entropy and reward from a single rollout identify prompts with high non-degeneracy probability (shaded). The relationship is nonlinear and evolves during training.

Figure 1 shows the relationship between these single-rollout features and prompt non-degeneracy across different training checkpoints. We smooth the binary non-degeneracy outcomes to visualize the empirical probability of non-degeneracy as a function of each feature. Even a single rollout provides substantial predictive information: the first column shows clear ranges of entropy associated with higher non-degeneracy probability, indicating that entropy alone can identify promising prompts. Moreover, conditioning on correctly answered rollouts (right column), high entropy is particularly strongly associated with non-degeneracy. These results motivate using both entropy and reward as predictor features and, importantly, modeling their nonlinear interaction.

Additional rollouts improve prediction accuracy. While a single rollout provides substantial information, additional observations can make predictions more accurate and stable. In particular, observing both correct and incorrect rewards immediately guarantees non-degeneracy. Historical rollout features also provide useful context: for example, prompts that are consistently answered correctly are increasingly likely to remain degenerate as the policy improves.

To capture these signals and their nonlinear interactions, we use a two-layer, two-head MLP ϕ to predict the Beta parameters μ_i and ν_i (Figure 2). The first layer separately mean-pools features from current and historical rollouts (up to four previous epochs). Since mean pooling loses the number of observations, we additionally provide the rollout count. The resulting 4-dimensional representations are concatenated and passed through the second layer, whose two heads predict accuracy μ_i and concentration ν_i. More observations should increase prediction confidence, reflected by a higher ν_i.

Because the relationship between rollout features and non-degeneracy can evolve rapidly during RL training, as illustrated across the rows of Figure 1, we refit the predictor $\phi(H, r, n)$ after each RL iteration using data from the preceding five iterations. Further details are provided in the appendix.

3.2 Cost-Effective Prompt Selection

Next, we use our predictor to decide *which* prompts should receive additional rollouts. We generate rollouts progressively over multiple rounds, selecting prompts for which completing the rollout

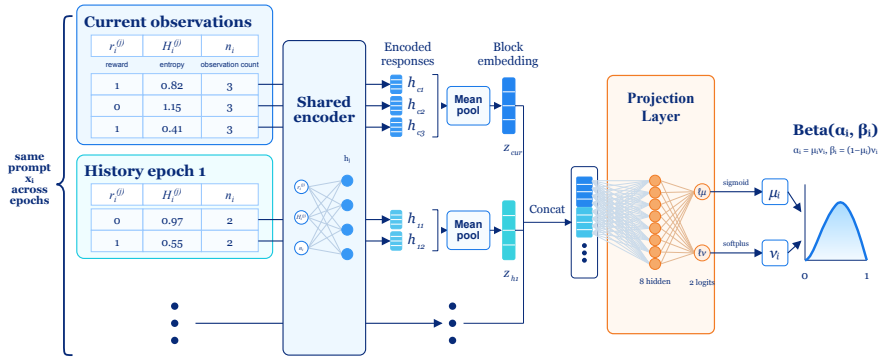


Figure 2: The predictor is a two-layered MLP that reads in input features including reward, entropy and rollouts count and generates the parameters for the Beta distribution. The input features goes through a shared encoder. Encoded features from the same checkpoint's rollout are mean-pooled, while those from past checkpoints are concatenated. The features are then projected onto the outputs.

Algorithm 1 Progressive Rollout Generation

Require: Prompt pool $\mathcal{P}$, target useful groups B, group size G.

1: Initialize $W_{\text{past}} \leftarrow 0$ (rollouts generated) and $H_{\text{past}} \leftarrow 0$ (non-degenerate groups obtained).
2: **while** $H_{\text{past}} < B$ **do**
3: Run Algorithm 2 to obtain the prompts worth generating, $\mathcal{S}$.
4: Run Algorithm 3 to calculate the number of rollouts d_i to generate for each $p_i \in \mathcal{S}$
5: Generate additional d_i rollouts for prompt $x_i \in \mathcal{S}$ and score the rewards.
6: $W_{\text{past}} \leftarrow W_{\text{past}} + \sum_{i\in\mathcal{S}} d_i$.
7: $H_{\text{past}} \leftarrow H_{\text{past}} +$ number of newly harvested non-degenerated groups.
8: **end while**
9: Assemble B completed non-degenerated groups; perform GRPO update; update $\phi(H, r, n)$.

group offers the most favorable cost-benefit tradeoff. We formalize this criterion below and show that the optimal selection can be obtained by sorting prompts according to their marginal cost.

Suppose that, at the current round, W_{past} rollouts have been generated and H_{past} non-degenerate groups have been obtained in previous rounds. For a candidate set of prompts $\mathcal{S}$, we define its amortized cost-benefit tradeoff as the total number of rollouts required per expected non-degenerate group:

$$J(\mathcal{S}) = \frac{W_{\text{past}} + \sum_{i\in\mathcal{S}}(G - n_i)}{H_{\text{past}} + \sum_{i\in\mathcal{S}} t_i} \tag{8}$$

Our goal is to select $\mathcal{S}$ minimizing $J(\mathcal{S})$. For an individual prompt, the corresponding marginal cost is:

$$c_i = \frac{(G - n_i)}{t_i} \tag{9}$$

which measures the number of additional rollouts required per expected non-degenerate group if its rollout group is completed.

The optimal selection can be obtained by sorting prompts according to their marginal costs and considering prefixes of the sorted list. Intuitively, a prompt should be included when its marginal cost is no greater than the resulting amortized cost.

Theorem 1 (Optimal prompt selection by marginal cost)**.** *For any round of progressive rollout with $W_{\text{past}} > 0$, consider all incomplete prompts with $t_i > 0$ and $0 \le n_i < G$, and sort them such that $c_{(1)} \le \cdots \le c_{(M)}$. For each prefix $\mathcal{S}_K = \{p_{(1)}, \ldots, p_{(K)}\}$, let $J(K) = J(\mathcal{S}_K)$. Define $K^* = \max \arg\min_K J(K), \mathcal{S}^* = \mathcal{S}_{K^*}$. Then,*

$$\mathcal{S}^* \in \arg\min_{\mathcal{S}} J(\mathcal{S}).$$

That is, an optimal prompt set is given by a prefix of the prompts sorted by marginal cost.

Algorithm 2 Cost-based prompt selection

Require: Incomplete prompts $\mathcal{P}$ with features x_i, observed rewards, and counts n_i; group size G; $W_{\text{past}}, H_{\text{past}}$; predictor ϕ
Ensure: Selected prompt set $\mathcal{S}^*$
1: **for** each $i \in \mathcal{P}$ **do**
2: $(\mu_i, \nu_i) \leftarrow \phi(x_i)$; $t_i \leftarrow$ Equation (4); $c_i \leftarrow (G - n_i)/t_i$
3: **end for**
4: Sort prompts into $p_{(1)}, \ldots, p_{(M)}$ by c_i so that $c_{(1)} \leq \cdots \leq c_{(M)}$
5: For each $0 \leq K \leq M$, $\mathcal{S}_K \leftarrow \{p_{(1)}, \ldots, p_{(K)}\}$, and compute $J(K)$ from Equation (8)
6: **return** $\mathcal{S}_{K^*}$ where $K^* = \arg\min_K J(K)$

Algorithm 3 Calculating the number of rollouts to generate

Require: Admitted prompt i with n_i observed rollouts; group size G; current benchmark J^*
Ensure: Number r of additional rollouts to generate
1: $r \leftarrow 1$
2: **while** $r < G - n_i$ **do**
3: Calculate $c_i^{\text{look}}(r, 0), c_i^{\text{look}}(0, r)$ according to Equation (12)
4: **if** $\max\{c_i^{\text{look}}(r, 0), c_i^{\text{look}}(0, r)\} > J^*$ **then break**
5: $r \leftarrow r + 1$
6: **end while**
7: **return** r

Moreover, letting $J^* = \min_K J(K)$, the canonical optimal solution $\mathcal{S}^*$ satisfies

$$p_i \in \mathcal{S}^* \iff c_i \leq J^*. \tag{10}$$

Equivalently, a prompt is selected if and only if its marginal cost is no greater than the optimal overall amortized cost.

See proof on page 14. Algorithm 2 gives the procedure for selecting prompts for additional rollouts.

3.3 Progressive rollout generation with outcome lookahead

The previous section determines *which* prompts should receive additional rollouts; we now determine *how many* to generate at each round. While the final group size G remains fixed, rollouts are progressively generated in adaptive batches. Recall that the Beta concentration ν_i captures predictive confidence: when ν_i is large, additional observations are less likely to change the prediction, whereas low ν_i indicates that the prediction may change substantially with new evidence. Accordingly, we generate more rollouts at once for confident predictions and fewer for uncertain ones. We implement this through outcome lookahead: if a prompt would remain cost-effective under every possible outcome of the next r rollouts, we generate all r at once; otherwise, we generate fewer, observe their outcomes, and update the prediction before continuing.

Formally, for an admitted prompt p_i, suppose $r = k_0 + k_1$ additional rollouts are generated, yielding k_0 incorrect and k_1 correct responses. Let $n_i^{\text{look}} = n_i + r < G$ and

$$\alpha_i' = \alpha_i + k_1, \qquad \beta_i' = \beta_i + k_0, \qquad n_i' = n_i + k_0 + k_1.$$

The lookahead non-degeneracy probability is obtained from Equation (4) by replacing (α_i, β_i, n_i) with $(\alpha_i', \beta_i', n_i')$:

$$t_i^{\text{look}}(k_0, k_1) = t_i(\alpha_i', \beta_i', n_i'). \tag{11}$$

The corresponding lookahead marginal cost is:

$$c_i^{\text{look}}(k_0, k_1) = \frac{G - n_i^{\text{look}}}{t_i^{\text{look}}(k_0, k_1)}. \tag{12}$$

We can safely generate r rollouts if the prompt remains cost-effective under every possible outcome of those r rollouts, i.e.:

$$\forall k_0, k_1 \geq 0 \text{ with } k_0 + k_1 = r, \quad c_i^{\text{look}}(k_0, k_1) \leq J^*. \tag{13}$$

We generate the largest number of rollouts for which this condition holds. Once it fails, we observe the generated rollouts, update the prediction, and reassess whether to continue. Theorem 2 in Appendix shows that it suffices to check the two extreme outcomes:

$$c_i^{\text{look}}(k_0, k_1) \leq \max\{c_i^{\text{look}}(0, r), c_i^{\text{look}}(r, 0)\}.$$

avoiding enumeration of all possible (k_0, k_1). The resulting procedure is given in Algorithm 3.

4 EXPERIMENT

We evaluate ADAROLL against state-of-the-art baselines on the six mathematical reasoning benchmarks described below, comparing both rollout efficiency and unweighted average accuracy. We first briefly describe our experimental setting. Then, we compare ADAROLL with existing rollout allocation methods across model scales and benchmarks, measuring training performance and rollout cost. We then conduct ablations to isolate the contributions of response entropy, additional rollout observations, and our progressive allocation strategy, providing insight into the efficiency gains of ADAROLL.

4.1 EXPERIMENTAL SETUP

Model, Dataset, and Benchmarks. We train on the union of DAPO-Math-17K-Cleaned (Yu et al., 2025) and the 7,500-example training split of LightEval-MATH (Lightman et al., 2023), which we refer to as *Math+DAPO*. DAPO-Math-17K-Cleaned contains 17,917 competition-style mathematics problems, while LightEval-MATH covers algebra, counting and probability, geometry, number theory, and precalculus. The resulting 25,417 question-answer pairs provide broad coverage of mathematical reasoning. We use the standard mathematical-reasoning prompt format, instructing the policy to produce a step-by-step solution followed by a boxed final answer.

We evaluate every 50 updates on MATH500 (Hendrycks et al., 2021; Lightman et al., 2024), AMC (Art of Problem Solving, 2024b), AIME 2024 (Art of Problem Solving, 2024a), Gaokao (Zhang et al., 2026), MinervaMath (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024). We report the unweighted average accuracy across the six benchmarks and measure generation cost by the total number of rollouts requested by each policy.

We train with GRPO (Shao et al., 2024b) for 250 iteration, using group size $G = 8$ and batch size 256. Further information on training hyperparameters is in Section A.

Baselines. We compare ADAROLLwith existing online prompt-selection and rollout-allocation methods for RLVR, using the same hyperparameters across methods. **Dynamic Sampling (GRPO)** (Shao et al., 2024c) generates all $G = 8$ rollouts at once and discards degenerate groups, serving as the vanilla baseline. **GRESO** (Zheng et al., 2025) estimates prompt difficulty from historical rewards and selects prompts with the highest expected non-degeneracy rate. **VIGOR** (Jiang et al., 2026) first generates one screening rollout and then allocates additional rollouts based on group reward variance. Since VIGOR uses a different allocation strategy and does not explicitly filter degenerate groups, we report only its final performance for comparison.

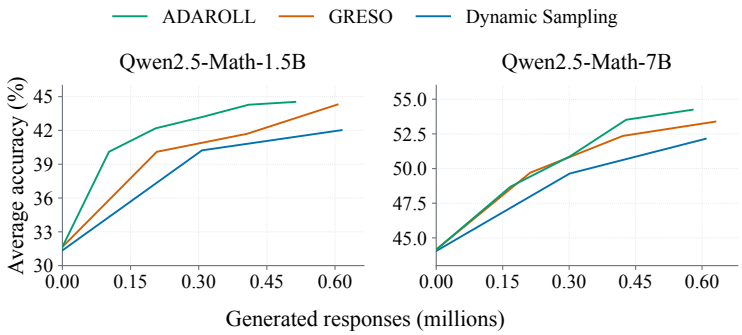


Figure 3: Average accuracy against generated rollouts. ADAROLL achieves a higher accuracy with fewer responses across both model sizes.

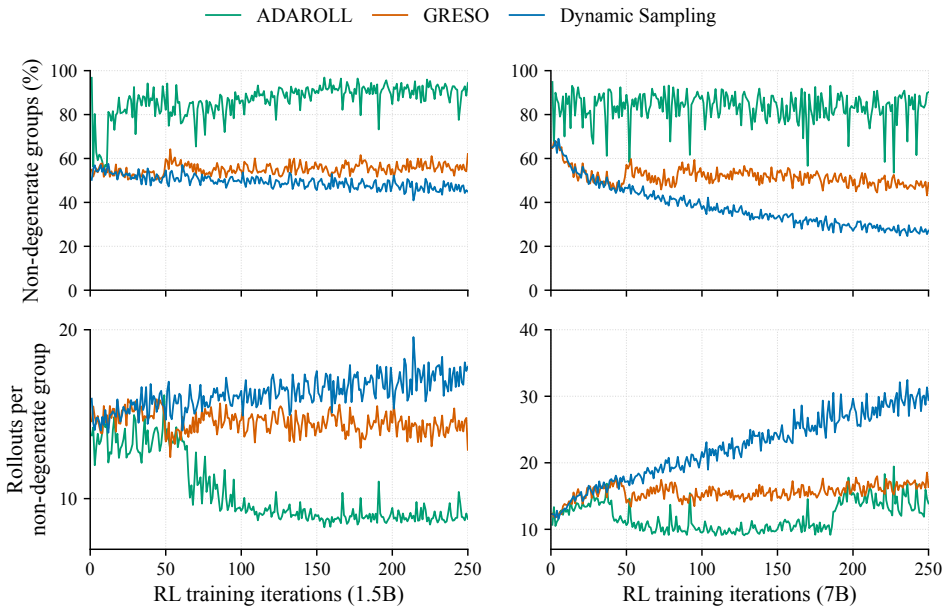


Figure 4: Percentage of the non-degenerate groups (top) and rollout efficiency (bottom) for the 1.5B (left) and 7B (right) models. The top row indicates that ADAROLL finds non-degenerated groups with an average accuracy of 86.5% and 83.4% on the 1.5B and 7B model respectively, outperforming GRESO's prediction accuracy by 31% on both models. The bottom row indicates that ADAROLL also just requires 10.37 and 11.86 rollouts on average to find a non-degenerate group, reducing costs by 28.6% and 23.8% relative to GRESO and by 36.1% and 47.0% relative to Dynamic Sampling, respectively. This shows that ADAROLL identifies non-degenerated prompts more accurately and efficiently.

Table 1: Math benchmark accuracy (%). Best performance is in bold and runner-up is underscored. ADAROLL outperforms comparable baselines on different model sizes by up to 2.5%, demonstrating its performance.

Method	MATH500	AIME24	AMC	Gaokao	MinervaMath	OlympiadBench	Avg.
Qwen2.5-Math-1.5B							
ADAROLL	<u>74.20</u>	15.00	<u>48.19</u>	**61.69**	**29.87**	**38.24**	<u>44.53</u>
VIGOR	**74.65**	**17.50**	47.59	<u>61.43</u>	<u>28.31</u>	<u>37.98</u>	**44.58**
GRESO	73.00	15.80	**51.80**	60.50	28.30	36.40	44.30
Dynamic Sampling	70.75	<u>15.83</u>	45.18	58.51	26.19	35.68	42.02
Qwen2.5-Math-7B							
ADAROLL	79.90	**35.83**	64.76	<u>67.27</u>	<u>34.19</u>	43.51	**54.24**
VIGOR	**80.95**	27.50	**68.37**	**68.38**	**35.11**	**44.96**	<u>54.21</u>
GRESO	79.45	<u>30.83</u>	<u>66.27</u>	66.10	34.01	<u>43.66</u>	53.39
Dynamic Sampling	<u>80.10</u>	27.50	62.95	65.00	34.01	43.32	52.15

4.2 RESULTS

ADAROLL outperform baselines and is more efficient. Figure 3 shows that ADAROLL consistently achieves higher accuracy with lower computation cost. On Qwen2.5-Math-1.5B, it achieves a 44.53% accuracy, exceeding GRESO by 0.23% and Dynamic Sampling by 2.51% while requiring at least 15% fewer rollouts to do so. ADAROLL scales to a larger, Qwen2.5-Math-7B, model, outperforming them by 0.85% and 2.09% with at least 5% fewer rollouts.

Table 1 shows that our method outperforms comparable baselines by up to 2.5% on the 1.5B model and by up to 2.1% on the 7B model, demonstrating ADAROLL's performance.

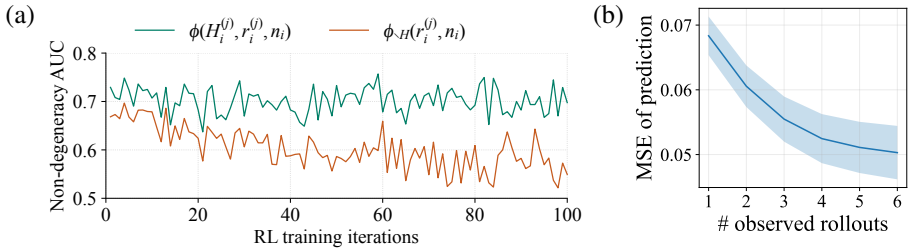


Figure 5: Ablations of non-degeneracy prediction. (a) Entropy of response improves the AUC of the non-degeneracy prediction by 16.5%. (b) Additional rollout observations reduce the MSE of non-degeneracy prediction by up to 26.5% after the predictor's quality is isolated from other effects. These ablation shows that response entropy and additional rollouts improves the prediction of non-degeneration.

Finally, Figure 4 highlights the improvements in rollout efficiency that AdaRoll provides. The top panels show that AdaRoll finds non-degenerated groups with an average accuracy of 86.5% and 83.4% on the 1.5B and 7B model respectively, outperforming GRESO's prediction accuracy by 31.3% and 31.7%. The bottom panels account for all generated rollouts, including those allocated to subsequently discarded prompts. AdaRoll requires an average of 10.37 and 11.86 rollouts per non-degenerate group at 1.5B and 7B, respectively–reductions of 28.6% and 23.8% relative to GRESO, and 36.1% and 47.0% relative to Dynamic Sampling. Together, these results show that AdaRoll are able to predict non-degeneracy rate with superior accuracy and concentrate generation on informative prompts while reducing the total cost of obtaining these training groups.

4.3 Ablation studies

Effect of response entropy on predicting non-degeneracy We conduct an ablation study to quantify the contribution of response entropy to our predictor ϕ. As shown in Fig. 5(a), we compare two otherwise identical Beta predictors: the full predictor, $\phi(H_i^{(j)}, r_i^{(j)}, n_i)$, and a variant without response entropy, $\phi_{\setminus H}(r_i^{(j)}, n_i)$. Removing the entropy feature reduces the AUC of 0.702 by 16.5% to 0.603. Moreover, the full predictor consistently achieves higher AUC at every evaluated step. This demonstrates that response entropy provides substantial predictive information beyond observed rewards for identifying non-degenerate prompts.

More rollouts lead to better predictions As discussed in Section 3.1, additional rollout observations improve the prediction in Equation (4). We quantify this benefit while isolating it from the fact that observing more rollouts itself increases the chance of encountering mixed rewards. Specifically, we fix $n_i = 2$ and $G = 8$ in the non-degeneracy prediction formula, while varying the pseudo-counts α_i and β_i predicted by the MLP after observing $0 < n \leq 6$ rollouts. Thus, changes in prediction accuracy arise solely from improved pseudo-count estimates. We evaluate the resulting non-degeneracy probability t_i using MSE against the ground truth, repeating the evaluation with 20 random samples and reporting the standard deviation as an error band. Results in Fig. 5(b) shows that, observing two rollouts improves prediction accuracy by 11.7% over using a single rollout, while additional observations yield improvements of up to 26.5%, demonstrating the effect of additional rollout observations.

5 Conclusion

We introduced AdaRoll, an adaptive rollout allocation framework for improving the efficiency of GRPO-based reinforcement learning. Rather than allocating a fixed rollout budget uniformly across prompts, AdaRoll predicts the value of additional rollouts using both response entropy and reward, continuously refining these predictions as new rollout information becomes available. It then progressively allocates computation according to a cost-benefit criterion, using outcome lookahead to determine how many rollouts can be generated before reassessing. Across model scales and mathematical reasoning benchmarks, AdaRoll reduces rollout cost while maintaining or improving training performance over existing allocation strategies.

AI USE STATEMENT

In this work, we used generative AI tools to assist in the writing of proofs (double checking the correctness), provide feedback on research experiments (diagnosing bugs), implement methods (the implementation have been carefully verified), and interpret results (visualizing key experiment results). We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, design or provide feedback on research methodology or experiments, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, and the rest of the required disclosure tasks are not applicable to this work. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Art of Problem Solving. Aime problems and solutions. https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions, 2024a. Accessed: 2026-09-22.

Art of Problem Solving. Amc problems and solutions. https://artofproblemsolving.com/wiki/index.php?title=AMC_Problems_and_Solutions, 2024b. Accessed: 2026-09-22.

Andrei Baroian and Rutger Berger. Prompt replay: speeding up grpo with on-policy reuse of high-signal prompts, 2026. URL https://arxiv.org/abs/2603.21177.

Zhaolin Gao, Joongwon Kim, Wen Sun, Thorsten Joachims, Sid Wang, Richard Yuanzhe Pang, and Liang Tan. Prompt curriculum learning for efficient llm post-training, 2025. URL https://arxiv.org/abs/2510.01135.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen 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 Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Heyang Jiang, Henry Liu, and Baharan Mirzasoleiman. Learning as reasoning unfolds: Progressive rollout allocation for efficient reinforcement learning. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=qoJJK90DGh.

Woojeong Kim, Ziyi Yang, Jing Nathan Yan, and Jialu Liu. Spend your rollouts where it counts: Rollout allocation for group-based rl post-training, 2026. URL https://arxiv.org/abs/2605.26606.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 3843–3857. Curran Associates, Inc., 2022. doi: 10.52202/068431-0278. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/18abbeef8cfe9203fdf9053c9c4fe191-Paper-Conference.pdf.

Xiaoyuan Li, Yubo Ma, Chengpeng Li, Fengbin Zhu, Yiyao Yu, Keqin Bao, Wenjie Wang, Fuli Feng, and Dayiheng Liu. Unified data selection for llm reasoning, 2026. URL https://arxiv.org/abs/2605.22389.

Shuang Liang, Haoyang Zhou, Yifan Gong, Guowei Wang, and Xiting Wang. Leeps: Latent-guided explore-exploit prompt sampling for efficient rlvr in large language models, 2026. URL https://arxiv.org/abs/2607.28077.

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, 2023. URL https://arxiv.org/abs/2305.20050.

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 B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/aca97732e30bcf1303bc22ac3924fd16-Paper-Conference.pdf.

Yun Qu, Qi Wang, Yixiu Mao, Vincent Tao Hu, Björn Ommer, and Xiangyang Ji. Can prompt difficulty be online predicted for accelerating rl finetuning of reasoning models?, 2026. URL https://arxiv.org/abs/2507.04632.

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, 2024a. URL https://arxiv.org/abs/2402.03300.

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, 2024b. URL https://arxiv.org/abs/2402.03300.

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

Yifan Sun, Jingyan Shen, Yibin Wang, Tianyu Chen, Zhendong Wang, Mingyuan Zhou, and Huan Zhang. Improving data efficiency for llm reinforcement fine-tuning through difficulty-targeted online data selection and rollout replay, 2026. URL https://arxiv.org/abs/2506.05316.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui 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. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=yfcpdY4gMP.

Jiahao Wu, Ning Lu, Shengcai Liu, Kun Wang, Yanting Yang, Baijiong Lin, Chen Jason Zhang, Li Qing, and Ke Tang. Train at moving edge: Online-verified prompt selection for efficient rl training of large reasoning model, 2026. URL https://arxiv.org/abs/2603.25184.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, 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 *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=2a36EMSSTp.

Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, Xipeng Qiu, Tianxiang Sun, Peng Li, Shiqiao Meng, Yanjun Zheng, Jun Zhan, Zhangyue Yin, Xiannian Hu, Guofeng Quan, and Qixiang Wang. Evaluating the performance of large language models on gaokao benchmark, 2026. URL https://arxiv.org/abs/2305.12474.

Haizhong Zheng, Yang Zhou, Brian R. Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for LLM reasoning via selective rollouts. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=x5lITYXmW2.

A Training and Evaluation Hyperparameters

We use the same policy-optimization, rollout, and evaluation hyperparameters for Qwen2.5-Math-1.5B and Qwen2.5-Math-7B, specified in Tables 2 and 3.

Table 2: Policy-optimization hyperparameters.

Hyperparameter	**Value**
Training batch size	256 prompt groups
Rollout group size G	8
Optimizer	AdamW
Learning rate	10^{-6}
Adam coefficients (β_1, β_2)	$(0.9, 0.999)$
Weight decay	0.01
Warmup steps	0
Minibatch size	64 prompt groups
Clipping range ϵ_{clip}	0.2
Maximum gradient norm	1.0
Entropy coefficient	10^{-3}
KL reward coefficient	0
KL regularization coefficient β	0

Table 3: Rollout generation and evaluation hyperparameters.

Hyperparameter	**Value**
Maximum prompt length	1,536 tokens
Maximum rollout length	2,560 tokens
Training temperature	1.0
Training top-p	1.0
Training top-k	Disabled
Evaluation rollouts per prompt	4
Evaluation temperature	1.0
Evaluation top-p	0.7
Evaluation top-k	Disabled

All runs use one node with four GPUs, PyTorch FSDP for optimization, and vLLM for rollout generation. Tensor parallelism is one, gradient checkpointing and padding removal are enabled, and the reference policy is CPU-offloaded for both model sizes. Actor parameters and optimizer state remain on GPU for the 1.5B model and are CPU-offloaded for the 7B model.

B Omitted details for Section 3.1

This section specifies the Beta predictor training during policy optimization, and how its output is converted into a probability that predicts if a partially observed prompt group will become non-degenerate. There is also further discussion about the construction of Figure 1.

B.1 Details on the training of the predictor MLP

Online construction of training examples. We collect the prompts that have gone through at least two rollouts from the past 5 RL training iterations, along with their past rollouts from previous epochs. Rollouts from past epochs will be used as solely the feature, whereas the rollouts from the current epoch will be split into input features and labels. Consider a prompt p_i for which in the current epoch, c_i responses have been generated in order, with binary rewards $r_i^{(1)}, \ldots, r_i^{(c_i)}$. We form a supervised example at every possible prefix depth for each prompt:

$$\mathcal{N}_i = \{0, \ldots, , c_i - \ell_{\min}\}, \qquad \ell_{\min} = 2. \tag{14}$$

At depth $n \in \mathcal{N}_i$, the first n responses constitute the currently observed prefix supplied to the MLP. The responses that were actually generated after that prefix form the prediction target:

$$m_{i,n} = c_i - n, \qquad s_{i,n} = \sum_{j=n+1}^{c_i} r_i^{(j)}. \tag{15}$$

By setting $\ell_{\min} = 2$, we ensure that each training example updates both μ_i and ν_i during training.

Now, a prompt that supports more prefix depths would naturally appear multiple times more often in the objective. Therefore, we assign every row derived from prompt i the weight

$$w_{i,n} = \frac{1}{|\mathcal{N}_i|}, \tag{16}$$

so that each prompt contributes a total weight of one. Response order is also preserved since later responses may have been generated as a result of decisions made from earlier outcomes.

For history rollouts from previous epochs, some prompts may have fewer generation history compared to others. Also when we rely on only the history rollouts to predict the model, current rollout is also not available. In case of unavailable slots, we replace the 4d encoded features to 0.

Beta-Binomial supervision. For each prefix, the MLP predicts $\mu_{i,n} \in (0, 1)$ and $\nu_{i,n} > 0$, which define

$$\alpha_{i,n} = \mu_{i,n}\nu_{i,n}, \qquad \beta_{i,n} = (1 - \mu_{i,n})\nu_{i,n}. \tag{17}$$

From this, the likelihood of observing $s_{i,n}$ successes among the $m_{i,n}$ suffix responses is:

$$P(s_{i,n}, m_{i,n} - s_{i,n} \mid \alpha_{i,n}, \beta_{i,n}) = \binom{m_{i,n}}{s_{i,n}} \frac{\prod_{j=0}^{s_{i,n}-1}(\alpha_{i,n} + j) \prod_{j=0}^{m_{i,n}-s_{i,n}-1}(\beta_{i,n} + j)}{\prod_{j=0}^{m_{i,n}-1}(\alpha_{i,n} + \beta_{i,n} + j)} \tag{18}$$

We minimize the weighted negative log likelihood:

$$\mathcal{L}(\theta) = -\frac{1}{\sum_{i,n} w_{i,n}} \sum_{i,n} w_{i,n} \log P(s_{i,n}, m_{i,n} - s_{i,n} \mid \alpha_{i,n}, \beta_{i,n}) + \frac{\lambda}{\sum_{i,n} w_{i,n}} \|\theta\|_2^2, \qquad \lambda = 1, \tag{19}$$

Such a setup identifies both the mean and concentration of the predicted Beta distribution, unlike what is possible with binary supervision on the final group label. It also allows partial groups to act as training data through their observed suffix counts.

Refitting the predictor. The predictor is refitted once after every successful sampling cycle using at most five of the most recent observation batches. All used response features in this window are standardized using the shared mean and standard deviation. We optimize the small network on CPU with L-BFGS, learning rate 1, a strong-Wolfe line search, and at most 50 iterations. Parameters are warm-started across iterations. Each MLP refitting (which contains up to 50 iterations) takes 59ms to 93ms on CPU, depending on the number of history epochs used.

B.2 Further details on Figure 1

We explain how Figure 1 is plot. For each row, we collected the prompts and its rollout from iterations 6-10, 16-20, $106 - 110$ of vanilla GRPO (without any filtering) with group size $G = 8$

respectively. We then sort every single rollout by the feature of the subplot. We apply a rolling window of 5% of the rollouts and gather the prompts corresponding to the rollouts within that rolling window. The non-degeneracy probability is then calculated as the percentage of non-degenerated prompts within that window.

serves to show how the entropy-non-degeneracy relationship changes as the policy evolves. Each prompt occurrence i contains $G = 8$ responses with entropy statistics $H_{i,1}, \ldots, H_{i,G}$ and binary rewards $r_{i,1}, \ldots, r_{i,G}$.

C Additional details for Selective Rollout Generation

Initialization of prompt selection In the first epoch, all prompts are new to the predictor so we force each prompt to generate one rollout to initialize the information about the prompt. Since the second epoch, history information is obtained, and thus evaluation on all the prompts is unnecessary, which explains the drop in the cost in Figure 4 after the first epoch. When $W_{\text{past}} = 0$, selection according to Algorithm 2 would generate one rollout for the highest-ranked subset. Therefore, we instead sort the prompts by the marginal tradeoff and take a subset containing the ones with the most efficient tradeoff. The subset size is $\lceil 0.9\bar{H} \rceil$, clipped to the number of rankable prompts, where $\bar{H}$ is the exponential moving average of non-degenerate groups harvested in preceding successful cycles.

Mismatch of the optimal tradeoff and the rollout budget The initialization in the previous paragraph aims to ensure that when optimal tradeoff is obtained and the algorithm halts, the rollout budget is consumed. Yet there's no guarantee. In practice, if we have gathered enough non-degenerated groups or have used enough rollouts, we early step the generation even though additional prompts can be selected per Algorithm 2; and if the algorithm halts with no additional prompts satisfying Equation (10), and we still wish to generate more rollouts, we relax the condition by slightly increasing the J^* to allow more prompts to be generated.

Discussion on the dispatch cost threshold In Algorithm 3, J^* of the current round of predictor rollout was used as the threshold, yet in practice, the optimal tradeoff in future rounds may be higher or lower. We decay the J^* heuristically towards G to ensure a stricter rollout dispatch.

D Omitted theorems and proofs in the main body

Proof of Theorem 1. Since S is finite, a minimizer exists. Let $\mathcal{S}$ minimize J over feasible subsets, and write $C = W_{\text{past}} + \sum_{i \in \mathcal{S}} m_i$, $T = H_{\text{past}} + \sum_{i \in \mathcal{S}} t_i > 0$, and $J^* = C/T$. For any omitted prompt $p_j \notin \mathcal{S}^*$ with $t_j > 0$, optimality gives

$$0 J(\mathcal{S} \cup \{j\}) - J^* = \frac{C + m_j}{T + t_j} - \frac{C}{T} = \frac{T m_j - C t_j}{T(T + t_j)} = \frac{t_j (c_j - J^*)}{T + t_j},$$

so $c_j > J^*$. For any selected prompt $p_i \in \mathcal{S}^*$ with $T - t_i > 0$, optimality likewise gives

$$0 \le J(\mathcal{S} \setminus \{i\}) - J^* = \frac{C t_i - T m_i}{T(T - t_i)} = \frac{t_i (J^* - c_i)}{T - t_i},$$

so $c_i \le J^*$. If $T = t_i$, then $\mathcal{S} = \{i\}$ and $J^* = (W_{\text{past}} + m_i)/t_i = c_i + W_{\text{past}}/t_i > c_i$. Thus every selected prompt has $c_i \le J^*$ and every omitted prompt has $c_i > J^*$, which means that the selected prompts form a prefix of the sorted list. □

Theorem 2. *Let $0 \le n_i < G$ and $\alpha_i, \beta_i > 0$. For every integer $0 \le r < G - n_i$,*

$$\max_{\substack{k_0 + k_1 = r \\ k_0, k_1 \ge 0}} c_i^{look}(k_0, k_1) = \max\{c_i^{look}(r, 0), c_i^{look}(0, r)\}.$$

Proof. Fix r and set $m = G - n_i - r > 0$. If $k_0, k_1 > 0$ and $k_0 + k_1 = r$, then

$$0 < \sum_{j=1}^{n_i^{\text{look}}} r_i^{(j)} < n_i^{\text{look}},$$

so

$$\mathbf{1}\left\{\sum_{j=1}^{n_i^{\text{look}}} r_i^{(j)} = 0\right\} = \mathbf{1}\left\{\sum_{j=1}^{n_i^{\text{look}}} r_i^{(j)} = n_i^{\text{look}}\right\} = 0.$$

By Equations (4) and (12),

$$c_i^{\text{look}}(k_0, k_1) = \frac{m}{t_i^{\text{look}}(k_0, k_1)} = m.$$

Since $0 < t_i^{\text{look}}(r, 0), t_i^{\text{look}}(0, r) \leq 1$,

$$\begin{aligned} c_i^{\text{look}}(k_0, k_1) = m &\leq \max\left\{\frac{m}{t_i^{\text{look}}(r, 0)}, \frac{m}{t_i^{\text{look}}(0, r)}\right\} \\ &= \max\{c_i^{\text{look}}(r, 0), c_i^{\text{look}}(0, r)\}. \end{aligned}$$

If $k_0 = 0$ or $k_1 = 0$, then $(k_0, k_1) \in \{(0, r), (r, 0)\}$, and the same bound holds directly. Taking the maximum over all feasible pairs gives one inequality; feasibility of both endpoints gives the reverse inequality. □