POLY-EPO: TRAINING EXPLORATORY REASONING MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Exploration is a cornerstone of learning from experience: it enables agents to find solutions to complex problems, generalize to novel ones, and scale performance with test-time compute. In this paper, we present a framework for post-training language models (LMs) that explicitly encourages optimistic exploration and promotes a synergy between exploration and exploitation. The central idea is to train the LM to generate sets of responses that are collectively accurate under the reward function and exploratory in their reasoning strategies. We first develop a general recipe for optimizing LMs with set reinforcement learning (set RL) under arbitrary objective functions, showing how standard RL algorithms can be adapted to this setting through a modification to the advantage computation. We then propose Polychromic Exploratory Policy Optimization (POLY-EPO), which instantiates this framework with an objective that explicitly synergizes exploration and exploitation. Across a range of reasoning benchmarks, we show that POLY-EPO improves generalization, as evidenced by higher pass@k coverage, preserves greater diversity in model generations, and effectively scales with test-time compute.

1 INTRODUCTION

Exploration is a fundamental component of learning from experience. First, it is necessary for an agent to discover a successful strategy when learning to solve a highly complex problem, such as identifying a valid proof technique for a difficult theorem. Second, it enables generalization to novel problems by helping the agent build a diverse repertoire of successful strategies; an agent learning to play chess should ideally master multiple opening lines rather than rely on a single strategy that may fail against a different opponent at test time. Third, exploration is also essential for the scalability of test-time compute methods, such as verification, aggregation, and search, whose effectiveness depends on diversity in the model's generations (Wang et al., 2023; Yao et al., 2023; Besta et al., 2024; Zhu et al., 2025).

While reinforcement learning (RL) has substantially improved the reasoning capabilities of post-trained language models (LMs), effective exploration remains a fundamental challenge. Although pretrained base models exhibit high initial diversity in their generations, RL fine-tuning rapidly collapses this diversity onto a narrow set of high-reward behaviors (Yue et al., 2025; Cui et al., 2025; Song et al., 2025). To address this problem effectively, an algorithm should ideally satisfy the following desiderata:

1. The algorithm should *encourage optimism under uncertainty* (Lai & Robbins, 1985; Auer et al., 2002). In other words, it should assign a positive learning signal to trajectories that attempt novel reasoning strategies, even when those strategies have not yet yielded high reward. This is to ensure that the policy continues to explore promising strategies and broaden its capabilities during training. For example, when attempting to solve an algorithmic problem, an agent exploring a dynamic programming formulation should continue refining that strategy even if its initial implementations fail unit tests, rather than immediately reverting to a familiar but less scalable exhaustive search approach.
2. The algorithm should *encourage the model to achieve a synergy between exploration and exploitation*. In other words, the learning signal for each generation should depend on

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.

whether that generation helps the model jointly achieve high task performance and explore diverse strategies. This enables the model to learn, through optimization, to balance between these goals.

3. The algorithm must be scalable for large scale language model post-training efforts. In particular, the algorithm's sample complexity and computational complexity should not be a significant overhead.

Standard RL algorithms, such as PPO (Schulman et al., 2017b) or GRPO (Shao et al., 2024), do not explicitly incentivize the optimistic exploration required to master diverse reasoning paths. Prior work has attempted to address this issue through reward shaping, typically by augmenting the canonical reward with a weighted exploration bonus (Cheng et al., 2025; Tuyls et al., 2025; Song et al., 2025; Dai et al., 2025; Zhang et al., 2025). Broadly, these methods attempt to solve for a given input x, the problem $\max_\theta \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[r(x,y) + \lambda d(x,y)\right]$, where π_θ is the parameterized policy, $r(x,y)$ is the task reward and $d(x,y)$ is an exploration bonus, such as an entropy bonus, UCB bonus, or semantic diversity bonus. However, this formulation treats exploitation and exploration as separate objectives and does not explicitly encourage achieving these goals synergistically. A generation can receive high learning signal by being reward-maximizing or by being exploratory, without regard to whether it helps the model strike a balance between exploration and exploitation. Consequently, the method depends on careful tuning or scheduling of λ, and even then, does not directly optimize for a persistent balance.

In this paper, we develop a principled and scalable approach for post-training language models that balances exploration and exploitation through *set reinforcement learning* (set RL) (Hamid et al., 2026). Set RL optimizes a policy to sample an optimal *set* of generations according to a set-level objective. Our contributions are as follows:

1. **A general recipe for set RL on language models.** We introduce a general and scalable recipe for optimizing language models under arbitrary set objectives that avoids the complexity of naively sampling sets independently. We show how standard RL algorithms, such as GRPO (Shao et al., 2024), can be adapted towards this end, providing a practical training paradigm.
2. **Polychromic Exploratory Policy Optimization.** Building on this framework, we propose Polychromic Exploratory Policy Optimization (POLY-EPO), which optimizes a *polychromic objective* that scores sets as the product of their average reward and the diversity of the high-level reasoning strategies they contain. We introduce a scalable procedure for estimating the diversity in sampled responses by using a language model judge (LM-judge) to cluster all generations based on their reasoning strategy.
3. **Synergistic exploration and exploitation.** We show that our algorithm's advantage function incorporates the covariance between average reward and diversity of strategies across sets. As such, the learning signal of POLY-EPO inherently harmonizes exploration and exploitation.

We evaluate our method on reasoning tasks that admit multiple valid solution strategies, as well as on large-scale mathematical reasoning benchmarks. Our experiments show that POLY-EPO improves generalization by achieving higher pass@k coverage with gains of up to 20% on mathematical reasoning test sets. The method also scales more effectively with test-time compute under majority voting. Finally, our analysis of training dynamics reveals greater diversity in model generations and broader coverage of training-set problems.

2 PRELIMINARIES

In this section, we look at an overview of the set reinforcement learning framework (Hamid et al., 2026). Set reinforcement learning (set RL) (Hamid et al., 2026) is a framework that generalizes standard RL by assigning reward to sets of sampled actions or generations, rather than to each one independently. Importantly, the reward function couples all the actions or generations under a shared learning signal. We discuss only the single-turn formulation of set RL here since that is the focus of our paper and refer the reader to section C for an overview of the sequential formulation.

Language Model Formulation. In single-turn settings, such as in language model training, we consider a set-level objective over n i.i.d. generations from the same prompt. Let $y_{1:n} := \{y_1, \ldots, y_n\}$ where each $y_i \overset{\text{i.i.d.}}{\sim} \pi_\theta(\cdot \mid x)$, and let $f(x, y_{1:n})$ be our set-level objective. We will assume that f is symmetric in its arguments $y_1, \ldots, y_n$. Then, set RL aims to solve:

$$\max_\theta \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y_{1:n}\sim\pi_\theta(\cdot|x)}\Big[f(x, y_{1:n})\Big]. \tag{1}$$

In other words, we use a set-level reward that is shared by all generations within the set, which is natural for training objectives that require generations to collectively exhibit some desirable property.

Note that we require that the set objective function f is such that we cannot write $\mathbb{E}_{y_{1:n}\sim\pi_\theta(\cdot|x)}[f(x, y_{1:n})]$ as $\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[g(x, y)]$ where g is independent of θ for all policy parameters $\theta \in \Theta$. This distinguishes set RL from the standard RL setting, in which optimization is driven by per-generation rewards. This is an important defining feature because it is always possible to define $g_\theta(x, y) = \mathbb{E}_{y_{2:n}}[f(x, y, y_{2:n})]$, so that $\mathbb{E}_{y_{1:n}}[f(x, y_{1:n})]$ as $\mathbb{E}_y[g_\theta(x, y)]$. However, this does not reduce the problem to standard RL, since the induced reward g_θ depends on the policy itself. In particular, it is not an externally specified, policy-independent reward function. This distinguishes the set RL framework from other multi-sample frameworks where one uses leave-one-out estimators to collapse the shared learning signal to an individualized one (Tang et al., 2025).

Using the score-function identity to the joint density $\prod_{i=1}^n \pi_\theta(y_i \mid x)$, we get the following policy gradient:

$$\nabla_\theta \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y_{1:n}\sim\pi_\theta(\cdot|x)}[f(x, y_{1:n})] = \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y_{1:n}\sim\pi_\theta(\cdot|x)}\left[(f(x, y_{1:n}) - \hat{f}(x))\sum_{i=1}^n \nabla_\theta \log \pi_\theta(y_i \mid x)\right] \tag{2}$$

where $\hat{f}(x)$ is a set-level baseline. The defining feature of eq. (2) is that all generations in the sampled set $y_{1:n}$ share the same scalar learning signal, $f(x, y_{1:n}) - \hat{f}(x)$. Note that, $\hat{f}(x)$ is not defined on any proper subset of $y_{1:n}$, which disallows leave-one-out estimators for example. This is fundamentally different from standard RL, where each generation receives its own advantage. In set RL, the objective f couples the samples in the set, and the gradient assigns equal credit to all elements in the set through a shared set-level signal. Leave-one-out estimators are disallowed because it breaks the shared set-level credit assignment.

The main practical challenge with eq. (2) is statistical and computational: a naive empirical estimator would require repeatedly sampling full sets of size n for each prompt in order to estimate both the set reward and the baseline with low variance. In the next section, we show how to construct a more efficient estimator that retains the set-RL learning signal while being practical for large-scale LM training.

3 Optimizing Language Models with Set RL

In this section, we discuss a general recipe for optimizing language models (LMs) using the set RL framework under any arbitrary objective function f that is symmetric in its arguments. In particular, we will find a tractable estimator of the set RL policy gradient in eq. (2). The key ingredient will be a method for constructing the appropriate advantage function for a generation corresponding to the set RL framework.

Suppose that our set objective, $f(x, y_1, \cdots, y_n)$, is defined over sets of size n. If we wanted to use an empirical estimate (e.g., a Monte Carlo estimate) of the gradient in eq. (2), then we would need to sample several sets of size n conditioned on a single prompt which can be expensive in terms of sample complexity. In this section, we consider how we can instantiate set reinforcement learning using a computationally feasible gradient estimator.

Set Construction. For a fixed prompt x, we first draw N i.i.d. generations $y_1, \ldots, y_N \sim \pi_\theta(\cdot \mid x)$. We require that the number of samples, N, is greater than the size of sets, n, in the set RL framework. Next, from these samples, we construct K sets, $G_1, \cdots, G_K$, of size $n < N$, in a combinatorial fashion without replacement: $G_j = \{y_{(j_1)}, \ldots, y_{(j_n)}\}, \quad j_1 < \cdots < j_n, \{j_1, \ldots, j_n\} \subseteq \{1, \ldots, N\}$.

Algorithm 1 Gradient of On-Policy Implementation of Set RL

Require: Set objective f, Policy π_θ, batch of inputs B, number of rollouts N, set size n

for each input $x \in B$ **do**
  Sample $y_1, \ldots, y_N \sim \pi_\theta(\cdot \mid x)$
  // Construct sets either by enumerating all $\binom{N}{n}$ subsets or by uniformly sampling K subsets from them
  Construct sets $G_1, \cdots, G_K$ from $\{y_{1:N}\}$ without replacement
  Compute score of each set $\{f(x, G_l)\}_{l=1}^K$ and baseline $b = \frac{1}{K}\sum_{l=1}^K f(x, G_l)$
  Compute set advantage of each set $\widehat{A^\sharp}(x, G_l; f) = f(x, G_l) - b$ for $l = 1, \cdots, K$
  // Let $\mathcal{G}(y)$ be all the sets in $G_1, \cdots, G_K$ that contain y
  Compute marginal set advantage $\widehat{A^\sharp_{\text{marg}}}(x, y_j; f) := \frac{1}{|\mathcal{G}(y_j)|}\sum_{G \in \mathcal{G}(y_j)} \widehat{A^\sharp}(x, G)$ for $j = 1, \cdots, N$
  $\hat{g}(x) \leftarrow \sum_{j=1}^N \nabla_\theta \log \pi_\theta(y_j \mid x) \cdot \widehat{A^\sharp_{\text{marg}}}(x, y_j; f)$
end for
$\hat{g} \leftarrow \frac{1}{|B|}\sum_{x \in B} \hat{g}(x)$
return $\hat{g}$

The maximum number of sets that can be constructed in this manner is $K = \binom{N}{n}$. Using all $\binom{N}{n}$ sets can reduce the variance of the resulting gradient estimator. When evaluating f is computationally expensive, scoring all $\binom{N}{n}$ sets may be impractical. In this case, we instead sample K sets uniformly (without replacement) from the collection of all $\binom{N}{n}$ possible sets. Either of these choices leads us to an unbiased estimator of the set RL gradient as we will see later.

Set Scoring. Now that we have constructed K sets, for each constructed set G_j, we compute the set-level score under the set objective function: $f(x, G_j) = f\big(x, y_{(j_1)}, \ldots, y_{(j_n)}\big)$. We construct a variance-reduction baseline using the sampled sets: $\hat{f}(x) := \frac{1}{K}\sum_{j=1}^K f(x, G_j)$. In other words, the baseline is simply an average of the scores of all sets under our objective function f. Given the baseline, for each set G_j, we can now construct a set advantage function which is simply

$$\widehat{A^\sharp}(x, G_j; f) = f(x, G_j) - \hat{f}(x). \tag{3}$$

So far, we have computed everything at the set-level, i.e. we constructed sets, scored them and computed each set's advantage. We now derive an advantage function at the level of a single generation.

Marginal set advantage of a single generation. Intuitively, we will set the advantage of each generation to be simply the sum of the advantages of all sets containing the generation. Let $\mathcal{G}(y)$ be the collection of all sets that contain the fixed generation y. Note that $\mathcal{G}(y) = \{G \in G_{1:K} \mid y \in G\} \subset G_{1:K}$. We define the marginal set advantage of the generation y to be:

$$\widehat{A^\sharp_{\text{marg}}}(x, y; f) := \frac{1}{|\mathcal{G}(y)|}\sum_{G \in \mathcal{G}(y)} A^\sharp(x, G; f). \tag{4}$$

We will soon motivate the construction of this advantage function from the point of view of getting an unbiased estimator for the set RL gradient and, in §5, through a deeper connection between set RL and standard RL.

Final Estimator. Using this, we use the following set RL gradient estimator:

$$\nabla_\theta \widehat{\mathbb{E}}_{x \sim \mathcal{D}, y_{1:n} \sim \pi_\theta(\cdot|x)}\left[f(x, y_{1:n})\right] \propto \widehat{\mathbb{E}}_{x \sim \mathcal{D}, y_1, \cdots, y_N \sim \pi_\theta(\cdot|x)}\left[\sum_{i=1}^N \nabla_\theta \log \pi_\theta(y_i \mid x)\widehat{A^\sharp_{\text{marg}}}(x, y_i; f)\right]. \tag{5}$$

Now, we have all the ingredients required for optimizing an LM with respect to the set RL framework. Since the marginal set advantage gives us an advantage function over single generations, we can use existing *standard* RL algorithms, such as PPO Schulman et al. (2017b) or GRPO Shao et al. (2024), to optimize the set objective by replacing their standard advantage, $A(x, y_i)$, with $\widehat{A^\sharp_{\text{marg}}}(x, y_i)$. The pseudocode for our general recipe is provided in algorithm 1.

Note that our estimator is an unbiased estimator of the true policy gradient of set RL (up to a scaling factor). This is true for both the case where we construct all $\binom{N}{n}$ sets and the case where we sample K sets uniformly without replacement. This is particularly helpful because if one wants to use an objective that is computationally expensively, constructing all $\binom{N}{n}$ sets is impractical. We show the unbiasedness in the following proposition:

Proposition 3.1. *Fix a prompt x, and let $y_1, \cdots, y_N \overset{\text{i.i.d.}}{\sim} \pi_\theta(\cdot \mid x)$ be our independently sampled N generations and let $f : \mathcal{X} \times \mathcal{Y}^{\oplus n} \to \mathbb{R}$ be our set objective. Then,*

$$\mathbb{E}[\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(y_i \mid x) \widehat{A^\sharp_{\text{marg}}}(x, y_i; f)] = M \nabla_\theta \mathbb{E}_{y_{1:n} \sim \pi_\theta(\cdot|x)} \big[f(x, y_{1:n}) \big],$$

where $M \in \mathbb{R}_{>0}$ is a constant scaling factor that depends on the number of sets we construct. Consequently, after also taking expectation over $x \sim \mathcal{D}$ and scaling the learning rate, the estimator is an unbiased estimator of the set RL gradient.

The proof of this is provided in §E. The main idea of the proof is that constructing the sets in a combinatorial fashion causes our proposed gradient to be a U-statistic estimator of the set RL gradient which is unbiased. In §D.1, we revisit this question from a different point of view and justify why set RL can be implemented using standard RL algorithms. In particular, we show how the construction of the marginal set advantage of a generation justifies using standard RL algorithms to optimize set RL by investigating how the logits of a softmax policy changes over gradient updates.

4 Polychromic Exploratory Policy Optimization

In this section, we introduce our method, Polychromic Exploratory Policy Optimization (Poly-EPO). We use the set RL framework discussed in §3 to optimize an objective that explicitly encourages the model to balance exploration and exploitation. Concretely, we want the model to increase the likelihood of generations that are not only accurate with respect to our reward function, but also explore diverse reasoning strategies. To do so, we optimize the *polychromic objective* (Hamid et al., 2026) using set RL.

Polychromic Objective. We use the polychromic objective adapted to the LM reasoning setting:

$$f_{\text{poly}}(x, y_1, \cdots, y_n) = \frac{1}{n} \sum_{i=1}^{n} r(x, y_i) \cdot d(x, y_1, \cdots, y_n), \tag{6}$$

where $d(x, y_1, \cdots, y_n)$ is a function that measures the diversity of the set of generations, $y_{1:n}$. Notably, due to the advantage being shared by generations in a set, under polychromic objectives, an unsuccessful generation (i.e. a generation y such that $r(x, y) = 0$) can still receive a positive set-advantage (i.e., $A^\sharp_{\text{marg}}(x, y; f_{\text{poly}}) > 0$) if it is highly exploratory.

Intuitively, the polychromic objective assigns a score to a set of generations that is dependent on both its ability to explore diverse strategies and exploit the reward signal. Since this is a product and since each of these two factors is lower bounded by 0, the set must optimize both exploration and exploitation, and cannot ignore one for the other.

Diversity Function. We now discuss how we measure diversity within a set of generations. Our main ingredient is a language-model judge (LM-judge) used to cluster responses sampled from the policy according to their underlying reasoning strategy or semantic approach. Because LM-judges are already integrated into modern post-training pipelines—including settings such as proof generation and other non-verifiable tasks—this design allows Poly-EPO to leverage infrastructure that is already widely deployed. Moreover, clustering responses by semantic similarity is typically a substantially easier problem than solving the original task itself, which allows our method to exploit the generator–verifier gap.

Recall that, for each prompt x, we first sample N independent generations and then construct sets of size n from them. We begin by taking all N generations associated with the same prompt and querying an LM-judge to cluster them according to the semantic similarity of their reasoning strategies. Let

$\mathcal{C}(y_i) \in 1, \ldots, N)$ denote the cluster assignment of generation y_i. We then define the diversity of a set $(y_1, \ldots, y_n)$ as

$$d\big(x, y_{1:n}\big) = \frac{\text{No. of distinct clusters represented in } y_{1:n}}{n} = \frac{|\{\mathcal{C}\left(y_1\right), \ldots, \mathcal{C}\left(y_n\right)\}|}{n}. \tag{7}$$

By construction, $d(x, y_{1:n}) \in [0, 1]$. The pseudocode for computing the polychromic objective score for each set is provided in algorithm 2. To improve judge reliability, we use in-context learning to steer the LM-judge toward clustering generations based on their underlying strategy—capturing both high-level and low-level reasoning structure—rather than superficial differences such as tone, phrasing, or writing style. The full judge prompt is provided in the appendix (see §B.1), where we detail the rubric, few-shot exemplars, and other design choices used to improve clustering quality.

Final Algorithm. The final algorithm, Polychromic Exploratory Policy Optimization (POLY-EPO), computes the marginal set-RL advantage of each generation under the polychromic objective and then plugs this signal into any standard RL update rule. We sample N generations independently per prompt, exactly as in standard RL algorithms such as GRPO and CISPO. This amounts to instantiating the general set-RL recipe in algorithm 1, with set scores being under the polychromic objective computed using the diversity function in eq. (7). Additional implementation details can be found in section B.

5 ANALYSIS

In this section, we will analyse how the polychromic objective encourages the policy to synergistically explore and exploit. In set RL, a generation y receives increased probability mass (i.e., $\log \pi_\theta^{k+1}(y \mid x) - \log \pi_\theta^k(y \mid x) > 0$) if and only if it has a positive marginal set advantage and non-zero probability under the rollout policy (see figure D.1). Consider a fixed generation $y \in \mathcal{Y}$ with $\pi_\theta(y \mid x) > 0$ and the polychromic objective f_{poly}. The marginal set advantage of y can be decomposed as follows:

$$\begin{aligned} A^{\sharp}_{\text{marg}}(x, y; f_{\text{poly}}, \theta) = \; & \underbrace{(\frac{1}{n} r(x, y) + \frac{n-1}{n} \mathbb{E}_Y[r(x, Y)]) \mathbb{E}_{Y_{2:n}}[d(x, y, Y_{2:n})]}_{\textit{Term 1}\text{: Mean reward of sets containing } y \,\times\, \text{Mean diversity of sets containing } y} \\ & + \underbrace{\text{Cov}_{Y_{2:n}}(\frac{1}{n} r(x, y) + \frac{1}{n} \sum_{i=2}^{n} r(x, Y_i), d(x, y, Y_{2:n}))}_{\textit{Term 2}\text{: Exploration–Exploitation synergy in sets containing } y} \quad - \quad \underbrace{b(\theta)}_{\text{baseline}}. \end{aligned} \tag{8}$$

This decomposition reveals how set RL with polychromic objective evaluates each generation. We now interpret the two terms that depend on the fixed generation y.

- *Term 1* reflects the expected contribution of generation y to the product of reward and diversity in sets that contain it. This term depends on both the reward and diversity contributions of other generations in the set. Even if y itself does not contribute reward (or diversity), this term can still be positive provided other sampled generations contribute reward (or diversity) in expectation. **Generations that explore optimistically receive a positive learning signal even if they failed to yield high rewards**.
- *Term 2* captures how the presence of generation y affects the covariance between reward and diversity across sets containing y. Intuitively, this term favors generations that enable sets to simultaneously achieve high reward and high diversity. **Generations that help achieve a synergy between exploration and exploitation receive positive learning signal**.

This directly reflects the two desiderata that motivate our method. First, the objective supports *optimistic exploration*: trajectories that attempt novel reasoning strategies can receive positive learning signal even before they become individually successful. Second, the objective *intrinsically encodes the exploration–exploitation trade-off*: the advantage assigned to a trajectory depends in part on whether it helps the model form sets that jointly achieve high reward and high diversity, rather than on a manually tuned weighting coefficient. In this sense, the policy update itself internalizes the balance between exploration and exploitation.

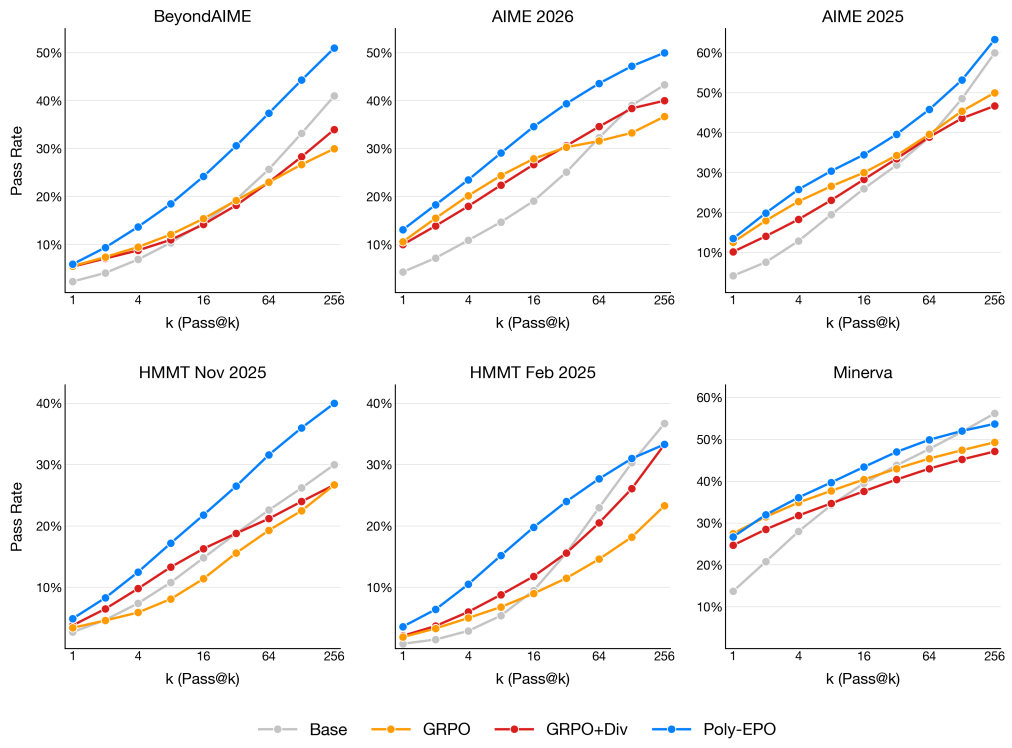


Figure 1: **Pass@k evaluations on test sets**. The x-axis is the number of attempts k used in the evaluation while the y-axis is the coverage of the test set. The base model used here is `Qwen3-4B-Base` and the LM-judge used by POLY-EPO and GRPO+DIV for clustering responses is `Qwen-3-4B-Instruct`.

6 EXPERIMENTS

We now empirically evaluate POLY-EPO. In this section, we train and evaluate on mathematical reasoning benchmarks. In §F.1, we study synthetic domains with verifiable rewards and infinitely many reward-maximizing strategies. Our goal is to test whether the exploration induced by POLY-EPO translates into better reasoning performance, in particular through improved generalization to unseen problems and more effective use of test-time compute. We use `Qwen-3-4B-Base` as our pretrained base model and train on `POLARIS-53k` (An et al., 2025), a dataset of mathematical reasoning problems. We use 128 prompts per batch, 8 rollouts per prompt, and 2 epochs of RL training. We evaluate on a broad suite of held-out benchmarks: BeyondAIME (Seed et al., 2025), AIME 2026, AIME 2025, HMMT November 2025 (Balunović et al., 2025), HMMT February 2025 (Balunović et al., 2025), and Minerva (Lewkowycz et al., 2022).

We compare against GRPO (Shao et al., 2024) and against a standard RL baseline with an exploration bonus, which we denote by GRPO+DIV. Its objective is

$$\mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[r(x,y)+\lambda\, d(x,y\mid y_{1:N})\right]. \tag{9}$$

Here, $d(x,y\mid y_{1:N})$ is a diversity bonus defined using the same LM-judge-based clustering procedure as in POLY-EPO. Specifically, given rollouts $y_1,\dots,y_N$ sampled for prompt x, we first cluster these generations. Let $\mathcal{C}(y)$ be the set of generations that belong to the same cluster as y. Then define $d(x,y\mid y_{1:N}) = \left(\frac{N}{|\mathcal{C}(y)|}-1\right)/(N-1)$. Thus, the bonus assigned to a generation decreases as the size of its cluster increases. Conceptually, GRPO+DIV is similar to prior approaches that add exploration bonuses to the reward (Song et al., 2025; Tuyls et al., 2025).

We instantiate POLY-EPO and GRPO+DIV using `Qwen-3-4B-Instruct` as the LM-based judge for clustering generations. Note that the judge must possess strong instruction-following capabilities, as it is required to cluster responses according to a provided rubric that specifies grouping based

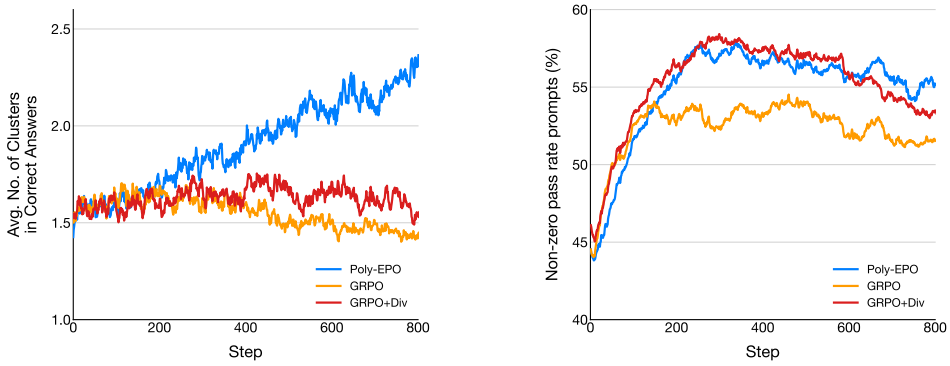


Figure 2: **Training dynamics on `POLARIS-53k`. Left:** Average number of unique reasoning-strategy clusters among the correct generations sampled for each prompt during training. Clusters are assigned by `Qwen-3-4b-Instruct`, which groups generations according to their high-level reasoning strategy. Higher values indicate a greater diversity of successful reasoning approaches. **Right:** Fraction of training prompts for which the model generates at least one correct rollout among its sampled responses during training.

on the underlying high-level reasoning strategy. Note that both POLY-EPO and GRPO+DIV use the same clustering method - the only difference is the diversity function that uses the clustering; POLY-EPO uses the clustering to score diversity of sets whereas GRPO+DIV computes the diversity contribution of single generations. Additional implementation details are provided in §B.

Pass@k Evaluations. We summarize the pass@k results in fig. 1. Overall, the pass@k performance of POLY-EPO improves more strongly as k increases, indicating that it produces generations with higher effective diversity. In contrast, GRPO and GRPO+DIV exhibit substantial pass@k degradation; indeed, the pretrained base model (`Qwen-3-4B-Base`) begins to outperform them as early as $k = 32$. This improved diversity under repeated sampling is especially valuable in domains such as mathematics, where one can pair generation with strong external verifiers such as Lean.

Training Dynamics. We now analyze the training dynamics. We first measure the average number of unique reasoning-strategy clusters among the correct generations sampled for each prompt, where cluster assignments are produced by the model `Qwen-3-4b-Instruct`. The results are shown in fig. 2 (left). For GRPO, this clustering is used purely for post-hoc evaluation and logging; no diversity signal is used during training. We observe that under GRPO, the number of clusters begins to decline after roughly 200 training steps, indicating a contraction in the diversity of successful strategies. Under GRPO+DIV, the metric remains approximately flat relative to its initial value, suggesting limited expansion of the successful strategy set. In contrast, POLY-EPO steadily increases the number of unique clusters to substantially higher levels than either baseline, indicating more effective exploration and the discovery of a broader range of successful reasoning approaches.

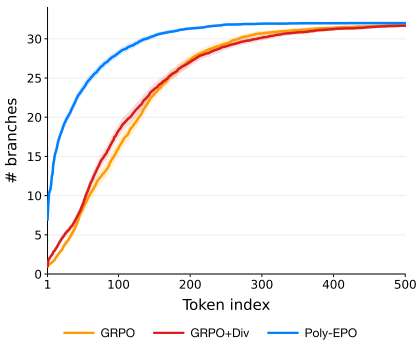


Figure 3: Average number of active branches as a function of token position on BeyondAIME. POLY-EPO promotes broader branching during reasoning generation.

We also complement the LM-judge clustering analysis with a token-level measure of generation diversity based on branching structure. For each test prompt, we sample k generations and organize them according to their shared token prefixes: two generations are said to branch at token h if they agree through token $h-1$ but differ at token h. As shown in fig. 3, POLY-EPO produces substantially earlier branching than GRPO, indicating that its generations diverge into distinct reasoning trajectories earlier in the rollout. Because early branching

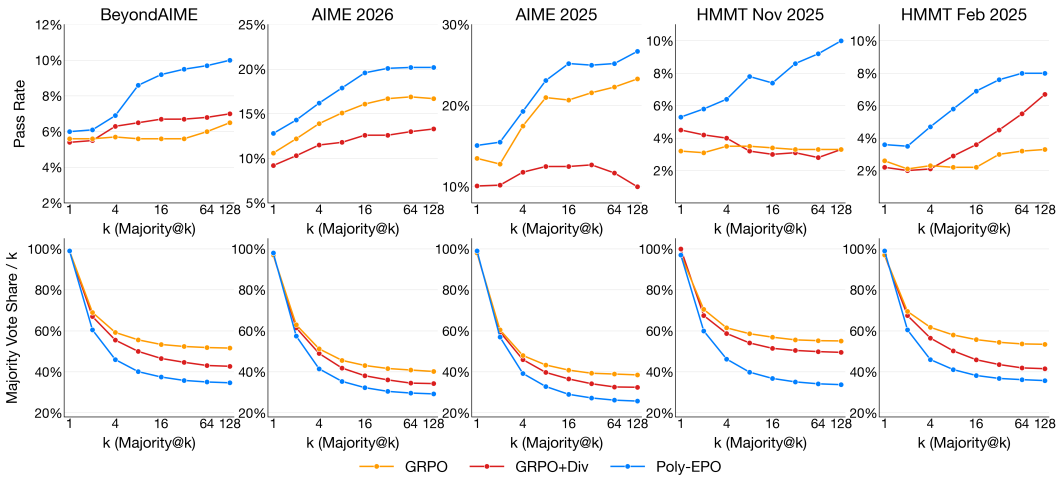


Figure 4: **Majority@k evaluations on Test Sets**. The x-axis denotes the number of sampled generations k used at test time. Top row: pass rate after selecting the answer returned by majority voting over the k samples. Bottom row: majority vote share, i.e., the fraction of the k votes assigned to the winning answer. Higher values indicate stronger agreement among sampled generations, while lower values indicate greater response diversity. All methods are initialized from `Qwen3-4B-Base`. The LM-judge used by POLY-EPO and GRPO+DIV for clustering responses is `Qwen-3-4B-Instruct`.

more naturally corresponds to committing to distinct reasoning strategies, these results indicate that POLY-EPO promotes broader strategic exploration at test time.

We next measure the fraction of prompts in each training batch for which the model samples at least one correct generation. As shown in fig. 2 (right), methods that explicitly encourage exploration achieve higher coverage: both GRPO+DIV and POLY-EPO outperform GRPO for much of training. This suggests that promoting exploration not only preserves strategic diversity, but also improves the probability of uncovering successful solutions during training. These two results, jointly, allow us to see that the polychromic objective does indeed enable the model to synergistically explore and exploit.

Majority Voting Evaluations. Finally, we analyze whether POLY-EPO benefits from increased test-time compute. We evaluate using majority voting (Wang et al., 2022), where k generations are sampled and the most frequent final answer is selected. The results are shown in fig. 4. We first observe that models trained with POLY-EPO retain substantially greater diversity at inference time: across benchmarks, the majority vote share under POLY-EPO is consistently lower than under both GRPO and GRPO+DIV, indicating that probability mass is distributed across a broader set of candidate answers rather than collapsing onto a single response.

Despite this higher entropy over final answers, POLY-EPO attains equal or stronger majority-vote accuracy as k increases, often yielding the largest gains from additional samples. This suggests that the extra diversity induced by the polychromic objective is not merely random variation, but reflects useful diversity in reasoning trajectories. We conjecture that exploration during post-training improves calibration and generalization, allowing additional test-time samples to be converted more effectively into performance gains.

7 CONCLUSION

We presented POLY-EPO, a set RL algorithm that trains policies to explore diverse reasoning strategies while balancing exploration and exploitation through an explicit synergy between reward and diversity. POLY-EPO improves test-time performance, especially when given multiple attempts per problem. Key limitations remain: POLY-EPO relies on LM-judge clustering, which may be susceptible to reward hacking, and the scaling behavior of set RL is not yet well understood.

AI USE STATEMENT

In this work, we used generative AI tools to assist with implementing methods and creating or editing software code, as well as to edit the manuscript to improve readability and clarity. We also used generative AI tools to help identify potentially relevant literature.

We have not used generative AI tools to generate synthetic datasets; develop theoretical models or conceptual frameworks; formulate mathematical claims; provide critical ingredients for proving mathematical claims; assist in writing proofs; propose or refine hypotheses; design or provide feedback on research methodology or experiments; assist with translation; clean or reformat datasets; support qualitative or thematic data analysis; or interpret experimental results.

We have reviewed and tested all AI-assisted work. AI-assisted manuscript edits were reviewed by the authors, and suggestions were incorporated only at the authors' discretion. Literature identified with the assistance of generative AI was independently checked against the original sources for relevance and accuracy. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu, Mingxuan Wang, and Lingpeng Kong. Polaris: A post-training recipe for scaling reinforcement learning on advanced reasoning models, 2025. URL https://hkunlp.github.io/blog/2025/Polaris.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2–3):235–256, May 2002. ISSN 0885-6125. doi: 10.1023/A:1013689704352. URL https://doi.org/10.1023/A:1013689704352.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions, February 2025. URL https://matharena.ai/.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690, March 2024. ISSN 2159-5399. doi: 10.1609/aaai.v38i16.29720. URL http://dx.doi.org/10.1609/aaai.v38i16.29720.

Shalabh Bhatnagar, Richard S. Sutton, Mohammad Ghavamzadeh, and Mark Lee. Natural actor–critic algorithms. *Automatica*, 45(11):2471–2482, 2009. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2009.07.008. URL https://www.sciencedirect.com/science/article/pii/S0005109809003549.

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, 2025. URL https://arxiv.org/abs/2508.10751.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. Reasoning with exploration: An entropy perspective, 2025. URL https://arxiv.org/abs/2506.14758.

Yinlam Chow, Guy Tennenholtz, Izzeddin Gur, Vincent Zhuang, Bo Dai, Sridhar Thiagarajan, Craig Boutilier, Rishabh Agarwal, Aviral Kumar, and Aleksandra Faust. Inference-aware fine-tuning for best-of-n sampling in large language models, 2025. URL https://arxiv.org/abs/2412.15287.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. The entropy mechanism of reinforcement learning for reasoning language models, 2025. URL https://arxiv.org/abs/2505.22617.

Runpeng Dai, Linfeng Song, Haolin Liu, Zhenwen Liang, Dian Yu, Haitao Mi, Zhaopeng Tu, Rui Liu, Tong Zheng, Hongtu Zhu, and Dong Yu. Cde: Curiosity-driven exploration for efficient reinforcement learning in large language models, 2025. URL https://arxiv.org/abs/2509.09675.

Thomas Degris, Martha White, and Richard S. Sutton. Off-policy actor-critic, 2013. URL https://arxiv.org/abs/1205.4839.

Jingtong Gao, Ling Pan, Yejing Wang, Rui Zhong, Chi Lu, Maolin Wang, Qingpeng Cai, Peng Jiang, and Xiangyu Zhao. Navigate the unknown: Enhancing llm reasoning with intrinsic motivation guided exploration, 2026. URL https://arxiv.org/abs/2505.17621.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645 (8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Anthony GX-Chen, Jatin Prakash, Jeff Guo, Rob Fergus, and Rajesh Ranganath. Kl-regularized reinforcement learning is designed to mode collapse, 2025. URL https://arxiv.org/abs/2510.20817.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies, 2017. URL https://arxiv.org/abs/1702.08165.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL https://arxiv.org/abs/1801.01290.

Jubayer Ibn Hamid, Ifdita Hasan Orney, Ellen Xu, Chelsea Finn, and Dorsa Sadigh. Polychromic objectives for reinforcement learning, 2026. URL https://arxiv.org/abs/2509.25424.

Zhiyuan Hu, Yucheng Wang, Yufei He, Jiaying Wu, Yilun Zhao, See-Kiong Ng, Cynthia Breazeal, Anh Tuan Luu, Hae Won Park, and Bryan Hooi. Rewarding the rare: Uniqueness-aware rl for creative problem solving in llms, 2026. URL https://arxiv.org/abs/2601.08763.

Riashat Islam, Zafarali Ahmed, and Doina Precup. Marginalized state distribution entropy regularization in policy optimization, 2019. URL https://arxiv.org/abs/1912.05128.

Sham M Kakade. A natural policy gradient. In T. Dietterich, S. Becker, and Z. Ghahramani (eds.), *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001. URL https://proceedings.neurips.cc/paper_files/paper/2001/file/4b86abe48d358ecf194c56c69108433e-Paper.pdf.

T.L Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985. ISSN 0196-8858. doi: https://doi.org/10.1016/0196-8858(85)90002-8. URL https://www.sciencedirect.com/science/article/pii/0196885885900028.

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, 2022. URL https://arxiv.org/abs/2206.14858.

Tianjian Li, Yiming Zhang, Ping Yu, Swarnadeep Saha, Daniel Khashabi, Jason Weston, Jack Lanchantin, and Tianlu Wang. Jointly reinforcing diversity and quality in language model generations, 2025. URL https://arxiv.org/abs/2509.02534.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1509.02971.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models, 2025a. URL https://arxiv.org/abs/2505.24864.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective, 2025b. URL https://arxiv.org/abs/2503.20783.

MiniMax, Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, Chengjun Xiao, Chengyu Du, Chi Zhang, Chu Qiao, Chunhao Zhang, Chunhui Du, Congchao Guo, Da Chen, Deming Ding, Dianjun Sun, Dong Li, Enwei Jiao, Haigang Zhou, Haimo Zhang, Han Ding, Haohai Sun, Haoyu Feng, Huaiguang Cai, Haichao Zhu, Jian Sun, Jiaqi Zhuang, Jiaren Cai, Jiayuan Song, Jin Zhu, Jingyang Li, Jinhao Tian, Jinli Liu, Junhao Xu, Junjie Yan, Junteng Liu, Junxian He, Kaiyi Feng, Ke Yang, Kecheng Xiao, Le Han, Leyang Wang, Lianfei Yu, Liheng Feng, Lin Li, Lin Zheng, Linge Du, Lingyu Yang, Lunbin Zeng, Minghui Yu, Mingliang Tao, Mingyuan Chi, Mozhi Zhang, Mujie Lin, Nan Hu, Nongyu Di, Peng Gao, Pengfei Li, Pengyu Zhao, Qibing Ren, Qidi Xu, Qile Li, Qin Wang, Rong Tian, Ruitao Leng, Shaoxiang Chen, Shaoyu Chen, Shengmin Shi, Shitong Weng, Shuchang Guan, Shuqi Yu, Sichen Li, Songquan Zhu, Tengfei Li, Tianchi Cai, Tianrun Liang, Weiyu Cheng, Weize Kong, Wenkai Li, Xiancai Chen, Xiangjun Song, Xiao Luo, Xiao Su, Xiaobo Li, Xiaodong Han, Xinzhu Hou, Xuan Lu, Xun Zou, Xuyang Shen, Yan Gong, Yan Ma, Yang Wang, Yiqi Shi, Yiran Zhong, Yonghong Duan, Yongxiang Fu, Yongyi Hu, Yu Gao, Yuanxiang Fan, Yufeng Yang, Yuhao Li, Yulin Hu, Yunan Huang, Yunji Li, Yunzhi Xu, Yuxin Mao, Yuxuan Shi, Yuze Wenren, Zehan Li, Zelin Li, Zhanxu Tian, Zhengmao Zhu, Zhenhua Fan, Zhenzhen Wu, Zhichao Xu, Zhihang Yu, Zhiheng Lyu, Zhuo Jiang, Zibo Gao, Zijia Wu, Zijian Song, and Zijun Sun. Minimax-m1: Scaling test-time compute efficiently with lightning attention, 2025. URL https://arxiv.org/abs/2506.13585.

Cursor Research, :, Aaron Chan, Ahmed Shalaby, Alexander Wettig, Aman Sanger, Andrew Zhai, Anurag Ajay, Ashvin Nair, Charlie Snell, Chen Lu, Chen Shen, Emily Jia, Federico Cassano, Hanpeng Liu, Haoyu Chen, Henry Wildermuth, Jacob Jackson, Janet Li, Jediah Katz, Jiajun Yao, Joey Hejna, Josh Warner, Julius Vering, Kevin Frans, Lee Danilek, Less Wright, Lujing Cen, Luke Melas-Kyriazi, Michael Truell, Michiel de Jong, Naman Jain, Nate Schmidt, Nathan Wang, Niklas Muennighoff, Oleg Rybkin, Paul Loh, Phillip Kravtsov, Rishabh Yadav, Sahil Shah, Sam Kottler, Alexander M Rush, Shengtong Zhang, Shomil Jain, Sriram Sankar, Stefan Heule, Stuart H. Sul, Sualeh Asif, Victor Rong, Wanqi Zhu, William Lin, Yuchen Wu, Yuri Volkov, Yury

Zemlyanskiy, Zack Holbrook, and Zhiyuan Zhang. Composer 2 technical report, 2026. URL https://arxiv.org/abs/2603.24477.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2017a. URL https://arxiv.org/abs/1502.05477.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017b. URL https://arxiv.org/abs/1707.06347.

ByteDance Seed, :, Jiaze Chen, Tiantian Fan, Xin Liu, Lingjun Liu, Zhiqi Lin, Mingxuan Wang, Chengyi Wang, Xiangpeng Wei, Wenyuan Xu, Yufeng Yuan, Yu Yue, Lin Yan, Qiying Yu, Xiaochen Zuo, Chi Zhang, Ruofei Zhu, Zhecheng An, Zhihao Bai, Yu Bao, Xingyan Bin, Jiangjie Chen, Feng Chen, Hongmin Chen, Riwei Chen, Liangqiang Chen, Zixin Chen, Jinsong Chen, Siyan Chen, Kaiyuan Chen, Zhi Chen, Jin Chen, Jiecao Chen, Jinxin Chi, Weinan Dai, Ning Dai, Jiahui Dai, Shihan Dou, Yantao Du, Zhengyin Du, Jianhui Duan, Chen Dun, Ting-Han Fan, Jiazhan Feng, Junda Feng, Ziyuan Feng, Yuwei Fu, Wenqi Fu, Hanjie Fu, Hao Ge, Hongyi Guo, Mingji Han, Li Han, Wenhao Hao, Xintong Hao, Qianyu He, Jerry He, Feng He, Wen Heng, Zehua Hong, Qi Hou, Liang Hu, Shengding Hu, Nan Hu, Kai Hua, Qi Huang, Ziyue Huang, Hongzhi Huang, Zihao Huang, Ting Huang, Wenhao Huang, Wei Jia, Bin Jia, Xiaoying Jia, Yuhua Jiang, Haobin Jiang, Ziheng Jiang, Kaihua Jiang, Chengquan Jiang, Jianpeng Jiao, Xiaoran Jin, Xing Jin, Xunhao Lai, Zheng Li, Xiang Li, Liyi Li, Hongkai Li, Zheng Li, Shengxian Wan, Ya Wang, Yunshui Li, Chenggang Li, Niuniu Li, Siyu Li, Xi Li, Xiao Li, Aoyan Li, Yuntao Li, Nianning Liang, Xinnian Liang, Haibin Lin, Weijian Lin, Ye Lin, Zhicheng Liu, Guanlin Liu, Guanlin Liu, Chenxiao Liu, Yan Liu, Gaohong Liu, Juncai Liu, Chundian Liu, Deyi Liu, Kaibo Liu, Siyao Liu, Qi Liu, Yongfei Liu, Kang Liu, Gan Liu, Boyi Liu, Rui Long, Weiqiang Lou, Chenwei Lou, Xiang Luo, Yao Luo, Caiping Lv, Heyang Lv, Bole Ma, Qianli Ma, Hongzhi Ma, Yiyuan Ma, Jin Ma, Wenchang Ma, Tingting Ma, Chen Mao, Qiyang Min, Zhe Nan, Guanghan Ning, Jinxiang Ou, Haojie Pan, Renming Pang, Yanghua Peng, Tao Peng, Lihua Qian, Lihua Qian, Mu Qiao, Meng Qu, Cheng Ren, Hongbin Ren, Yong Shan, Wei Shen, Ke Shen, Kai Shen, Guangming Sheng, Jinlong Shi, Wenlei Shi, Guang Shi, Shuai Shuai Cao, Yuxin Song, Zuquan Song, Jing Su, Yifan Sun, Tao Sun, Zewei Sun, Borui Wan, Zihan Wang, Xiaohui Wang, Xi Wang, Shuguang Wang, Jun Wang, Qinlong Wang, Chenyuan Wang, Shuai Wang, Zihan Wang, Changbao Wang, Jiaqiang Wang, Shihang Wang, Xuwu Wang, Zaiyuan Wang, Yuxuan Wang, Wenqi Wang, Taiqing Wang, Chengzhi Wei, Houmin Wei, Ziyun Wei, Shufa Wei, Zheng Wu, Yonghui Wu, Yangjun Wu, Bohong Wu, Shuang Wu, Jingqiao Wu, Ning Wu, Shuangzhi Wu, Jianmin Wu, Chenguang Xi, Fan Xia, Yuqiao Xian, Liang Xiang, Boren Xiang, Bowen Xiao, Zhen Xiao, Xia Xiao, Yongsheng Xiao, Chao Xin, Shulin Xin, Yuwen Xiong, Jingjing Xu, Ziwen Xu, Chenyin Xu, Jiayi Xu, Yifan Xu, Wei Xu, Yufei Xu, Shikun Xu, Shipeng Yan, Shen Yan, Qingping Yang, Xi Yang, Tianhao Yang, Yuehang Yang, Yuan Yang, Ximing Yang, Zeyu Yang, Guang Yang, Yifan Yang, Xuesong Yao, Bairen Yi, Fan Yin, Jianian Yin, Ziqiang Ying, Xiangyu Yu, Hongli Yu, Song Yu, Menghan Yu, Huan Yu, Siyu Yuan, Jun Yuan, Yutao Zeng, Tianyang Zhan, Zheng Zhang, Yun Zhang, Mofan Zhang, Wang Zhang, Ru Zhang, Zhi Zhang, Tianqi Zhang, Xinyi Zhang, Zhexi Zhang, Sijun Zhang, Wenqiang Zhang, Xiangxiang Zhang, Yongtao Zhang, Yuyu Zhang, Ge Zhang, He Zhang, Yue Zhang, Renjie Zheng, Ningxin Zheng, Zhuolin Zheng, Yaowei Zheng, Chen Zheng, Xiaoyun Zhi, Wanjun Zhong, Cheng Zhong, Zheng Zhong, Baoquan Zhong, Xun Zhou, Na Zhou, Huan Zhou, Hang Zhu, Defa Zhu, Wenjia Zhu, and Lei Zuo. Seed1.5-thinking: Advancing superb reasoning models with reinforcement learning, 2025. URL https://arxiv.org/abs/2504.13914.

Younggyo Seo, Lili Chen, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. State entropy maximization with random encoders for efficient exploration, 2021. URL https://arxiv.org/abs/2102.09430.

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Yuda Song, Julia Kempe, and Remi Munos. Outcome-based exploration for llm reasoning, 2025. URL https://arxiv.org/abs/2509.06941.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'99, pp. 1057–1063, Cambridge, MA, USA, 1999. MIT Press.

Fahim Tajwar, Guanning Zeng, Yueer Zhou, Yuda Song, Daman Arora, Yiding Jiang, Jeff Schneider, Ruslan Salakhutdinov, Haiwen Feng, and Andrea Zanette. Maximum likelihood reinforcement learning, 2026. URL https://arxiv.org/abs/2602.02710.

Yunhao Tang, Kunhao Zheng, Gabriel Synnaeve, and Rémi Munos. Optimizing language models for inference time objectives using reinforcement learning, 2025. URL https://arxiv.org/abs/2503.19595.

Jens Tuyls, Dylan J. Foster, Akshay Krishnamurthy, and Jordan T. Ash. Representation-based exploration for language models: From test-time to post-training, 2025. URL https://arxiv.org/abs/2510.11686.

Christian Walder and Deep Karkhanis. Pass@k policy optimization: Solving harder reinforcement learning problems, 2025. URL https://arxiv.org/abs/2505.15201.

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay, 2017. URL https://arxiv.org/abs/1611.01224.

Fang Wu, Weihao Xuan, Ximing Lu, Mingjie Liu, Yi Dong, Zaid Harchaoui, and Yejin Choi. The invisible leash: Why rlvr may or may not escape its origin, 2026. URL https://arxiv.org/abs/2507.14843.

Jian Yao, Ran Cheng, Xingyu Wu, Jibin Wu, and Kay Chen Tan. Diversity-aware policy optimization for large language model reasoning, 2025. URL https://arxiv.org/abs/2505.23433.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL https://arxiv.org/abs/2305.10601.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan 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, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model?, 2025. URL https://arxiv.org/abs/2504.13837.

Xuan Zhang, Ruixiao Li, Zhijian Zhou, Long Li, Yulei Qin, Ke Li, Xing Sun, Xiaoyu Tan, Chao Qu, and Yuan Qi. Count counts: Motivating exploration in llm reasoning with count-based intrinsic rewards, 2025. URL https://arxiv.org/abs/2510.16614.

Rosie Zhao, Alexandru Meterez, Sham Kakade, Cengiz Pehlevan, Samy Jelassi, and Eran Malach. Echo chamber: Rl post-training amplifies behaviors learned in pretraining, 2025. URL https://arxiv.org/abs/2504.07912.

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

Tianyu Zheng, Tianshun Xing, Qingshui Gu, Taoran Liang, Xingwei Qu, Xin Zhou, Yizhi Li, Zhoufutu Wen, Chenghua Lin, Wenhao Huang, Qian Liu, Ge Zhang, and Zejun Ma. First return, entropy-eliciting explore, 2025b. URL https://arxiv.org/abs/2507.07017.

King Zhu, Hanhao Li, Siwei Wu, Tianshun Xing, Dehua Ma, Xiangru Tang, Minghao Liu, Jian Yang, Jiaheng Liu, Yuchen Eleanor Jiang, Changwang Zhang, Chenghua Lin, Jun Wang, Ge Zhang, and Wangchunshu Zhou. Scaling test-time compute for llm agents, 2025. URL https://arxiv.org/abs/2506.12928.

A Related Work

Policy Gradient Methods. Policy gradient methods (Sutton et al., 1999; Sutton & Barto, 2018; Kakade, 2001) are a foundational class of reinforcement learning algorithms that directly optimize a policy to maximize expected rewards. Recent advances in reducing variance and improving sample efficiency (Bhatnagar et al., 2009; Degris et al., 2013; Lillicrap et al., 2016; Wang et al., 2017; Schulman et al., 2017a;b) have made them widely adopted for language model (LM) fine-tuning. In the LM setting, these methods typically omit a learned critic in favor of empirical advantage estimates (Guo et al., 2025; Yu et al., 2025; Zheng et al., 2025a; Liu et al., 2025b; MiniMax et al., 2025). Concurrently, Hamid et al. (2026) introduced the set reinforcement learning framework, which optimizes an agent with respect to a set-level learning signal. Our work bridges and complements both lines of research: the general set RL recipe we propose can seamlessly adapt standard RL algorithms by replacing the advantage with the marginal set advantage to implement set RL at scale.

Furthermore, as test-time compute has become a standard paradigm for performance gains, recent research has explored training objectives aligned with inference-time metrics (Tang et al., 2025; Chow et al., 2025; Chen et al., 2025; Walder & Karkhanis, 2025). While some of these frameworks also optimize over a number of generations for the specific instance of pass@k training (Walder & Karkhanis, 2025; Tang et al., 2025), they rely on a leave-one-out baseline. This causes the objective to collapse to individualized credit assignment that set RL, by definition, does not allow (Hamid et al., 2026). Crucially, if one were to strictly optimize the pass@k objective using the set RL framework, generations with incorrect final answers would always receive a negative advantage, as we showed in §D.1. On the other hand, Chen et al. (2025) propose an algorithm for pass@k training that also computes a U-statistic estimator. However, their advantage is not the same as the set advantage since they divide by the standard deviation in the set scores, which induces a significant deviation from our general set RL recipe. Additionally, they analytically compute the advantage of each generation under the pass@k training framework, which may not be possible for arbitrary objectives.

Exploration and Diversity. Balancing exploration and exploitation remains a critical challenge during LM post-training. While some works argue that RL fine-tuning acts primarily as an entropy-reducing mechanism that does not surface new capabilities (Yue et al., 2025; Cui et al., 2025; Wu et al., 2026; Zhao et al., 2025), others have demonstrated that RL fine-tuning can successfully expand reasoning capabilities to held-out tasks (Liu et al., 2025a; Research et al., 2026). Our work is simply motivated by the principle that effective exploration amplifies learning from experience. Several modifications to the RL pipeline have been proposed to achieve such exploration. Standard approaches like entropy bonuses (Haarnoja et al., 2017; 2018; Schulman et al., 2017b; Seo et al., 2021; Islam et al., 2019; Zheng et al., 2025b) encourage local exploration, though their efficacy in inducing trajectory-level exploration over high-level strategies during LM post-training remains unclear. Similarly, UCB and count-based bonuses (Song et al., 2025; Zhang et al., 2025; Tuyls et al., 2025) can incentivize novel behaviors, but they rely on weighting hyperparameters rather than explicitly driving a synergy between exploration and exploitation. Other methods implicitly induce exploration through curiosity-driven techniques (Dai et al., 2025; Gao et al., 2026) or novel objectives

independent of diversity measures (Tajwar et al., 2026; Chen et al., 2025; Walder & Karkhanis, 2025; Cui et al., 2025; Yu et al., 2025; GX-Chen et al., 2025).

More closely related to our approach are works that promote exploration via objectives targeting the semantic diversity of generations (Li et al., 2025; Yao et al., 2025; Hu et al., 2026). However, while these distribute probability mass across multiple *correct* generations, they do not explicitly encourage optimistic exploration by providing a positive learning signal to exploratory trajectories with incorrect final answers. On the other hand, Song et al. (2025); Tuyls et al. (2025) use the standard RL framework with an exploration bonus that can encourage such optimistic exploration but the balance between exploration and exploitation is dictated by a weighting hyperparameter and the objective does not lead to an advantage function that explicitly encourages a synergy between these goals. In contrast, the set reinforcement learning framework with polychromic objectives (Hamid et al., 2026) achieves both optimistic exploration and synergistic exploitation. Yet, their algorithm requires a learned critic and relies on computationally heavy vine sampling, making it difficult to scale for LM post-training. By resolving these sample and computational complexities, our approach allows LMs to leverage the full benefits of polychromic objectives.

B Implementation Details

In this section, we discuss the implementation details of Poly-EPO. Our implementation builds upon the codebase from Tajwar et al. (2026), which is in turn based on the Verl framework (Sheng et al., 2024). Recall that our general recipe for set RL simply adapts the advantage function used in a standard RL algorithm. As our backbone, we use the following standard RL objective (Schulman et al., 2017b; Shao et al., 2024; Guo et al., 2025; Liu et al., 2025b):

$$J(\theta) = \mathbb{E}_{x\sim\mathcal{D}, y_{1:N}\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\left[\frac{1}{N}\sum_{i=1}^{N}\frac{1}{T_i}\sum_{t=1}^{|y_i|}\min\left[\omega_{i,t}\hat{A}_{i,t}, \text{clip}\left(\omega_{i,t}, 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}}\right)\hat{A}_{i,t}\right]\right],$$

where $\omega_{i,t} = \frac{\pi_\theta(y_{i,t}|x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t}|x, y_{i,<t})}$.

For the GRPO baseline, $T_i = |y_i|$ to ensure the objective is averaged over the length of each individual generation. The advantage estimator is defined as:

$$\hat{A}_{i,t} = r(x, y_i) - \text{mean}(\{r(x, y_1), \ldots, r(x, y_N)\}).$$

Note that we omit the standard deviation normalization term originally used in Shao et al. (2024), as we found this choice led to better empirical performance—a modification consistent with recent findings in Dr.GRPO (Liu et al., 2025b) and other works, such as Research et al. (2026).

For the GRPO+DIV baseline, the advantage estimator incorporates a diversity reward and is computed as:

$$\hat{A}_{i,t} = r(x, y_i) + \lambda d(x, y_i) - \text{mean}\left(\{r(x, y_j) + \lambda d(x, y_j)\}_{j=1}^{N}\right).$$

Finally, for Poly-EPO, we do not normalize by the individual generation lengths. Instead, following the Dr.GRPO implementation in Verl, we set $T_i = T_{\max}$, where $T_{\max}$ is the maximum response length. The advantage is computed as:

$$\hat{A}_{i,t} = \widehat{A^{\sharp}_{\text{marg}}}(x, y_i)$$

as detailed in section 3. Note that although we proved, in Proposition 3.1, that our estimator is unbiased only after scaling by a constant factor that can be absorbed into the learning rate, we do not introduce this factor in implementation and do not consider this to be an additional hyperparameter that needs tuning.

B.1 LM-Judge and Diversity Function

For a given prompt x, we sample N candidate responses $y_1, \ldots, y_N \overset{\text{i.i.d}}{\sim} \pi_\theta(\cdot \mid x)$ and use a language model as judge to cluster these generations. The primary objective of this clustering phase is to group

responses strictly according to the underlying mathematical reasoning strategies employed, rather than surface-level textual similarities. To systematically guide the judge, we provide a comprehensive system prompt comprising task specifications, rigorous clustering criteria, formatting constraints, and curated in-context examples. The core directives are defined as follows:

1. **Strategy-centric clustering:** The judge must categorize responses based on both macro-strategy (the overarching conceptual framework) and micro-strategy (the specific techniques used at key intermediate steps), deliberately disregarding stylistic variations or tone. Crucially, this clustering process is completely independent of the final derived answer. If two response trajectories exhibit identical logical reasoning but diverge due to a minor arithmetic error, they must be assigned to the same cluster. The provided few-shot examples explicitly demonstrate this invariant.
2. **Isolation of degenerate responses (Cluster 100):** We reserve a dedicated, out-of-bounds index ("cluster 100") for degenerate generations. Empirically, we observe that during optimization, the policy network may sometimes discover reward-hacking behaviors, such as directly predicting the final answer without executing intermediate logical steps or generating unintelligible text. The judge is explicitly instructed to isolate all such spurious generations into this dedicated "cluster 100" bucket. Because the number of generations per prompt, N, is strictly less than 100, this fixed numeric identifier prevents index collisions and facilitates the systematic tracking of undesirable model behaviors throughout training.

The full prompt supplied to the LM-judge consists of: (i) a static instruction block detailing the task, rules, and few-shot examples, and (ii) an instance-specific suffix containing the problem context alongside the sampled responses.

Instruction Block for LM-Based Clustering of Reasoning Strategies

```
Your ONLY task is to cluster the {n_responses} responses into
    ↪ buckets based on their reasoning algorithm, including both
    ↪ the overall strategy and the methods used at key intermediate
    ↪  steps.

**INPUT FORMAT:** You will receive:
1) A "Context" describing the task.
2) A numbered list of Responses from 1 to {n_responses}. Each
    ↪ response contains a reasoning process and final answer.
Note: Responses may or may not explicitly state their strategy; you
    ↪  must infer the strategy by analyzing the mathematical steps
    ↪ taken.

**CLUSTERING CRITERIA:**
(1) Macro-strategy: The overall conceptual framework (e.g.,
    ↪ recursion vs infinite series; prime factorization vs gcd-
    ↪ based formula).
(2) Micro-strategy: The specific method used to resolve key
    ↪ intermediate steps. Examples include: how absolute values are
    ↪  removed (+- case split vs squaring), how intervals are
    ↪ partitioned, or how a basis is chosen.

**CLUSTERING RULES:** - Cluster strictly based on logic and
    ↪ approach. NOT on wording, tone, formatting, or final answer.
- Two responses share a cluster_id IF AND ONLY IF they use the same
    ↪  macro-strategy AND the same micro-strategy at every key step.
    ↪
- Arithmetic errors do NOT create new clusters if the underlying
    ↪ logic is identical.
- **SPECIAL CLUSTER 100:** You MUST assign `cluster_id: 100` to any
    ↪  response that is:
   * Gibberish (random characters, nonsense strings).
   * Irrelevant to the math problem (off-topic text).
```

```
   * Non-mathematical reasoning (e.g., writing code to solve it
      ↪ instead of math, or making a random guess at the final
      ↪ answer without logical steps).

**OUTPUT RULES (STRICT):** 1. Respond ONLY with a JSON object. No
   ↪ text outside the JSON.
2. The JSON must contain exactly {n_responses} keys: "1", "2", ...,
   ↪  "{n_responses}".
3. The value for each key must be:
   "chain_of_thought": "Macro: [short description]. Micro: [short
      ↪ description]."
   "cluster_id": integer.
4. chain_of_thought must be concise and avoid repeating the actual
   ↪ calculations.

**Few-Shot Example 1:**

**Context:**
What is the smallest value of x such that |5x - 1| = |3x + 2|?

**Responses:**
1. We can split this into two cases: 5x - 1 = 3x + 2 or 5x - 1 =
   ↪ -(3x + 2). Solving the first gives 2x = 3 so x = 1.5. The
   ↪ second gives 8x = -1 so x = -0.125.
2. The expression 5x-1 changes sign at 1/5, and 3x+2 changes at
   ↪ -2/3. For x < -2/3, we have -(5x-1) = -(3x+2). For -2/3 < x <
   ↪  1/5, we have -(5x-1) = 3x+2. Solve -(5x - 1) = 3x + 2 for
   ↪ the range, yielding x = -0.125.
3. Using the property that |a|=|b| implies a=b or a=-b, we get 5x-1
   ↪  = 3x+2 (x=1.5) and 5x-1 = -3x-2 (x=-0.125). So the answer is
   ↪  x = -0.125
4. To get rid of the absolute values, square both sides: (5x - 1)^2
   ↪  = (3x + 2)^2. This expands to 25x^2 - 10x + 1 = 9x^2 + 12x +
   ↪  4. Solve 16x^2 - 22x - 3 = 0. So, x = -1/8, 3/2. Final
   ↪ answer is x = -1/8
5. Either 5x - 1 = 3x + 2 or 5x - 1 = -3x - 2. This leads to x =
   ↪ 3/2 and x = 0. So, final answer is x = 0.
6. I think the answer is probably 0 or maybe 1.5.
7. asdf qwer zxcv 9999 ---- ??? Let's write Python to check each x
   ↪ from -10 to 10: `if abs(5*x-1) == abs(3*x+2): print(x)`. The
   ↪ answer is -0.125.

**Expected Output:**
{{
"1": {{"chain_of_thought": "Macro: Algebraic casework. Micro:
   ↪ Direct +- case split to remove absolute values.", "cluster_id
   ↪ ": 1}},
"2": {{"chain_of_thought": "Macro: Interval analysis. Micro:
   ↪ Testing expression sign changes across number line partitions
   ↪ .", "cluster_id": 2}},
"3": {{"chain_of_thought": "Macro: Algebraic casework. Micro:
   ↪ Direct +- case split to remove absolute values.", "cluster_id
   ↪ ": 1}},
"4": {{"chain_of_thought": "Macro: Algebraic transformation. Micro:
   ↪  Squaring both sides to create and solve a quadratic equation
   ↪ .", "cluster_id": 3}},
"5": {{"chain_of_thought": "Macro: Algebraic casework. Micro:
   ↪ Direct +- case split to remove absolute values (contains
   ↪ arithmetic error).", "cluster_id": 1}},
"6": {{"chain_of_thought": "Macro: Non-mathematical. Micro: Random
   ↪ guessing without any logical derivation.", "cluster_id":
   ↪ 100}},
```

```
"7": {{"chain_of_thought": "Macro: Gibberish/Non-mathematical.
    ↪ Micro: Contains random non-English text, nonsense strings,
    ↪ and code-based iteration.", "cluster_id": 100}}
}}

**Few-Shot Example 2:**

**Context:**
What is the least common multiple of 72 and 96?

**Responses:**
1. 72 = 2^3 * 3^2. 96 = 2^5 * 3^1. To find the LCM, we take the
    ↪ highest power of each prime factor present: 2^5 * 3^2 = 32 *
    ↪ 9 = 288.
2. Prime factors of 72: 2, 2, 2, 3, 3. Prime factors of 96: 2, 2, 2,
    ↪  2, 2, 3. The union of these sets is five 2s and two 3s.
    ↪ Total: 276.
3. First find the GCD using the Euclidean algorithm: 96 = 72(1) +
    ↪ 24; 72 = 24(3) + 0. GCD is 24. LCM is (72 * 96) / 24.
4. 72 = 8*9, 96 = 8*12. The answer is 288. The answer is 288. The
    ↪ answer is 288. The answer is 288.

**Expected Output:**
{{
"1": {{"chain_of_thought": "Macro: Prime factorization analysis.
    ↪ Micro: LCM via maximum exponents of prime factors.", "
    ↪ cluster_id": 1}},
"2": {{"chain_of_thought": "Macro: Prime factorization analysis.
    ↪ Micro: LCM via maximum exponents of prime factors (contains
    ↪ arithmetic error).", "cluster_id": 1}},
"3": {{"chain_of_thought": "Macro: Product-GCD relationship. Micro:
    ↪  GCD calculation via Euclidean algorithm followed by the LCM
    ↪ formula.", "cluster_id": 2}},
"4": {{"chain_of_thought": "Macro: Excessive repetition. Micro:
    ↪ Response loops the final answer multiple times at the end.",
    ↪ "cluster_id": 100}}
}}
```

Instance-Specific Suffix. For a given problem instance, we dynamically construct the prompt suffix by populating the template below with the actual mathematical context and the sampled responses:

Instance-Specific Prompt Suffix

```
**Context:**
<problem description>

**Responses:**
1. <response 1>
2. <response 2>
...
{n_responses}. <response {n_responses}>
```

During the evaluation phase, the static instruction block and the dynamically instantiated suffix are concatenated into a unified user message and passed to the LM-judge for inference. The LM-judge is used to cluster all N responses, $y_1, \cdots, y_N$, sampled conditioned on a prompt. This gives us a cluster assignment, $\mathcal{C}(y)$, for each generation y. In order to prevent the model from artificially inducing diversity by generating random guesses at the final answer or by generating unintelligible text, we remove the cluster assignments for all such generations when computing the diversity of any set. In

Algorithm 2 Compute Polychromic Objective

Require: Prompt x, generations $\{y_{1:N}\}$, sets $\{G_{1:K}\}$, judge $\mathcal{M}$, reward r
1: $\{\mathcal{C}(y_1), \ldots, \mathcal{C}(y_N)\} \leftarrow \mathcal{M}(x, \{y_{1:N}\})$ *// Step 1: Semantic Clustering using LM-Judge*
2: **for** $l = 1$ to K **do**
3: $\quad U_l \leftarrow \{\mathcal{C}(y) \mid y \in G_l, \mathcal{C}(y) \neq$ Garbage Cluster/Cluster "100"$\}$
4: $\quad \bar{r}_l \leftarrow \frac{1}{n} \sum_{y \in G_l} r(x, y)$
5: $\quad f_{\text{poly}}(x, G_l) \leftarrow \bar{r}_l \cdot \frac{|U_l|}{n}$ {Score is 0 if only 1 cluster is present}
6: **end for**
7: **return** $\{f_{\text{poly}}(x, G_l)\}_{l=1}^{K}$

other words, when computing the diversity of any set using eq. (7), we remove any cluster assignment to "cluster 100" from being in the set in the numerator.

The pseudocode for computing the score of sets under the polychromic objective is shown in algorithm 2.

B.2 Hyperparameters

The training hyperparameters for the experiments on mathematical reasoning (§6) are provided in table 1. For GRPO+DIV, we tuned the hyperparameter λ in eq. (9) and settled on using $\lambda = 0.25$; for larger values, the run is often unstable and collapsed in performance.

Parameter	Value
Base model	`Qwen-3-4b-base`
Generations per prompt	8
Set size (for set RL)	4
Number of sets (for set RL)	70
Max prompt length	1024
Learning rate	1×10^{-6}
KL coefficient	0.0
Clip ratio ϵ_{low}	0.20
Clip ratio ϵ_{high}	0.28
Entropy coefficient	0.0
Rollout temperature	1.0
Prompts per batch	128
Prompts per minibatch	64
Max response length	4096
Training steps	850
Device	4 × NVIDIA H200

Table 1: Training hyperparameters.

C Sequential Formulation of Set RL

In the idealized sequential formulation, at each visited state s, the policy samples a set of n actions independently from $\pi_\theta(\cdot \mid s)$ where $n > 1$ is a fixed integer. Given $a_1, \ldots, a_n \overset{\text{i.i.d.}}{\sim} \pi_\theta(\cdot \mid s)$, a set-level reward $f(s, a_{1:n})$ is assigned to the entire set. Repeating this procedure recursively at every timestep generates a *tree* of states visited by our policy. At depth t, the tree contains n^t states, which we denote by $s_t^{(1)}, \ldots, s_t^{(n^t)}$. For each such state $s_t^{(i)}$, let $(a_t)_{1:n}^{(i)}$ denote the n-tuple of actions sampled from $\pi_\theta(\cdot \mid s_t^{(i)})$. The goal is to solve the following optimization problem defined using set

value function:

$$\max_{\theta} V^{\sharp}_{\pi_\theta}(s; f) := \max_{\theta} \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty}\sum_{i=1}^{n^t} \gamma^t f\left(s_t^{(i)}, (a_t)_{1:n}^{(i)}\right) \mid s_0 = s\right]. \tag{10}$$

In particular, we require that f is an objective such that $\mathbb{E}_{a_{1:n}\sim\pi_\theta(\cdot|s)}\left[f(s, a_{1:n})\right]$ cannot be written as $\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[g(s, a)\right]$ where g is independent of θ for all policy parameters $\theta \in \Theta$. In other words, we cannot collapse the set RL problem, at any state, to a standard RL one. Intuitively, set RL, in the sequential formulation, encourages the policy to maximize the expected score of the *tree* induced by the policy. In contrast, standard RL encourages the policy to maximize the expected reward of the *single trajectory*. The sequential formulation is computationally impractical in almost all settings because the number of sampled states grows exponentially with depth.

D Additional Analysis

D.1 Analysis of the General Recipe

In this section, we ask the following question: *why can standard RL algorithms be used to optimize set RL by simply replacing the advantage to be the marginal set advantage as shown in eq.* (4)*?* First, we recall a result analyzing logit shifts in standard reinforcement learning. Suppose, we parametrize our policy (in this case, the LM) as a softmax distribution. Then, Cui et al. (2025) showed that, after one gradient update with learning rate α, the shift in probability mass can be expressed as:

$$\log \pi_\theta^{k+1}(y \mid x) - \log \pi_\theta^{k}(y \mid x) = \alpha \pi_\theta^k(y \mid x) A(x, y; \pi_\theta^k) \tag{11}$$

where $A(x, y; \pi_\theta^k)$ is the advantage of the generation with respect to the policy π_θ^k under the standard RL algorithm.

Now, we look at logit shifts in the set reinforcement learning framework. The result follows from the observation in Hamid et al. (2026) on the *implicit individual credit* that a generation receives in the set RL framework. In adapted form, the logit shift looks as follows:

Lemma D.1. *In set reinforcement learning, assuming the objective function f is symmetric in its arguments and the learning rate is α, the shift in probability mass on a fixed generation y after one step gradient update from π_θ^k to π_θ^{k+1} can be written as:*

$$\log \pi_\theta^{k+1}(y \mid x) - \log \pi_\theta^{k}(y \mid x) = \alpha \pi_\theta^k(y \mid x) \left[\mathbb{E}_{y_{2:n}\sim\pi_\theta^k(\cdot|x)}\left[f(x, y, y_{2:n})\right] - \mathbb{E}_{y_{1:n}\sim\pi_\theta^k(\cdot|x)}\left[f(x, y_{1:n})\right]\right]. \tag{12}$$

The proof can be found in §E. This result tells us that, after one update using set RL, the log likelihood of a generation being sampled changes according to the contribution of the generation to the score of sets that contain it. In particular, the marginal set advantage builds the bridge between these two frameworks. Concretely, comparing eq. (11) with eq. (12), we can simply define the marginal set advantage of a generation as

$$A^{\sharp}_{\text{marg}}(x, y; f, \pi_\theta^k) := \mathbb{E}_{y_{2:n}\sim\pi_\theta^k(\cdot|x)}\left[f(x, y, y_{2:n})\right] - \mathbb{E}_{y_{1:n}\sim\pi_\theta^k(\cdot|x)}\left[f(x, y_{1:n})\right]. \tag{13}$$

This enables us to use any standard RL algorithm to update our policy using set RL by simply replacing the advantage function with our marginal set advantage, since in doing so, the logit shift is equivalent.

This derivation points us to a fundamental difference between standard RL and set RL. In the standard RL setting, the logit shift for a particular generation y is simply governed by $A(x, y)$ which is a measure of how much better the generation is compared to the baseline in terms of returns. In contrast, in the set RL framework, the logit shift for generation y is governed by the contribution of y to sets' scores under the objective f, i.e., the marginal set advantage of y.

Apart from being useful for deriving our algorithm, this also helps us understand what a set objective function explicitly encourages. As an example, consider the pass@n objective $f_{\text{pass@n}}(x, y_{1:n}) = \max_{i\in\{1,\cdots,n\}} r(x, y_i)$ (Tang et al., 2025; Walder & Karkhanis, 2025; Chen et al., 2025). In

the set RL framework, the marginal set advantage of any generation that achieves reward 0 is $A^{\sharp}_{\text{marg}}(x, y_{\text{incorrect}}) = (1-p)^n - (1-p)^{n-1}$, where p is the probability of sampling a generation that achieves reward 1. As such, the marginal set advantage of an incorrect generation is non-positive. On the other hand, the marginal set advantage of a correct generation is $A^{\sharp}_{\text{marg}}(x, y_{\text{correct}}) = (1-p)^n$, which is always non-negative.

Observe that the marginal set advantage of a generation gives us insight into what the algorithm encourages precisely because of eq. (12). We will now use this as our apparatus to understand what different objectives optimize for in the next section.

D.2 Advantage Comparison with Reward Shaping in Standard RL

Advantage under Reward Shaping in Standard RL. Next, consider a standard RL approach with an exploration bonus. In this framework, the objective is

$$\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[\tilde{r}(x,y)\right] = \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[r(x,y) + \lambda \cdot d(x,y)\right]$$

where $d(x, y)$ measures the diversity or exploration contribution of generation y. We assume $d(x, y) \in [0, 1]$, although the analysis does not depend on the specific implementation of d. Under this objective, a generation receives increased probability if its standard RL advantage is positive (Cui et al., 2025). The advantage can be written as

$$A(x, y; \tilde{r}, \theta) = r(x, y) - \mathbb{E}_Y[r(x, Y)] + \lambda\big(d(x, y) - \mathbb{E}_Y[d(x, Y)]\big). \tag{14}$$

We now highlight several fundamental differences between the marginal set advantage under Poly-EPO (eq. (8)) and the advantage induced by reward-shaped standard RL (eq. (14)). Let p denote the probability that the model samples a correct generation for prompt x.

- Consider a generation y with $r(x, y) = 0$ (an incorrect response) but high diversity contribution, i.e., $d(x, y)$ is large. Under the reward-shaped RL objective, y receives positive advantage only if $d(x, y) \geq \frac{p}{\lambda} + \mathbb{E}_Y[d(x, Y)]$. Thus, the exploration coefficient λ critically determines whether exploratory but incorrect generations are encouraged. Moreover, as model accuracy improves (i.e., p increases), this threshold becomes harder to satisfy (for e.g, with $\lambda = 0.5$, if $p > 0.5$, then this bound is impossible to satisfy), making it increasingly difficult for incorrect but diverse generations to receive positive advantage.

  In contrast, under set reinforcement learning, as a model's accuracy improves, exploratory generations can still receive positive learning signal even when incorrect. Because the set objective shares credit among all generations in a set, incorrect but diverse generations can benefit from the presence of correct generations in the same set. Consequently, the incentive for exploration does not vanish as success rate improves and does not rely on tuning a coefficient such as λ.
- Furthermore, Poly-EPO naturally assigns advantage based on how a generation contributes to achieving a synergy between exploration and exploitation over sets. This behavior arises from two components: the set RL framework and the multiplicative structure of the polychromic objective. Notably, this effect does not arise in the reward-shaped RL term and would not arise if one simply applied standard RL with a reward of the form $r(x, y) \cdot d(x, y)$, as considered in some prior work (Li et al., 2025). In such formulations, the covariance term in eq. (8) does not appear, and the learning signal cannot explicitly encourage generations that help coordinate exploration and exploitation across sets.

E PROOFS

First, we prove Proposition 3.1.

Proposition 3.1 (Restated) Fix a prompt x, and let $y_1, \cdots, y_N \overset{\text{i.i.d.}}{\sim} \pi_\theta(\cdot \mid x)$ be our independently sampled N generations and let $f : \mathcal{X} \times \mathcal{Y}^{\oplus n} \to \mathbb{R}$ be our set objective. Then,

$$\mathbb{E}[\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(y_i \mid x) \widehat{A^\sharp_{\text{marg}}}(x, y_i; f)] = M \nabla_\theta \mathbb{E}_{y_{1:n} \sim \pi_\theta(\cdot|x)} \big[f(x, y_{1:n})\big],$$

where $M \in \mathbb{R}_{>0}$ is a constant (and depends on the number of sets we construct). Consequently, after also taking expectation over $x \sim \mathcal{D}$ and scaling the learning rate, the estimator is an unbiased estimator of the set RL gradient.

Proof. We first prove the case where we construct all $K = \binom{N}{n}$ sets from $y_{1:N}$ in a combinatorial fashion. Let $G_{1:K}$ be all the unique sets we construct and let $\mathcal{G}(y)$ be the collection of sets that contain the generation y. Starting from our estimator, we get:

$$\begin{aligned}
&\mathbb{E}\left[\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(y_i \mid x) A^\sharp_{\text{marg}}(x, y_i)\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(y_i \mid x) \left(\sum_{G \in \mathcal{G}(y_i)} (f(x, G) - \frac{1}{K}\sum_{j=1}^{K} f(x, G_j)\right)\right] \\
&= \mathbb{E}\left[\sum_{G \in G_{1:K}} (f(x, G) - \frac{1}{K}\sum_{j=1}^{K} f(x, G_j)) \sum_{y_i \in G} \nabla_\theta \log \pi_\theta(y_i \mid x)\right] \\
&= \mathbb{E}\left[\sum_{G \in G_{1:K}} f(x, G) \sum_{y_i \in G} \nabla_\theta \log \pi_\theta(y_i \mid x)\right] - \mathbb{E}\left[\sum_{G \in G_{1:K}} \frac{1}{K}\sum_{j=1}^{K} f(x, G_j) \sum_{y_i \in G} \nabla_\theta \log \pi_\theta(y_i \mid x)\right] \\
&= K\mathbb{E}\left[f(x, G) \sum_{y_i \in G} \nabla_\theta \log \pi_\theta(y_i \mid x)\right] - \frac{1}{K}\mathbb{E}\left[\sum_{j=1}^{K} f(x, G_j) \sum_{G \in G_{1:K}} \sum_{y_i \in G} \nabla_\theta \log \pi_\theta(y_i \mid x)\right] \\
&= K\mathbb{E}\left[f(x, G) \sum_{y_i \in G} \nabla_\theta \log \pi_\theta(y_i \mid x)\right] - \frac{1}{K}\mathbb{E}\left[\sum_{j=1}^{K} f(x, G_j) \cdot \binom{N-1}{n-1} \sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(y_i \mid x)\right] \\
&= K\mathbb{E}\left[f(x, G) \sum_{y_i \in G} \nabla_\theta \log \pi_\theta(y_i \mid x)\right] - \frac{1}{K}\binom{N-1}{n-1}\mathbb{E}\left[\sum_{j=1}^{K} f(x, G_j) \cdot \sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(y_i \mid x)\right] \\
&= K\mathbb{E}\left[f(x, G) \sum_{y_i \in G} \nabla_\theta \log \pi_\theta(y_i \mid x)\right] - \frac{1}{K}\binom{N-1}{n-1}\sum_{j=1}^{K}\sum_{i=1}^{N} \mathbb{E}\left[f(x, G_j) \cdot \nabla_\theta \log \pi_\theta(y_i \mid x)\right]
\end{aligned}$$

Now we expand the second term. We can write the sum over tokens $y_1, \cdots, y_N$ as:

$$\begin{aligned}
&\frac{1}{K}\binom{N-1}{n-1}\sum_{j=1}^{K}\sum_{i=1}^{N} \mathbb{E}\left[f(x, G_j) \cdot \nabla_\theta \log \pi_\theta(y_i \mid x)\right] \\
&= \frac{1}{K}\binom{N-1}{n-1}\sum_{j=1}^{K}\left(\mathbb{E}\left[f(x, G_j) \cdot \sum_{y_i \in G_j} \nabla_\theta \log \pi_\theta(y_i \mid x)\right] + \sum_{y_i \notin G_j} \mathbb{E}\left[f(x, G_j) \cdot \nabla_\theta \log \pi_\theta(y_i \mid x)\right]\right)
\end{aligned}$$

$$= \binom{N-1}{n-1} \mathbb{E}\left[f(x, G) \cdot \sum_{y_i \in G} \nabla_\theta \log \pi_\theta(y_i \mid x) \right] - 0$$

Combining, we get that:

$$\mathbb{E}\left[\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(y_i \mid x) A^\sharp_{\text{marg}}(x, y_i) \right] = \left(\binom{N}{n} - \binom{N-1}{n-1} \right) \mathbb{E}\left[f(x, G) \sum_{y_i \in G} \nabla_\theta \log \pi_\theta(y_i \mid x) \right].$$

By Pascal's rule, the scaling factor $M = \binom{N}{n} - \binom{N-1}{n-1}$ is greater than 0 as long as $N > n$.

The proof for the case where we construct K sets by uniformly sampling (without replacement) is provided in the next proposition.

□

Proposition E.1. *Let $\mathcal{S} := \{S \subseteq \{1, \cdots, N\} : |S| = n\}$ which gives us a collection of all possible sets of size n without replacement (more precisely, it gives us a collection of indices which defines the collection of all possible sets). Let $K_{\text{all}} := |\mathcal{S}| = \binom{N}{n}$ which is the maximum number of sets of size n one can construct from the N generations, and let $d := \binom{N-1}{n-1}$. Let $\mathcal{T} \subseteq \mathcal{S}$ be a uniformly sampled subset of size K, sampled without replacement and independently of $y_{1:N}$.*

Define $q_K := \Pr_{\mathcal{T}}(C_i(\mathcal{T}) > 0) = 1 - \frac{\binom{\binom{N-1}{n}}{K}}{\binom{\binom{N}{n}}{K}}$, where we use the convention that $\binom{a}{K} = 0$ when $K > a$. Then

$$\mathbb{E}_{y_{1:N}, \mathcal{T}}\left[\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(y_i \mid x)\, A^\sharp_{\text{marg}}(x, y_i \mid \mathcal{T}) \right] = M_K \nabla_\theta \mathbb{E}_{y_{1:n} \sim \pi_\theta(\cdot|x)}\left[f(x, y_{1:n}) \right],$$

where

$$M_K = \frac{N}{n} q_K - 1.$$

In particular, if $N > n$ and $K > 1$, then $M_K > 0$. In other words, up to the positive scalar factor M_K, we have an unbiased estimator of the set-RL gradient.

Proof. For notational convenience, write $\nabla_i := \nabla_\theta \log \pi_\theta(y_i \mid x)$. We first decompose the left hand side as:

$$\begin{aligned}
&\mathbb{E}_{y_{1:N}, \mathcal{T}}\left[\sum_{i=1}^{N} \nabla_i A^\sharp_{\text{marg}}(x, y_i \mid \mathcal{T}) \right] \\
&= \underbrace{\mathbb{E}_{y_{1:N}, \mathcal{T}}\left[\sum_{i=1}^{N} \nabla_i \mathbf{1}\{C_i > 0\} \frac{1}{C_i} \sum_{\substack{S \in \mathcal{T} \\ i \in S}} f_S \right]}_{\text{Term 1}} - \underbrace{\mathbb{E}_{y_{1:N}, \mathcal{T}}\left[\sum_{i=1}^{N} \nabla_i \mathbf{1}\{C_i > 0\} \widehat{f}_{\mathcal{T}}(x) \right]}_{\text{Term 2}}.
\end{aligned}$$

We first simplify Term 1.

$$\begin{aligned}
\mathbb{E}_{y_{1:N}, \mathcal{T}}\left[\sum_{i=1}^{N} \nabla_i \mathbf{1}\{C_i > 0\} \frac{1}{C_i} \sum_{\substack{S \in \mathcal{T} \\ i \in S}} f_S \right] &= \mathbb{E}_{y_{1:N}, \mathcal{T}}\left[\sum_{i=1}^{N} \nabla_i \sum_{G \in \mathcal{S}, y_i \in G} \frac{\mathbf{1}\{G \in \tau\}}{C_i} f(x, G) \right] \\
&= \mathbb{E}_{y_{1:N}}\left[\sum_{i=1}^{N} \nabla_i \sum_{G \in \mathcal{S}, y_i \in G} f(x, G) \mathbb{E}_\tau\left[\frac{\mathbf{1}\{G \in \tau\}}{C_i} \right] \right]
\end{aligned}$$

Now, note that

$$\mathbb{E}_\tau\left[\sum_{G\in\mathcal{G}(y_i)}\frac{\mathbf{1}\{G\in\tau\}}{C_i}\right]=\sum_{G\in\mathcal{G}(y_i)}\mathbb{E}_\tau\left[\frac{\mathbf{1}\{G\in\tau\}}{C_i}\right].$$

On the other hand,

$$\mathbb{E}_\tau\left[\sum_{G\in\mathcal{G}(y_i)}\frac{\mathbf{1}\{G\in\tau\}}{C_i}\right]=\mathbb{E}_\tau\left[\mathbf{1}\{C_i>0\}\right]=:q_K,$$

where q_K is the probability that y_i belongs to one of the K sampled sets in τ.

Next, we note that because the sets are sampled uniformly and independently of $y_{1:N}$, we have that, for any G and G' in $\mathcal{S}$,

$$\mathbb{E}_\tau\left[\frac{\mathbf{1}\{G\in\tau\}}{C_i}\right]=\mathbb{E}_\tau\left[\frac{\mathbf{1}\{G'\in\tau\}}{C_i}\right].$$

As such

$$\sum_{G\in\mathcal{G}(y_i)}\mathbb{E}_\tau\left[\frac{\mathbf{1}\{G\in\tau\}}{C_i}\right]=d\mathbb{E}_\tau\left[\frac{\mathbf{1}\{G\in\tau\}}{C_i}\right]=q_K,$$

and so,

$$\mathbb{E}_\tau\left[\frac{\mathbf{1}\{G\in\tau\}}{C_i}\right]=\frac{q_K}{d}.$$

This allows us to simplify:

$$\begin{aligned}
\mathbb{E}_{y_{1:N},\mathcal{T}}\left[\sum_{i=1}^N\nabla_i\mathbf{1}\{C_i>0\}\frac{1}{C_i}\sum_{\substack{S\in\mathcal{T}\\ i\in S}}f_S\right]&=\mathbb{E}_{y_{1:N}}\left[\sum_{i=1}^N\nabla_i\sum_{G\in\mathcal{S},y_i\in G}f(x,G)\mathbb{E}_\tau\left[\frac{\mathbf{1}\{G\in\tau\}}{C_i}\right]\right]\\
&=\frac{q_K}{d}\mathbb{E}_{y_{1:N}}\left[\sum_{i=1}^N\nabla_i\sum_{G\in\mathcal{S},y_i\in G}f(x,G)\right]\\
&=\frac{q_K}{d}\mathbb{E}_{y_{1:N}}\left[\sum_{G\in\mathcal{S}}f(x,G)\sum_{y_i\in G}\nabla_i\right]\\
&=\frac{q_K}{d}\sum_{G\in\mathcal{S}}\mathbb{E}_{y_{1:N}}\left[f(x,G)\sum_{y_i\in G}\nabla_i\right]\\
&=\frac{q_K}{d}\binom{N}{n}\nabla_\theta\mathbb{E}_{y_{1:n}}[f(x,y_{1:n})].
\end{aligned}$$

Next, we simplify Term 2. Since $\widehat{f}_\mathcal{T}(x)=\frac{1}{K}\sum_{U\in\mathcal{T}}f(x,U)$, we have

$$\mathbb{E}_{y_{1:N},\tau}\left[\sum_{i=1}^N\nabla_i\mathbf{1}\{c_i>0\}\widehat{f}_\mathcal{T}(x)\right]=\mathbb{E}_{y_{1:N}}\left[\sum_{i=1}^N\nabla_i\sum_{U\in\mathcal{S}}f(x,U)\mathbb{E}_\mathcal{T}\left[\mathbf{1}\{C_i>0\}\frac{\mathbf{1}\{U\in\mathcal{T}\}}{K}\right]\right].$$

Note that the inner expectation is conditioned on a fixed U. Now, for a fixed U and a fixed i, there are two cases. Case 1: $y_i\in U$. In this case $\mathbb{E}_\mathcal{T}\left[\mathbf{1}\{C_i>0\}\frac{\mathbf{1}\{U\in\mathcal{T}\}}{K}\right]=\frac{1}{K}\mathbb{P}(U\in\mathcal{T})=\frac{1}{K}\cdot\frac{K}{K_{\text{all}}}=\frac{1}{K_{\text{all}}}$. Case 2: $y_i\notin U$. In this case, $f(x,U)$ is independent of y_i, in which case we use the fact that $\mathbb{E}_{y_{1:N}}[f(x,U)\nabla_i]=0$. So, we can simplify as:

$$\begin{aligned}
\mathbb{E}_{y_{1:N},\tau}\left[\sum_{i=1}^{N}\nabla_i \mathbf{1}\{c_i>0\}\widehat{f}_{\mathcal{T}}(x)\right] &= \mathbb{E}_{y_{1:N}}\left[\sum_{i=1}^{N}\nabla_i \sum_{U\in\mathcal{S}} f(x,U)\mathbb{E}_{\mathcal{T}}\left[\mathbf{1}\{C_i>0\}\frac{\mathbf{1}\{U\in\mathcal{T}\}}{K}\right]\right] \\
&= \mathbb{E}_{y_{1:N}}\left[\sum_{U\in\mathcal{S}} f(x,U)\sum_{y_i\in U}\frac{1}{K_{\text{all}}}\nabla_i\right] \\
&= \frac{1}{K_{\text{all}}}\sum_{U\in\mathcal{S}}\mathbb{E}_{y_{1:N}}\left[f(x,U)\sum_{y_i\in U}\nabla_i\right] \\
&= \nabla_\theta \mathbb{E}_{y_{1:n}}\left[f(x,y_{1:n})\right].
\end{aligned}$$

Finally, we verify the expression for q_K. The event $C_i = 0$ means that none of the K sampled sets contains i. There are $K_{\text{all}} - \binom{N-1}{n-1}$ sets in $\mathcal{S}$ that do not contain i, so

$$\Pr(C_i = 0) = \frac{\binom{K_{\text{all}}-d}{K}}{\binom{K_{\text{all}}}{K}}.$$

Therefore,

$$q_K = \Pr(C_i > 0) = 1 - \frac{\binom{K_{\text{all}}-d}{K}}{\binom{K_{\text{all}}}{K}}.$$

Combining these two terms and doing a routine check shows that the scaling factor is positive as long as $N > n$ and $K > 1$. □

Now, we prove Lemma D.1.

Lemma D.1 (Restated) In set reinforcement learning, assuming the objective function f is symmetric in its arguments and the learning rate is α, the shift in probability mass on a fixed generation y after one step gradient update from π_θ^k to π_θ^{k+1} can be written as:

$$\log \pi_\theta^{k+1}(y \mid x) - \log \pi_\theta^k(y \mid x) = \alpha n \pi_\theta^k(y \mid x)\left[\mathbb{E}_{y_{2:n}\sim\pi_\theta^k(\cdot|x)}\left[f(x,y,y_{2:n})\right] - \mathbb{E}_{y_{1:n}\sim\pi_\theta^k(\cdot|x)}\left[f(x,y_{1:n})\right]\right]. \tag{15}$$

Proof. This follows from Hamid et al. (2026) by noting that when a set is homogeneous, then its scaffold value, $\Lambda_f(y^{\oplus n};\pi_\theta^k,x)$, can be written as the probability of sampling y times the marginal set advantage of y - both under π_θ^k. □

F Additional Experiments

F.1 Synthetic Domains with Infinitely Many Valid Strategies

We now test our method in a controlled setting where each task admits infinitely many distinct strategies that can yield a correct answer. Our goal is to test whether Poly-EPO discovers a broader repertoire of successful strategies than GRPO. This probes a central property of exploration: an effective learner should not only achieve reward, but also discover diverse ways of achieving it through exploration. We consider two tasks in this section:

- **Polynomial Solving.** In this task, the model is given a polynomial relation and must output a pair $(x, y) \in \mathbb{R}^2$ that satisfies it. For example, given $y = x^2 + 3x + 7$, one valid solution is $(x = 1, y = 11)$.
- **Multi-digit Multiplication.** In this task, the model is asked to compute the product of two integers by decomposing the operands and distributing the multiplication. For example, to compute 343×67, one valid decomposition is $(340 + 3)(70 - 3)$, which expands to $(340 \times 70) - (340 \times 3) + (3 \times 70) - (3 \times 3) = 22981$.

For both tasks, we use `Qwen-3-1.7B-Base` as the base model and compare between GRPO and POLY-EPO. We use `Gemini-2.0-Flash` as the LM-judge for clustering generations when measuring diversity. For multi-digit multiplication, the judge clusters responses according to the strategy used; for example, a solution based on factorization is assigned to a different cluster than one based on partial products over place values. For polynomial solving, diversity is defined directly over the final answer: two outputs (x_1, y_1) and (x_2, y_2) are assigned to different clusters whenever $x_1 \neq x_2$ or $y_1 \neq y_2$.

fig. 5 shows the training dynamics of the models, including how the diversity of generations evolves over training. In particular, we measure the number of distinct clusters among correct responses for each prompt, averaged over prompts in a batch, to quantify the diversity of successful strategies learned by the policy. We also measure the number of distinct clusters among incorrect responses to quantify the diversity of incorrect but exploratory generations produced by the policy.

We find that GRPO quickly collapses to a single successful strategy. In contrast, POLY-EPO continues to explore and discovers more than five times as many distinct successful strategies. Furthermore, POLY-EPO preserves higher diversity over incorrect generations, suggesting that when exploring optimistically, it has a larger coverage over strategies that are being attempted. These results indicate that POLY-EPO is substantially more effective at maintaining exploration and uncovering novel high-reward strategies in domains with many valid solutions.

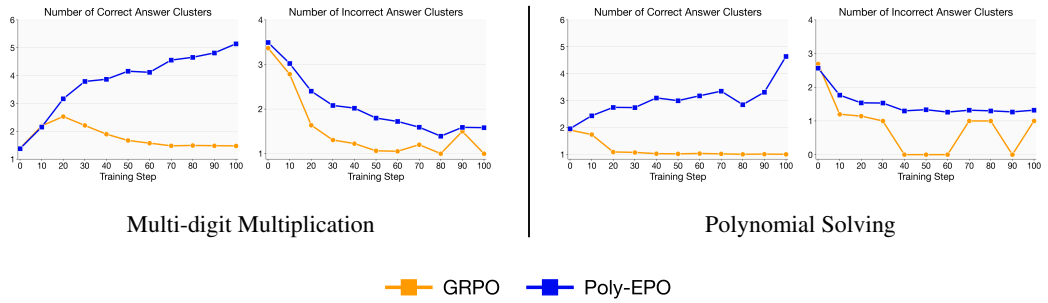
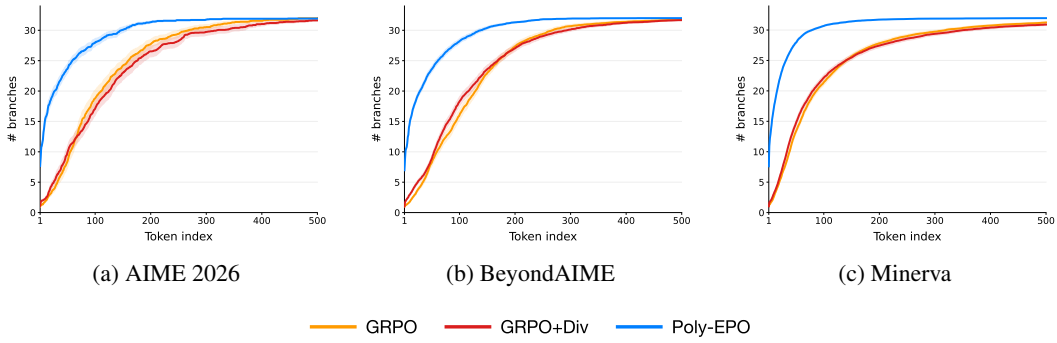


Figure 5: **Training dynamics on synthetic domains**. The two left plots correspond to **multi-digit multiplication**, and the two right plots correspond to **polynomial solving**. We measure strategy diversity by clustering generations and reporting the number of distinct clusters among correct and incorrect responses for each prompt, averaged across the batch. Relative to GRPO, POLY-EPO maintains substantially greater diversity throughout training and discovers many more successful strategies.

F.2 BRANCHING EXPERIMENTS

We further analyze the distinct behaviors induced by each post-training method by studying the diversity of model generations on the test set after fine-tuning with GRPO or POLY-EPO. In this analysis, we characterize diversity through the *branching structure* of sampled generations. For each prompt x in the test set, we sample k generations $y_1, \ldots, y_k \sim \pi_\theta(\cdot \mid x)$ and organize them into a tree according to their shared token prefixes. Specifically, if two generations agree on their prefix through token $h-1$ but differ at token h, we say that they branch at token h. Since all generations begin with the same beginning-of-sequence token (`[BOS]`), the tree has a single root.

In fig. 6, we visualize example rollout trees and also quantify branching by measuring, at each token position, the number of active branches among sampled generations. We find that models trained with POLY-EPO branch substantially earlier in the generation process than models trained with GRPO. Although the GRPO-trained model eventually reaches a similar number of branches at later token positions, its generations remain coupled for much longer before diverging. This suggests that POLY-EPO encourages the model to differentiate its reasoning trajectories earlier, whereas GRPO tends to delay diversification until later in the rollout. Because early branching more naturally corresponds to committing to distinct reasoning strategies, these results indicate that POLY-EPO promotes broader strategic exploration at test time.



(a) AIME 2026 (b) BeyondAIME (c) Minerva

Figure 6: **POLY-EPO promotes broader branching during reasoning generation. Top:** Branching structure of 8 sampled rollouts on an AIME 2026 problem for POLY-EPO (left) and GRPO (right). Nodes represent shared-prefix clusters, and edges denote branching into distinct continuations. **Bottom:** Average number of active branches as a function of token position on AIME 2026, BeyondAIME, and Minerva. POLY-EPO consistently maintains more branches earlier in generation, indicating broader exploration over reasoning trajectories.