ACTOR-CURATOR: CO-ADAPTIVE CURRICULA VIA POLICY-IMPROVEMENT BANDITS FOR POST-TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Post-training large foundation models with reinforcement learning typically involves selecting training problems from massive and heterogeneous datasets, where the choice of data has a critical impact on training stability, sample efficiency, and final performance. In this work, we propose ACTOR-CURATOR, a scalable and fully automated framework for reinforcement learning post-training of large language models (LLMs) that learns to adaptively curate training problems. ACTOR-CURATOR trains a neural *curator* that dynamically selects problems from large problem banks by directly optimizing for expected policy performance improvement. We formulate problem selection as a non-stationary stochastic bandit problem, derive a principled loss function based on online stochastic mirror descent, and establish regret guarantees under partial feedback. Empirically, ACTOR-CURATOR consistently outperforms uniform sampling and strong learning-based baselines across a wide range of challenging reasoning benchmarks, demonstrating improved training stability and efficiency. Notably, it achieves relative gains of **28.6% on AIME2024** and **30.5% on ARC-1D** over the strongest baseline and up to **80% speedup**. These results suggest that ACTOR-CURATOR provides a practical and principled approach to scalable, adaptive curriculum learning for LLM post-training.

1 INTRODUCTION

Reinforcement learning (RL) has become a central paradigm for post-training foundation models, enabling improvements in reasoning, alignment, and task-specific performance beyond supervised fine-tuning (Shao et al., 2024). In this setting, the choice, ordering, and frequency of training problems play a critical role in determining convergence speed, training stability, and final generalization performance, motivating the use of curriculum learning to adaptively select training data (Bengio et al., 2009; Tzannetos et al., 2023; Parashar et al., 2025). However, applying curriculum learning to modern foundation model post-training is challenging: post-training datasets are **large, diverse, and continuously evolving**, while actor updates induce **complex, non-stationary training dynamics**. Traditional curriculum learning approaches—based on manual difficulty annotations, hand-designed problem buckets, or tabular per-problem statistics (Asada et al., 1996; Wu & Tian, 2017; Yengera et al., 2021)—do not scale to such settings, fail to generalize to unseen problems, and are brittle when problem utility changes as the policy improves. More-

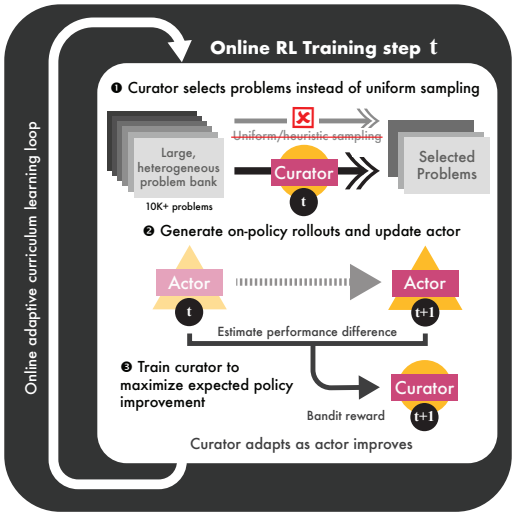


Figure 1: **Online training loop of ACTOR-CURATOR.** At each reinforcement learning step, a learned curator adaptively selects problems from a large problem bank instead of uniform sampling. The actor is updated on these problems, after which a bandit-style reward based on post-update policy improvement is used to train the curator. As the actor improves, the curator adapts to prioritize problems that yield the greatest expected performance gains.

over, effective curricula must balance **exploration** of under-sampled problems with **exploitation** of those that most improve the current policy, further complicating scalable curriculum design.

In this work, we propose ACTOR-CURATOR (AC), a scalable and fully automated problem curation framework for RL post-training that jointly trains an actor and a curator in an online, on-policy manner. At the core of ACTOR-CURATOR is a learned *curator* that adaptively selects training problems at each iteration and function-approximates over large, heterogeneous datasets. The curator is trained to directly maximize a **policy improvement objective**, assigning higher probability to problems expected to induce the greatest improvement in the actor's performance. Unlike prior curricula that rely on heuristic signals such as absolute mean advantage or difficulty proxies (Chen et al., 2025a; Gao et al., 2025; Wang et al., 2025), our objective is derived from expected policy improvement, providing a principled and actor-aware learning signal. As a result, ACTOR-CURATOR naturally *adapts to evolving actor training dynamics* and allows it to be seamlessly combined with a wide range of RL algorithms.

To optimize the curator, we formalize problem selection as a **non-stationary stochastic bandit** problem with partial feedback. The curator is trained online and on-policy alongside the actor using **online stochastic mirror descent** (OSMD) Lattimore & Szepesvari (2017), which explicitly balances exploration and exploitation under non-stationarity. This differs from prior approaches that primarily rely on regression-style objectives and do not explicitly model the bandit structure or **partial observability** inherent in adaptive data selection (Tzannetos et al., 2023; Gao et al., 2025). To scale beyond tabular formulations, we derive a function-approximation variant of OSMD that trains the curator as a neural network, enabling generalization across problems and robustness to large, dynamic datasets. Finally, we introduce a PPO-style proximal clipping objective to stabilize curator optimization in practice.

Empirically, ACTOR-CURATOR enables effective curriculum learning at scale *without human annotations, difficulty labels, or manual dataset structuring*. Across diverse reasoning benchmarks—**Countdown, Zebra, MATH, AIME, and ARC-1D**—it consistently outperforms uniform sampling and strong baselines, achieving up to **30%** higher peak performance on ARC-1D, **28%** on AIME24, and up to **80%** faster convergence to comparable performance.

In summary, our main contributions are:

- **Automated problem curation framework for RL post-training.** We introduce ACTOR-CURATOR, a scalable framework that learns a neural curator to adaptively select training problems in an online, on-policy manner, enabling curriculum learning over large, heterogeneous datasets without human annotations or manual structuring.
- **A policy-improvement–driven bandit formulation of data curation.** We cast problem selection as a non-stationary stochastic bandit problem and derive a principled learning signal grounded in policy improvement theory, optimized via an OSMD-based bandit objective with regret guarantees under partial feedback.

2 PROBLEM FORMULATION

We study problem curation for reinforcement learning (RL) post-training of large language models (LLMs), where training is performed over large and heterogeneous collections of problems. Our goal is to design an adaptive data selection strategy that determines which training problems an LLM should train on at each iteration in order to maximize overall post-training performance.

2.1 RL POST-TRAINING SETTING

Let $\mathcal{X} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{|\mathcal{X}|}$ denote a large collection of training problems, and let $p_{\mathcal{X}}$ be a fixed evaluation distribution over $\mathcal{X}$. Let π denote a pretrained autoregressive language model, which induces a conditional distribution $\boldsymbol{y} \sim \pi(\cdot \mid \boldsymbol{x})$ over solutions for each problem $\boldsymbol{x}$. A reward model $R : \mathcal{Y} \times \mathcal{X} \to [0, 1]$ assigns a scalar score to each solution. The post-training objective is to maximize expected reward under the evaluation distribution:

$$J(\pi) \;\triangleq\; \mathbb{E}_{\boldsymbol{x}\sim p_{\mathcal{X}},\, \boldsymbol{y}\sim\pi(\cdot|\boldsymbol{x})}\big[R(\boldsymbol{y} \mid \boldsymbol{x})\big]. \tag{1}$$

In this work, the reinforcement learning algorithm, reward model, and rollout procedure are fixed. Our focus is on how training problems are selected across iterations.

2.2 TRAINING DYNAMICS

Training proceeds in iterations. At iteration t, a *curator* selects a subset of training problems $\mathcal{X}^t \subset \mathcal{X}$. Given this selection, the *actor* π^t is rolled out on each $\boldsymbol{x} \in \mathcal{X}^t$ to produce solutions and corresponding rewards

$$\mathcal{Y}^t_{\boldsymbol{x}} \triangleq \{\boldsymbol{y}^{(j)} \sim \pi^t(\cdot \mid \boldsymbol{x})\}_{j=1}^{|\mathcal{Y}_{\boldsymbol{x}}|}, \qquad \mathcal{R}^t_{\boldsymbol{x}} \triangleq \{R(\boldsymbol{y}^{(j)} \mid \boldsymbol{x})\}_{j=1}^{|\mathcal{Y}_{\boldsymbol{x}}|}.$$

These trajectories form the dataset

$$\mathcal{D}^t \triangleq \big\{(\boldsymbol{x}, \mathcal{Y}^t_{\boldsymbol{x}}, \mathcal{R}^t_{\boldsymbol{x}}) \mid \boldsymbol{x} \in \mathcal{X}^t\big\}, \qquad \pi^{t+1} \leftarrow \mathcal{A}(\pi^t, \mathcal{D}^t). \tag{2}$$

Here $\mathcal{A}$ may correspond to any standard post-training algorithm (e.g., GRPO (Shao et al., 2024) or GSPO (Ahmadian et al., 2024)). We emphasize that the curator influences learning only indirectly through data selection, while the actor is solely responsible for policy optimization.

2.3 CURRICULUM LEARNING AS PROBLEM SELECTION

The central problem addressed in this work is how to choose the training subsets $\mathcal{X}^t$ across iterations. Different choices of $\mathcal{X}^t$ induce different actor updates and therefore different trajectories of policy improvement. We formalize curriculum learning as a sequential decision-making problem. At each iteration t, a *curator* selects a subset of training problems $\mathcal{X}^t \subset \mathcal{X}$. This selection induces a performance improvement $J(\pi^{t+1}) - J(\pi^t)$. The curator's objective is to maximize cumulative performance gains over training:

$$\max_{\{\mathcal{X}^t\}_{t=1}^T} \sum_{t=1}^{T} \big(J(\pi^{t+1}) - J(\pi^t)\big). \tag{3}$$

2.4 CHALLENGES OF ADAPTIVE PROBLEM SELECTION

Effective curation is challenging for several reasons:

- **Large action space.** The problem set $\mathcal{X}$ is large and might change across time, making it infeasible to manually define curricula or track per-problem statistics.
- **Partial feedback.** At each iteration, feedback is observed only for the problems selected for training; the utility of unselected problems remains unknown.
- **Non-stationarity.** The usefulness of a problem depends on the current actor π^t and changes as the actor improves. It is also highly dependent on the actor update method.
- **Exploration–exploitation trade-off.** The curator must balance exploring under-sampled problems whose utility is uncertain with exploiting problems that are known to drive policy improvement.

These challenges motivate a curriculum learning approach that operates at scale, learns online from partial feedback, and explicitly accounts for the non-stationary relationship between training problems and policy improvement.

3 METHOD

We now present ACTOR-CURATOR, a curriculum learning framework that trains a learned *curator* to adaptively select training problems for RL post-training of large language models. The key idea is to treat problem selection as a non-stationary bandit problem and to train the curator to directly maximize *policy improvement*—the expected performance gain induced by each actor update—using online stochastic mirror descent (OSMD) under partial feedback.

3.1 OVERVIEW OF THE TRAINING LOOP

Training proceeds in iterations. At iteration t, the following steps are performed:

1. We sample a training subset $\mathcal{X}^t \subset \mathcal{X}$ based on probabilities produced by the curator.
2. The actor π^t is rolled out on $\mathcal{X}^t$ to collect trajectories and updated to π^{t+1} using any RL update.
3. Compute a per-problem policy improvement estimate for problems in $\mathcal{X}^t$ using π^t and π^{t+1}
4. The curator is updated online using bandit feedback derived from these improvement estimates.

The curator is trained jointly with the actor in an on-policy manner, allowing the curriculum to adapt dynamically as the actor improves. Figure 2 illustrates this process. Pseudocode is provided in Algorithm 1.

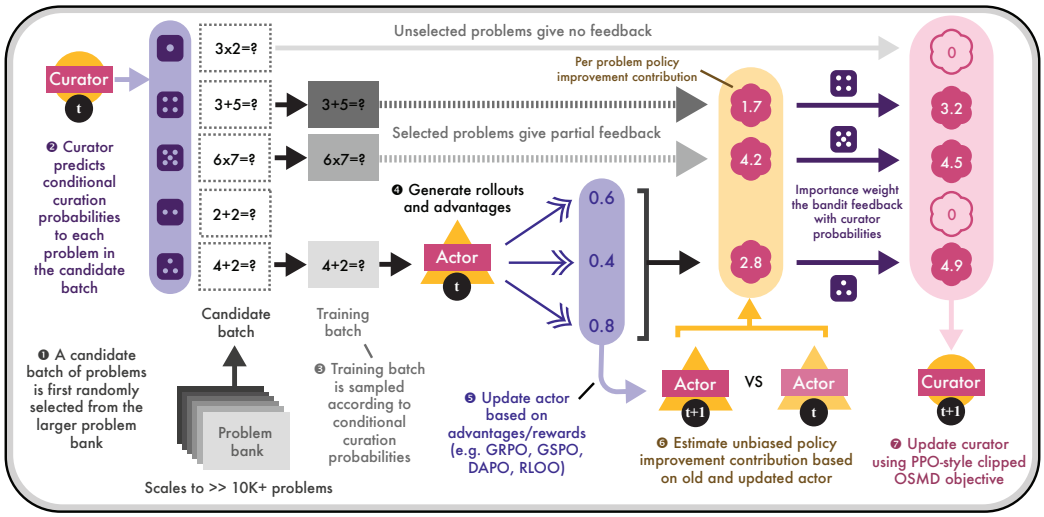


Figure 2: **Single training iteration of ACTOR-CURATOR.** At each iteration, a candidate set of problems is sampled from a fixed proposal distribution. The curator reweights this candidate set to select training problems for the actor. After the actor update, per-problem policy improvement is estimated using pre- and post-update policies. The curator observes bandit feedback only on selected problems and is updated using a PPO-style approximation of online stochastic mirror descent (OSMD).

Algorithm 1 Actor–Curator: Self-driven curriculum learning

1: **Input:** dataset $\mathcal{X}$, pretrained LLM (actor) π^0, reward model R, curator model C_{ϕ^0}, proposal distribution $\tilde{q}$
2: **Hyperparameters:** number of training steps T, rollouts per problem $|\mathcal{Y}_{\boldsymbol{x}}|$, candidate batch size $|\tilde{\mathcal{X}}^t|$, training batch size $|\mathcal{X}^t|$
3: **for** $t = 0$ to $T-1$ **do** ▷ Training steps
4: $\quad\mathcal{D}^t \leftarrow \emptyset$
5: $\quad$**Proposal step:** sample a candidate batch $\tilde{\mathcal{X}}^t \subset \mathcal{X}$ according to $\tilde{q}$
6: $\quad$**Selection step:** sample a training batch $\mathcal{X}^t \subset \tilde{\mathcal{X}}^t$ using the curator C_{ϕ^t}
7: $\quad$**for** each problem $\boldsymbol{x} \in \mathcal{X}^t$ **do**
8: $\quad\quad$Roll out solutions $\mathcal{Y}^t_{\boldsymbol{x}} = \{\boldsymbol{y}^{(j)} \sim \pi^t(\cdot \mid \boldsymbol{x})\}_{j=1}^{|\mathcal{Y}_{\boldsymbol{x}}|}$
9: $\quad\quad$Compute rewards $\mathcal{R}^t_{\boldsymbol{x}} = \{R(\boldsymbol{y}^{(j)} \mid \boldsymbol{x})\}_{j=1}^{|\mathcal{Y}_{\boldsymbol{x}}|}$
10: $\quad\quad$Add $(\boldsymbol{x}, \mathcal{Y}^t_{\boldsymbol{x}}, \mathcal{R}^t_{\boldsymbol{x}})$ to $\mathcal{D}^t$
11: $\quad$**end for**
12: $\quad$**Actor update:** $\pi^{t+1} \leftarrow \mathcal{A}(\pi^t, \mathcal{D}^t)$
13: $\quad$**Curator utilities:** compute $\hat{\boldsymbol{U}}^t$ using equation 10 (and π^{t+1})
14: $\quad$**Curator update:** $\phi^{t+1} \leftarrow \arg\min_\phi \mathcal{L}_{\text{cur}}(\phi)$ using equation 12
15: **end for**

3.2 POLICY IMPROVEMENT AS THE CURATOR LEARNING SIGNAL

The curator's objective is to select problems that maximize improvement in the actor's performance under the fixed evaluation distribution $p_{\mathcal{X}}$.

Performance improvement identity. Define the performance improvement at iteration t as

$$u^t \triangleq J(\pi^{t+1}) - J(\pi^t).$$

In the single-turn setting, the performance difference identity (Kakade & Langford, 2002) gives

$$u^t = \mathbb{E}_{\boldsymbol{x}\sim p_{\mathcal{X}}} \mathbb{E}_{\boldsymbol{y}\sim\pi^{t+1}(\cdot|\boldsymbol{x})} \left[A_{\pi^t}(\boldsymbol{y} \mid \boldsymbol{x})\right], \quad A_{\pi^t}(\boldsymbol{y} \mid \boldsymbol{x}) \triangleq R(\boldsymbol{y} \mid \boldsymbol{x}) - \mathbb{E}_{\boldsymbol{y}'\sim\pi^t(\cdot|\boldsymbol{x})} \left[R(\boldsymbol{y}' \mid \boldsymbol{x})\right]. \quad (4)$$

Applying importance sampling yields

$$u^t = \mathbb{E}_{\boldsymbol{x}\sim p_{\mathcal{X}}} \mathbb{E}_{\boldsymbol{y}\sim\pi^t(\cdot|\boldsymbol{x})} \left[\frac{\pi^{t+1}(\boldsymbol{y} \mid \boldsymbol{x})}{\pi^t(\boldsymbol{y} \mid \boldsymbol{x})} A_{\pi^t}(\boldsymbol{y} \mid \boldsymbol{x})\right]. \quad (5)$$

Per-problem utility. Eq. (5) decomposes additively across problems. For each $\boldsymbol{x} \in \mathcal{X}$, define the per-problem utility

$$u^t_{\boldsymbol{x}} \triangleq p_{\mathcal{X}}(\boldsymbol{x}) \mathbb{E}_{\boldsymbol{y} \sim \pi^t(\cdot|\boldsymbol{x})} \left[\frac{\pi^{t+1}(\boldsymbol{y} \mid \boldsymbol{x})}{\pi^t(\boldsymbol{y} \mid \boldsymbol{x})} A_{\pi^t}(\boldsymbol{y} \mid \boldsymbol{x}) \right]. \tag{6}$$

By construction, $u^t = \sum_{\boldsymbol{x}} u^t_{\boldsymbol{x}}$. Although the actor update couples all selected problems, $u^t_{\boldsymbol{x}}$ provides a principled first-order credit assignment signal under small policy updates, grounded in policy improvement theory. We provide further explanation in App. E.

3.3 Tabular OSMD formulation

We cast curriculum learning as a non-stationary stochastic bandit problem, where each problem $\boldsymbol{x}$ corresponds to an arm with time-varying utility $u^t_{\boldsymbol{x}}$. The curator is optimized using **online stochastic mirror descent** (OSMD) under bandit feedback (Lattimore & Szepesvari, 2017). We start with the tabular formulation first, where the curator maintains a probability mass function $\boldsymbol{p}^t \in \Delta_\alpha(\mathcal{X})$ over a finite set of problems, where $\boldsymbol{p}^t$ is clipped to the sampling distribution $p^t(\boldsymbol{x} \mid \tilde{\mathcal{X}}^t) \geq \alpha > 0$. In Sec. 3.4 we show how to represent p^t using a learned model.

Utility estimation and OSMD bandit feedback. For each $\boldsymbol{x} \in \mathcal{X}^t$, let $\mathcal{Y}^t_{\boldsymbol{x}}$ denote rollouts from $\pi^t(\cdot \mid \boldsymbol{x})$. We estimate Eq. (6) via

$$\hat{U}^t_{\boldsymbol{x}} \triangleq p_{\mathcal{X}}(\boldsymbol{x}) \frac{\mathbb{I}\{\boldsymbol{x} \in \mathcal{X}^t\}}{p^t(\boldsymbol{x})} \hat{A}^t(\cdot \mid \boldsymbol{x}), \qquad \hat{A}^t(\cdot \mid \boldsymbol{x}) \triangleq \frac{1}{|\mathcal{Y}^t_{\boldsymbol{x}}|} \sum_{\boldsymbol{y} \in \mathcal{Y}^t_{\boldsymbol{x}}} \frac{\pi^{t+1}(\boldsymbol{y} \mid \boldsymbol{x})}{\pi^t(\boldsymbol{y} \mid \boldsymbol{x})} A(\boldsymbol{y} \mid \boldsymbol{x}) \tag{7}$$

where $\hat{A}^t$ is the importance normalized estimated average advantage. This estimate is agnostic to the specifics of the actor update method as long as an updated actor is produced.

Theorem 1 (Unbiasedness) $\mathbb{E}[\hat{U}^t_{\boldsymbol{x}}] = u^t_{\boldsymbol{x}}$.

We prove this in App. B. For decoder language models, $\pi^{t+1}(\boldsymbol{y} \mid \boldsymbol{x})$ is obtained via a single forward pass of the updated model on the previous solution.

OSMD update Given bandit feedback, the curator is updated with a negative-entropy regularizer:

$$p^{t+1} \leftarrow \arg\min_{\boldsymbol{p} \in \Delta_\alpha(\mathcal{X})} \left\{ -\eta \langle \boldsymbol{p}, \hat{\boldsymbol{U}}^t \rangle + \mathrm{KL}(\boldsymbol{p} \| \boldsymbol{p}^t) \right\}. \tag{8}$$

This yields the exponentiated-gradient update $p^{t+1}(\boldsymbol{x}) \propto p^t(\boldsymbol{x}) \exp(\eta \hat{U}^t_{\boldsymbol{x}})$.

3.4 Function approximation for curator training

Although the OSMD update in Eq. (8) is defined over a distribution on the entire problem set $\mathcal{X}$, explicitly maintaining and updating tabular probabilities is infeasible when $\mathcal{X}$ is large. We therefore parameterize the curator using a neural network that assigns a positive score to each problem and implicitly defines a probability distribution. The curator and induced distribution are defined as

$$C_\phi : \boldsymbol{x} \mapsto w_\phi(\boldsymbol{x}), \qquad p_\phi(\boldsymbol{x}) \triangleq \frac{w_\phi(\boldsymbol{x})}{\sum_{\boldsymbol{x}' \in \mathcal{X}} w_\phi(\boldsymbol{x}')}, \quad w_\phi(\boldsymbol{x}) > 0.$$

OSMD surrogate objective. To implement the OSMD update Eq. (8) with function approximation, we optimize the following surrogate objective:

$$\mathcal{L}_{\mathrm{cur}}(\phi) = \mathrm{KL}\big(p_\phi \parallel p^t\big) - \eta \langle p_\phi, \hat{\boldsymbol{U}}^t \rangle, \tag{9}$$

where p^t denotes the curator distribution from iteration t.

3.5 Two-stage sampling

Sampling directly from a curator distribution over the full problem set $\mathcal{X}$ is computationally infeasible at scale. We therefore adopt a two-stage sampling scheme that separates *coverage* from *adaptive curation* while preserving unbiased utility estimation.

At iteration t, we first sample a candidate set $\tilde{\mathcal{X}}^t \subset \mathcal{X}$ from a fixed proposal distribution $\tilde{q}$. Let

$$q(\boldsymbol{x}) \triangleq \Pr_{\tilde{\mathcal{X}} \sim \tilde{q}}\Big(\boldsymbol{x} \in \tilde{\mathcal{X}}\Big)$$

denote the induced marginal inclusion probability, and assume $q(\boldsymbol{x}) \geq q_{\min} > 0$ for all $\boldsymbol{x} \in \mathcal{X}$.

Conditioned on $\tilde{\mathcal{X}}^t$, the curator samples a training set $\mathcal{X}^t \subset \tilde{\mathcal{X}}^t$ according to the restricted distribution

$$p^t(\boldsymbol{x} \mid \tilde{\mathcal{X}}^t) \triangleq \frac{w^t(\boldsymbol{x})}{\sum_{\boldsymbol{x}' \in \tilde{\mathcal{X}}^t} w^t(\boldsymbol{x}')},$$

where $w^t(\boldsymbol{x}) > 0$ is the curator score at iteration t. This allows the curator to prioritize problems while operating only on a small candidate batch.

Utility estimation. The unbiased two-stage estimator corresponding to Eq. (7) is

$$\hat{U}^t_{\text{two},\boldsymbol{x}} \;\triangleq\; p_{\mathcal{X}}(\boldsymbol{x}) \, \frac{\mathbb{I}\{\boldsymbol{x} \in \mathcal{X}^t\}}{q(\boldsymbol{x}) \, p^t(\boldsymbol{x} \mid \tilde{\mathcal{X}}^t)} \; \hat{A}^t(\cdot \mid \boldsymbol{x}), \tag{10}$$

which corrects for both proposal and curation sampling probabilities. This estimator satisfies $\mathbb{E}[\hat{U}^t_{\text{two},\boldsymbol{x}}] = u^t_{\boldsymbol{x}}$ (see App. B). Substituting Eq. (10) into the OSMD update Eq. (8) yields the surrogate objective

$$\mathcal{L}_{\text{two}}(\phi) = \mathrm{KL}\big(p_\phi \parallel p^t\big) - \eta \sum_{\boldsymbol{x} \in \mathcal{X}^t} \frac{p_\phi(\boldsymbol{x} \mid \tilde{\mathcal{X}}^t)}{p^t(\boldsymbol{x} \mid \tilde{\mathcal{X}}^t)} \, \frac{p_{\mathcal{X}}(\boldsymbol{x}) \, \hat{A}^t(\cdot \mid \boldsymbol{x})}{q(\boldsymbol{x})}, \tag{11}$$

where $p_\phi(\cdot \mid \tilde{\mathcal{X}}^t)$ is the conditional curator distribution.

Regret guarantee. We now state a regret bound for curator optimization under two-stage sampling. The bound characterizes the curator's ability to track the best sequence of problem-selection distributions in hindsight despite non-stationary utilities. A proof is provided in App. C.

Theorem 2 *Assume the curator is updated using OSMD with a negative-entropy regularizer and receives bandit feedback $\hat{U}^t_{\text{two},\boldsymbol{x}}$ satisfying $\mathbb{E}[\hat{U}^t_{\text{two},\boldsymbol{x}}] = u^t_{\boldsymbol{x}}$. Then the cumulative dynamic regret satisfies*

$$\mathrm{Reg}_T \;\leq\; O\Big(T^{2/3} V_T^{1/3}\Big), \quad V_T \triangleq \sum_{t=2}^{T} \max_{\boldsymbol{x} \in \mathcal{X}} |u^t_{\boldsymbol{x}} - u^{t-1}_{\boldsymbol{x}}|$$

where Reg_T is the regret against the best available arm, ignoring uniform exploration which we define in App. D, and V_T is a measure of how rapid the utility of a problem changes over time t.

While we focus on two-stage sampling for efficiency, ACTOR-CURATOR is compatible with other approximate sampling schemes (e.g., Metropolis–Hastings), provided marginal inclusion probabilities are roughly proportional to curator-assigned weights.

3.6 PROXIMAL CURATOR OPTIMIZATION

Directly optimizing the KL-regularized objective in equation 9 can be unstable with neural network parameterization (Schulman et al., 2015). Following proximal policy optimization (PPO) (Schulman et al., 2017), we adopt a clipped surrogate objective. Define the importance ratio and sub-objective as

$$\rho_\phi(\boldsymbol{x}) \triangleq \frac{p_\phi(\boldsymbol{x} \mid \tilde{\mathcal{X}}^t)}{p^t(\boldsymbol{x} \mid \tilde{\mathcal{X}}^t)}, \quad g^t(\boldsymbol{x}) \triangleq \frac{p_{\mathcal{X}}(\boldsymbol{x}) \hat{A}^t(\cdot \mid \boldsymbol{x})}{q(\boldsymbol{x})}$$

Starting from equation 11, we replace the linear probability ratio term with a clipped surrogate

$$\begin{aligned} \mathcal{L}^{\text{PCO}}_{\text{cur}}(\phi) = -\eta \sum_{\boldsymbol{x} \in \tilde{\mathcal{X}}^t} \min\Big(&\rho_\phi(\boldsymbol{x}) \, g^t(\boldsymbol{x}), \\ &\mathrm{clip}(\rho_\phi(\boldsymbol{x}), \rho_{\min}, \rho_{\max}) \, g^t(\boldsymbol{x})\Big), \end{aligned} \tag{12}$$

where $\rho_{\min}, \rho_{\max}$ are clipping parameters. This objective constrains successive curator updates while preserving the behavior of online mirror descent in practice.

Table 1: **Peak validation performance** on problems within 100 training steps for different methods with qwen2.5-3b. ACTOR-CURATOR outperforms both other learning based methods (PCL) and methods that rely on human heuristics (SEC). We see similar results with llama3.2-3b-instruct, as shown in Tab. 4.

$\lvert\mathcal{X}\rvert$	BENCHMARK	METHOD					IMPROVEMENT	
		π_{ref}	UNIFORM	SEC	PCL	AC (OURS)	$+\Delta$	$+\Delta\%$
30,000	COUNTDOWN	0.00	44.74	58.87	57.24	**62.12**	+3.25	+5.52
	COUNTDOWN-HARD	0.00	41.00	51.50	48.00	**58.00**	+6.50	+12.62
30,000	ZEBRA	0.00	35.12	36.00	34.12	**37.62**	+1.62	+4.50
	ZEBRA-HARD	0.00	30.50	27.50	26.00	**34.50**	+4.00	+13.11
30,000	ARC-1D	0.00	26.74	27.87	26.37	**36.37**	+8.50	+30.51
	ARC-HARD	0.00	19.50	18.50	18.50	**31.00**	+11.50	+58.97
12,000	MATH500	61.80	**83.00**	81.00	79.79	81.00	-2.00	-2.41
	AIME24	3.33	23.33	20.00	23.33	**30.00**	+6.67	+28.57

4 EXPERIMENTAL RESULTS

Benchmarks. We evaluate ACTOR-CURATOR on five reasoning and mathematics benchmarks. **Countdown** is an arithmetic puzzle requiring the composition of integers and operations to reach a target value (Stojanovski et al., 2025). **Zebra** is a symbolic logic puzzle that requires finding assignments satisfying a set of constraints (Stojanovski et al., 2025). **ARC-1D** is the one-dimensional variant of the Abstraction and Reasoning Corpus, designed to test inductive generalization (Chollet, 2019; Xu et al., 2023). **MATH500** consists of competition-level mathematics problems (Hendrycks et al., 2021). **AIME24** contains problems from the 2024 American Invitational Mathematics Examination. We additionally consider hard subsets (**countdown-hard**, **zebra-hard**, **arc-hard**) for validation. We train on **30K** problems for Countdown, Zebra, and ARC-1D, and **12K** MATH problems for MATH500 and AIME24.

Experimental setup. We implement our post-training pipeline using VERL (Sheng et al., 2024). The curator is initialized from a pretrained Qwen3-0.6B model (Yang et al., 2025). Unless otherwise specified, the actor is trained using GSPO, a stabilized variant of GRPO (Ahmadian et al., 2024), on Qwen2.5-3B. Additional details are provided in App. G, including hyper-parameters.

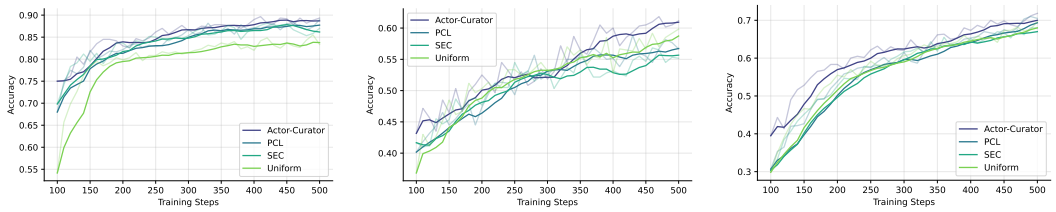
4.1 MAIN RESULTS

Baselines. We compare against state-of-the-art curriculum learning methods for LLM post-training, using the same backbone model and actor update. π_{ref} denotes the pretrained model without post-training. **Uniform sampling** draws training problems uniformly at random. **SEC** partitions problems into manually defined buckets and updates bucket probabilities based on the sum of absolute advantages (Chen et al., 2025a). **PCL** trains a value model to estimate success probabilities and prioritizes problems with predicted success near 50% (Gao et al., 2025). PCL is competitive with recent curriculum-based approaches (Yue et al., 2025; Zhang et al., 2025; Zheng et al., 2025b).

Performance. We evaluate performance on held-out test sets, recording metrics every 10 training steps. Following prior work (Gao et al., 2025), we report the peak performance achieved within the first 100 steps. Results are summarized in Tab. 1. Across both backbone models and most benchmarks, ACTOR-CURATOR consistently outperforms all baselines, with additional results in App. J. Notably, ACTOR-CURATOR achieves substantially larger gains on harder benchmarks such as **arc-hard** and **AIME24**, indicating that adaptive curation is particularly beneficial in challenging regimes.

Efficiency. As shown in Fig. 4, ACTOR-CURATOR reaches comparable or higher performance using significantly fewer training steps than uniform sampling, demonstrating improved sample efficiency.

Training dynamics. Across datasets, ACTOR-CURATOR exhibits more stable optimization and often continues to improve performance after baselines plateau, as shown in Figs. 3, 9 and 10.



(a) Countdown training curves (b) Zebra training curves (c) ARC-1D training curves

Figure 3: **Training dynamics.** ACTOR-CURATOR Test performance over training on three datasets, showing faster convergence and higher final accuracy.

4.2 ABLATIONS

Core components. We ablate two key design choices: the policy-improvement utility and the OSMD bandit objective. **Absolute adv** replaces the policy-improvement signal with mean absolute advantage, as in (Chen et al., 2025a). **Regression loss** trains the curator to predict the target utility value using a squared loss, rather than learning a classifier as in OSMD. The predicted utilities are then converted into a sampling distribution via a Boltzmann transform with temperature η, matching the temperature used in OSMD. As shown in Fig. 7b, both components are critical for achieving strong performance. We include a more detailed account of the motivation behind Absolute Adv. and regression in App. A, as well as their difference with ACTOR-CURATOR.

Actor update methods. ACTOR-CURATOR generalizes across actor optimization algorithms. In Fig. 7a, ACTOR-CURATOR significantly improves GRPO-based training relative to uniform sampling, demonstrating robustness to the choice of actor update.

Additional ablation. We provide additional ablation on curator model size (Fig. 7c) and candidate batch size (Fig. 8) in App. I.

4.3 INTERPRETATION AND ANALYSIS

Curriculum progression. As shown in Fig. 5, ACTOR-CURATOR initially prioritizes easier problems and gradually shifts toward harder ones over training. Problem difficulty is estimated using heuristic annotations.

Impact on actor updates. ACTOR-CURATOR induces consistently larger actor gradient norms than uniform sampling (Fig. 6), suggesting that curated problems produce more informative learning signals. This aligns with the curator's objective of prioritizing problems with higher expected policy improvement.

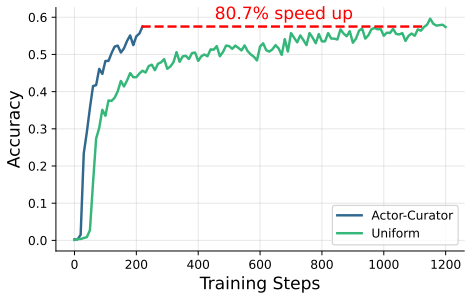


Figure 4: **Training speed-up.** ACTOR-CURATOR attains high test accuracy with significantly fewer steps.

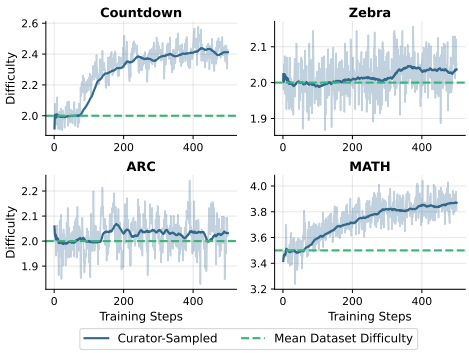


Figure 5: **Difficulty progression of curated problems.** ACTOR-CURATOR gradually increases the average difficulty over training.

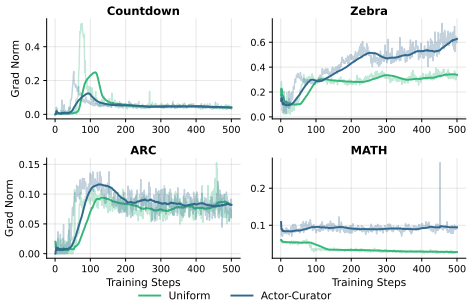


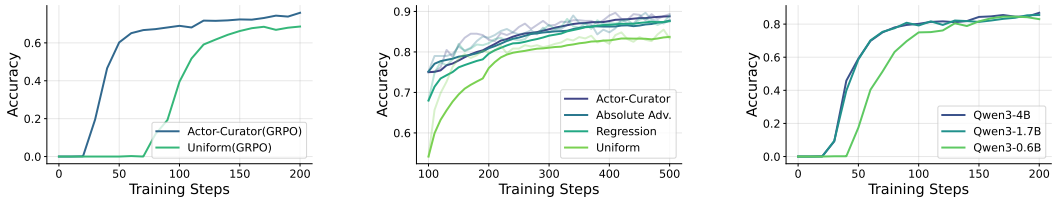
Figure 6: **Actor gradient norms over training.** ACTOR-CURATOR yields larger and more sustained updates.

5 RELATED WORK

RLVR. Reinforcement learning with verification (RLVR) has emerged as an effective paradigm for improving the capabilities of large language models (LLMs) during post-training (Guo et al., 2025; Setlur et al., 2024; Chen et al., 2025b; Wen et al., 2025). Prior work has largely focused on improving the actor update rules (Yu et al., 2025; Dong et al., 2025) or enhancing trajectory generation, often via search-based methods (Zhang et al., 2024; Light et al., 2025b;c). Our work is complementary: rather than modifying the actor or rollout process, we focus on learning a principled curriculum that selects which problems the actor should train on to maximize policy improvement.

Curriculum learning for RL. Classical curriculum learning approaches in reinforcement learning typically select tasks of intermediate difficulty, often defined via the agent's probability of success (Florensa et al., 2017; 2018; Wöhlke et al., 2020; Liu et al., 2025). For example, ProCuRL selects problems whose difficulty is estimated to be near a decision boundary using a learned value network (Tzannetos et al., 2023). In contrast, our approach (1) trains a large language model as a curator that directly operates over language problems, and (2) uses a theoretically grounded, policy-improvement-based target rather than heuristic difficulty estimates. This design allows our method to generalize across different actor update rules and scale to large, heterogeneous datasets.

Curriculum learning for LLMs. Recent work on self-improving and self-evolving LLMs has highlighted the importance of curriculum learning in RL-based post-training (Ye et al., 2024; Light et al., 2025a). Several methods adjust problem sampling using bandit-style objectives such as UCB (Chen et al., 2025a; Wang et al., 2025; Gao et al., 2025). However, most existing approaches rely on manual curriculum design, including human-labeled difficulty levels or pre-defined problem buckets that are sampled adaptively (Graves et al., 2017). In contrast, we propose one of the first fully automated curriculum learning frameworks for LLM post-training that requires no human annotations or manual dataset structuring.



(a) Actor update ablation (b) Core component ablation (c) Curator model ablation

Figure 7: **Ablation results.** (a) ACTOR-CURATOR is compatible with alternative actor update methods such as GRPO and yields consistent performance gains. (b) The combination of the OSMD curator objective and the policy-improvement target achieves superior performance compared to alternative targets and loss functions. (c) Varying curator model size leads to similar long-term performance, indicating robustness to curator capabilities.

6 CONCLUSION

Our results demonstrate that combining bandit-style optimization, neural function approximation, and policy-improvement-based feedback provides a powerful and general approach to curriculum learning for RL post-training. This combination enables efficient and stable adaptation of training data selection at scale, leading to faster learning and higher final performance, particularly on difficult reasoning problems. Importantly, these gains persist across diverse benchmarks and settings, suggesting that this design generalizes beyond specific tasks or datasets and offers a scalable foundation for improving reinforcement learning post-training of large language models.

Beyond empirical gains, ACTOR-CURATOR highlights a broader shift in how post-training systems should be designed. Rather than treating training data as a static resource, our results suggest that data selection itself can be optimized online as part of the learning process, adapting in tandem with the evolving policy. This perspective opens the door to post-training pipelines that are less reliant on meticulous dataset engineering and more resilient to distributional mismatch, noise, and continual data growth.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.

Minoru Asada, Shoichi Noda, Sukoya Tawaratsumida, and Koh Hosoda. Purposive behavior acquisition for a real robot by vision-based reinforcement learning. *Machine learning*, 23(2):279–303, 1996.

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

Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper_files/paper/2014/file/91ba7292e5388b90b58d0b839a7f19ec-Paper.pdf.

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

Yang Chen, Zhuolin Yang, Zihan Liu, Chankyu Lee, Peng Xu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Acereason-nemotron: Advancing math and code reasoning through reinforcement learning. *arXiv preprint arXiv:2505.16400*, 2025b.

François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, et al. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*, 2025.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pp. 482–495. PMLR, 2017.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pp. 1515–1528. PMLR, 2018.

Zhaolin Gao, Joongwon Kim, Wen Sun, Thorsten Joachims, Sid Wang, Richard Yuanzhe Pang, and Liang Tan. Prompt curriculum learning for efficient llm post-training. *arXiv preprint arXiv:2510.01135*, 2025.

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *international conference on machine learning*, pp. 1311–1320. Pmlr, 2017.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the nineteenth international conference on machine learning*, pp. 267–274, 2002.

Tor Lattimore and Csaba Szepesvari. Bandit algorithms. 2017. URL https://tor-lattimore.com/downloads/book/book.pdf.

Jonathan Light, Min Cai, Weiqin Chen, Guanzhi Wang, Xiusi Chen, Wei Cheng, Yisong Yue, and Ziniu Hu. Strategist: Self-improvement of llm decision making via bi-level tree search. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Jonathan Light, Wei Cheng, Benjamin Riviere, Wu Yue, Masafumi Oyamada, Mengdi Wang, Yisong Yue, Santiago Paternain, and Haifeng Chen. Disc: Dynamic decomposition improves llm inference scaling. In *Advances in Neural Information Processing Systems*, 2025b.

Jonathan Light, Yue Wu, Yiyou Sun, Wenchao Yu, Yanchi Liu, Xujiang Zhao, Ziniu Hu, Haifeng Chen, and Wei Cheng. Sfs: Smarter code space search improves llm inference scaling. In *The Thirteenth International Conference on Learning Representations*, 2025c.

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

Shubham Parashar, Shurui Gui, Xiner Li, Hongyi Ling, Sushil Vemuri, Blake Olson, Eric Li, Yu Zhang, James Caverlee, Dileep Kalathil, and Shuiwang Ji. Curriculum reinforcement learning from easy to hard tasks improves llm reasoning, 2025. URL `https://arxiv.org/abs/2506.06632`.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

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

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

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. Reasoning gym: Reasoning environments for reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2505.24760*, 2025.

Georgios Tzannetos, Bárbara Gomes Ribeiro, Parameswaran Kamalaruban, and Adish Singla. Proximal curriculum for reinforcement learning agents. *arXiv preprint arXiv:2304.12877*, 2023.

Zhenting Wang, Guofeng Cui, Yu-Jhe Li, Kun Wan, and Wentian Zhao. Dump: Automated distribution-level curriculum learning for rl-based llm post-training. *arXiv preprint arXiv:2504.09710*, 2025.

Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Tanglifu Tanglifu, Xiaowei Lv, et al. Light-r1: Curriculum sft, dpo and rl for long cot from scratch and beyond. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pp. 318–327, 2025.

Jan Wöhlke, Felix Schmitt, and Herke van Hoof. A performance-based start state curriculum framework for reinforcement learning. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1503–1511, 2020.

Yuxin Wu and Yuandong Tian. Training agent for first-person shooter game with actor-critic curriculum learning. In *International Conference on Learning Representations*, 2017.

Yudong Xu, Wenhao Li, Pashootan Vaezipoor, Scott Sanner, and Elias B Khalil. Llms and the abstraction and reasoning corpus: Successes, failures, and the importance of object-based representations. *arXiv preprint arXiv:2305.18354*, 2023.

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

Ziyu Ye, Rishabh Agarwal, Tianqi Liu, Rishabh Joshi, Sarmishta Velury, Quoc V Le, Qijun Tan, and Yuan Liu. Scalable reinforcement post-training beyond static human prompts: Evolving alignment via asymmetric self-play. *arXiv preprint arXiv:2411.00062*, 2024.

Gaurav Yengera, Rati Devidze, Parameswaran Kamalaruban, and Adish Singla. Curriculum design for teaching via demonstrations: theory and applications. *Advances in Neural Information Processing Systems*, 34:10496–10509, 2021.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772, 2024.

Ruiqi Zhang, Daman Arora, Song Mei, and Andrea Zanette. Speed-rl: Faster training of reasoning models via online curriculum learning. *arXiv preprint arXiv:2506.09016*, 2025.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025a.

Haizhong Zheng, Yang Zhou, Brian R Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for llm reasoning via selective rollouts. *arXiv preprint arXiv:2506.02177*, 2025b.

A BASELINES

A.1 GROUP-BASED POLICY OPTIMIZATION

Group-based policy optimization methods update the policy by grouping multiple rollouts from the same problem and computing advantages relative to the group baseline.

GRPO. Group Relative Policy Optimization (GRPO) (Shao et al., 2024) computes advantages by comparing each solution's reward to the mean reward of all solutions sampled from the same problem. For a problem x with rollouts $\mathcal{Y}_x = \{y^{(1)}, \ldots, y^{(m)}\}$ and corresponding rewards $\{R(y^{(j)}|x)\}_{j=1}^m$, the advantage for solution $y^{(i)}$ is:

$$A^{\text{GRPO}}(y^{(i)}|x) = R(y^{(i)}|x) - \frac{1}{m}\sum_{j=1}^{m} R(y^{(j)}|x). \tag{13}$$

The policy is then updated using a PPO-style clipped objective:

$$\mathcal{L}^{\text{GRPO}}(\pi) = \mathbb{E}_{x,y^{(i)}\sim\mathcal{Y}_x}\left[\min\left(\rho_i A^{\text{GRPO}}(y^{(i)}|x), \text{clip}(\rho_i, 1-\epsilon, 1+\epsilon)A^{\text{GRPO}}(y^{(i)}|x)\right)\right], \tag{14}$$

where $\rho_i = \frac{\pi(y^{(i)}|x)}{\pi_{\text{old}}(y^{(i)}|x)}$ is the importance ratio and ϵ is the clipping threshold.

GSPO. Group-Sequence Policy Optimization (Zheng et al., 2025a) addresses fundamental stability issues in GRPO by defining importance ratios at the sequence level rather than the token level. Unlike GRPO, which applies token-level importance weights that can introduce high-variance noise, GSPO computes importance ratios based on sequence likelihood, aligning with the principle of importance sampling.

GSPO optimizes the following sequence-level objective:

$$\mathcal{J}^{\text{GSPO}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\{y_i\}_{i=1}^G\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\left[\frac{1}{G}\sum_{i=1}^{G}\min\left(s_i(\theta)\hat{A}_i, \text{clip}(s_i(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_i\right)\right], \tag{15}$$

where the sequence-level importance ratio is defined as:

$$s_i(\theta) = \left(\frac{\pi_\theta(y_i|x)}{\pi_{\theta_{\text{old}}}(y_i|x)}\right)^{\frac{1}{|y_i|}} = \exp\left(\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\log\frac{\pi_\theta(y_{i,t}|x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t}|x, y_{i,<t})}\right). \tag{16}$$

The advantage computation remains the same as in GRPO.

A.2 MEAN ABSOLUTE ADVANTAGE AS CURRICULUM REWARD

An ideal curriculum should prioritize training problems that maximize the model's immediate learning outcomes. A natural way to quantify learning outcomes is through the magnitude of parameter updates induced by the selected training data, which can be approximated by the absolute advantage.

In the common setting of RL with verifiable binary rewards, mean absolute advantage has a attractive interpretation. When using group-based RL methods like GRPO with n rollouts per problem, the advantage for the i-th rollout is computed as:

$$\hat{A}_{t,i} = \frac{r_i - \text{mean}(r)}{\text{std}(r)} = \frac{r_i - p}{\sqrt{p(1-p)}}, \tag{17}$$

where p is the empirical success rate over the group. Since the reward r_i follows a Bernoulli distribution, the expected absolute advantage is:

$$\mathbb{E}[|\hat{A}_{t,i}|] = p\cdot\frac{1-p}{\sqrt{p(1-p)}} + (1-p)\cdot\frac{p}{\sqrt{p(1-p)}} = 2\sqrt{p(1-p)}. \tag{18}$$

The function $g(p) = 2\sqrt{p(1-p)}$ is symmetric around $p = 0.5$, strictly concave on $[0, 1]$, and reaches its maximum at $p = 0.5$. Thus, maximizing expected absolute advantage is equivalent to prioritizing problems at a success rate of 50%.

A.3 VALUE-BASED CURATION

Value-based curation methods maintain explicit estimates of the utility or learning value associated with training data, which are then used to guide adaptive curriculum selection during RL post-training.

The core principle of value-based curation is to learn a utility function Q_t that maps training data to expected learning outcomes at training step t. Let $\mathcal{S}$ denote the space over which utilities are estimated (e.g., individual problems, problem categories, or problem features). The utility function $Q_t : \mathcal{S} \to \mathbb{R}$ assigns a scalar value to each element $s \in \mathcal{S}$.

During curriculum selection, the curator samples from $\mathcal{S}$ according to a policy derived from Q_t. A common choice is the Boltzmann (softmax) policy:

$$p_t(s) = \frac{\exp(Q_t(s)/\eta)}{\sum_{s' \in \mathcal{S}} \exp(Q_t(s')/\eta)}, \tag{19}$$

where $\eta > 0$ is a temperature parameter that controls the exploration-exploitation tradeoff: higher temperatures lead to more uniform sampling (exploration), while lower temperatures concentrate probability mass on high-utility items (exploitation).

The utility function is updated over time based on observed learning outcomes. After selecting data according to p_t and performing an RL update, the curator observes a reward signal $r_t(s)$ that measures the actual learning value obtained. These observations are used to refine Q_{t+1}.

SEC. Self-Evolving Curriculum (SEC) (Chen et al., 2025a) instantiates value-based curation by defining the utility space $\mathcal{S} = C$ as a discrete set of problem categories (e.g., difficulty levels, problem types). The utility function $Q_t : C \to \mathbb{R}$ is represented as a lookup table, with one scalar value per category.

SEC uses temporal difference (TD) learning to update utilities:

$$Q_{t+1}(c) = \alpha r_t(c) + (1 - \alpha) Q_t(c), \tag{20}$$

where $\alpha \in (0, 1]$ is a learning rate and $r_t(c)$ is the mean absolute advantage aggregated over all problems from category c selected at step t. This exponential moving average naturally adapts to non-stationarity as category utilities change with model improvement.

Curriculum selection proceeds in two stages: first, a category is sampled according to the Boltzmann policy $p_t(c) \propto \exp(Q_t(c)/\eta)$; then, problems are uniformly sampled from the selected category.

Regression. An alternative instantiation defines the utility space $\mathcal{S} = \mathcal{X}$ directly at the problem level, estimating utilities for individual training problems. The utility function is parameterized by a neural network $Q_\phi : \mathcal{X} \to \mathbb{R}$ that takes problem representations (e.g., text embeddings) as input.

Rather than incremental TD updates, the curator is trained via supervised regression. Let $\mathcal{H}_t = \{(x_j, r_j)\}_{j=1}^{t}$ denote the history of problem-reward observations. The utility network is optimized to minimize mean squared error:

$$\mathcal{L}_{\text{MSE}}(\phi) = \sum_{(x_j, r_j) \in \mathcal{H}_t} \left(Q_\phi(x_j) - r_j\right)^2. \tag{21}$$

This can be optimized periodically (every K steps) via batch gradient descent, or online via stochastic gradient descent:

$$\phi_{t+1} = \phi_t - \beta \nabla_\phi \left(Q_\phi(x_t) - r_t(x_t)\right)^2, \tag{22}$$

where β is the learning rate. Problems are sampled directly according to $p_t(x) \propto \exp(Q_\phi(x)/\eta)$.

PCL (Gao et al., 2025) exemplifies the online regression approach. PCL updates Q_ϕ concurrently with policy training using only the current batch of observations. At each step t, PCL samples a candidate pool of km prompts and selects the m prompts whose predicted values are closest to a target threshold τ (typically 0.5):

$$D_m = \arg \min_{S \subseteq D_{km}, |S| = m} \sum_{x \in S} |Q_\phi(x) - \tau|. \tag{23}$$

This greedy selection can be viewed as an extreme limit of Boltzmann sampling with a sharply peaked distribution. Consider the modified utility $\tilde{Q}(x) = -|Q_\phi(x) - \tau|$, which measures negative distance from the threshold. As the temperature $\eta \to 0$, the Boltzmann policy $p(x) \propto \exp(\tilde{Q}(x)/\eta)$ concentrates all probability mass on prompts nearest to τ, recovering PCL's greedy selection. This deterministic selection strategy is computationally efficient and ensures the training batch contains only prompts of target difficulty, maximizing the effective ratio of informative gradient signals.

After selecting prompts, PCL generates n responses per prompt and updates both the policy and value model. The value model is trained on the observed rewards from the selected prompts:

$$\mathcal{L}_{\text{PCL}}(\phi) = \sum_{i=1}^{m} \left(Q_\phi(x_i) - \frac{1}{n} \sum_{j=1}^{n} r(x_i, y_{i,j}) \right)^2 . \tag{24}$$

Comparison to Actor-Curator. The Actor-Curator framework with OSMD differs fundamentally from value-based curation methods in its optimization objective and data selection mechanism. While value-based methods learn utilities $Q_t(s)$ to predict expected learning outcomes and sample accordingly, OSMD directly optimizes a curriculum distribution $\boldsymbol{p}_t$ to maximize expected policy improvement. Value-based approaches require estimating problem-level or category-level values and making selection decisions based on these estimates—a two-stage process that introduces approximation error. In contrast, OSMD treats curriculum optimization as a first-order problem: the gradient $\nabla_q J(\pi_t, \boldsymbol{p})$ directly specifies how to adjust the data distribution to improve the policy, without requiring intermediate value estimates. Furthermore, value-based methods typically rely on scalar reward signals $r_t(s)$ to update utilities, whereas OSMD leverages the full gradient information $\nabla_\theta \mathcal{L}(\pi_t; x)$ to measure the learning value of each problem. This allows OSMD to capture richer information about how individual problems affect policy optimization, beyond what a single scalar reward can convey.

B POLICY IMPROVEMENT ESTIMATION

This section proves unbiasedness of the per-problem bandit feedback estimators used to train the curator. Recall the per-problem utility at iteration t (Eq. equation 6):

$$u_{\boldsymbol{x}}^t = p_{\mathcal{X}}(\boldsymbol{x}) \mathop{\mathbb{E}}_{\boldsymbol{y}\sim\pi^t(\cdot|\boldsymbol{x})} \left[\frac{\pi^{t+1}(\boldsymbol{y} \mid \boldsymbol{x})}{\pi^t(\boldsymbol{y} \mid \boldsymbol{x})} A_{\pi^t}(\boldsymbol{y} \mid \boldsymbol{x}) \right].$$

We also recall the rollout-based estimator (Eq. equation **??**):

$$\hat{A}^t(\cdot \mid \boldsymbol{x}) \triangleq \frac{1}{|\mathcal{Y}_{\boldsymbol{x}}^t|} \sum_{\boldsymbol{y}\in\mathcal{Y}_{\boldsymbol{x}}^t} \frac{\pi^{t+1}(\boldsymbol{y} \mid \boldsymbol{x})}{\pi^t(\boldsymbol{y} \mid \boldsymbol{x})} A_{\pi^t}(\boldsymbol{y} \mid \boldsymbol{x}), \quad \mathcal{Y}_{\boldsymbol{x}}^t = \{\boldsymbol{y}^{(j)} \sim \pi^t(\cdot \mid \boldsymbol{x})\}_{j=1}^{|\mathcal{Y}_{\boldsymbol{x}}|}.$$

B.1 SINGLE-STAGE CASE

Theorem 3 (Unbiasedness (single-stage)) *For the estimator in Eq. equation 7,*

$$\hat{U}_{\boldsymbol{x}}^t = p_{\mathcal{X}}(\boldsymbol{x}) \frac{\mathbb{I}\{\boldsymbol{x} \in \mathcal{X}^t\}}{p^t(\boldsymbol{x})} \hat{A}^t(\cdot \mid \boldsymbol{x}),$$

we have $\mathbb{E}[\hat{U}_{\boldsymbol{x}}^t] = u_{\boldsymbol{x}}^t$ *for every* $\boldsymbol{x} \in \mathcal{X}$.

Proof: Fix an iteration t and a problem $\boldsymbol{x} \in \mathcal{X}$. In the single-stage setting, the training set $\mathcal{X}^t$ is sampled directly from $\mathcal{X}$ according to the curator distribution $p^t(\cdot)$, so that

$$\Pr(\boldsymbol{x} \in \mathcal{X}^t) = p^t(\boldsymbol{x}).$$

First, $\hat{A}^t(\cdot \mid \boldsymbol{x})$ is an unbiased estimator of the population quantity

$$\mathop{\mathbb{E}}_{\boldsymbol{y}\sim\pi^t(\cdot|\boldsymbol{x})} \left[\frac{\pi^{t+1}(\boldsymbol{y} \mid \boldsymbol{x})}{\pi^t(\boldsymbol{y} \mid \boldsymbol{x})} A_{\pi^t}(\boldsymbol{y} \mid \boldsymbol{x}) \right],$$

by i.i.d. rollout sampling and linearity of expectation.

Next, take expectation of $\hat{U}_{\boldsymbol{x}}^t$ conditioning on $\hat{A}^t(\cdot \mid \boldsymbol{x})$:

$$\mathbb{E}\Big[\hat{U}_{\boldsymbol{x}}^t \mid \hat{A}^t(\cdot \mid \boldsymbol{x})\Big] = p_{\mathcal{X}}(\boldsymbol{x})\ \mathbb{E}\left[\frac{\mathbb{I}\{\boldsymbol{x} \in \mathcal{X}^t\}}{p^t(\boldsymbol{x})}\right]\ \hat{A}^t(\cdot \mid \boldsymbol{x}) = p_{\mathcal{X}}(\boldsymbol{x})\, \hat{A}^t(\cdot \mid \boldsymbol{x}),$$

since $\mathbb{E}[\mathbb{I}\{\boldsymbol{x} \in \mathcal{X}^t\}] = \Pr(\boldsymbol{x} \in \mathcal{X}^t) = p^t(\boldsymbol{x})$.

Finally, taking expectation over rollout randomness yields

$$\mathbb{E}[\hat{U}_{\boldsymbol{x}}^t] = p_{\mathcal{X}}(\boldsymbol{x})\ \mathbb{E}\Big[\hat{A}^t(\cdot \mid \boldsymbol{x})\Big] = u_{\boldsymbol{x}}^t,$$

which proves the claim. □

B.2 TWO-STAGE CASE

Theorem 4 (Unbiasedness (two-stage)) *For the two-stage estimator in Eq. equation 10,*

$$\hat{U}_{\mathrm{two},\boldsymbol{x}}^t = p_{\mathcal{X}}(\boldsymbol{x})\ \frac{\mathbb{I}\{\boldsymbol{x} \in \mathcal{X}^t\}}{q(\boldsymbol{x})\, p^t(\boldsymbol{x} \mid \tilde{\mathcal{X}}^t)}\ \hat{A}^t(\cdot \mid \boldsymbol{x}),$$

we have $\mathbb{E}[\hat{U}_{\mathrm{two},\boldsymbol{x}}^t] = u_{\boldsymbol{x}}^t$ *for every* $\boldsymbol{x} \in \mathcal{X}$.

Proof: Fix an iteration t and a problem $\boldsymbol{x} \in \mathcal{X}$. By definition of the two-stage procedure (Section 3.5), the candidate set $\tilde{\mathcal{X}}^t$ is sampled from $\tilde{q}$, inducing the marginal inclusion probability $q(\boldsymbol{x}) = \Pr(\boldsymbol{x} \in \tilde{\mathcal{X}}^t)$. Conditioned on $\tilde{\mathcal{X}}^t$, the curator selects the training set $\mathcal{X}^t \subset \tilde{\mathcal{X}}^t$ according to $p^t(\cdot \mid \tilde{\mathcal{X}}^t)$.

As in the single-stage case, $\hat{A}^t(\cdot \mid \boldsymbol{x})$ is an unbiased estimator of the corresponding population expectation under $\boldsymbol{y} \sim \pi^t(\cdot \mid \boldsymbol{x})$.

Now condition on the realized candidate set $\tilde{\mathcal{X}}^t$ and on $\hat{A}^t(\cdot \mid \boldsymbol{x})$. If $\boldsymbol{x} \notin \tilde{\mathcal{X}}^t$, then $\mathbb{I}\{\boldsymbol{x} \in \mathcal{X}^t\} = 0$ almost surely. If $\boldsymbol{x} \in \tilde{\mathcal{X}}^t$, then by the selection step,

$$\mathbb{E}\Big[\mathbb{I}\{\boldsymbol{x} \in \mathcal{X}^t\} \mid \tilde{\mathcal{X}}^t\Big] = p^t(\boldsymbol{x} \mid \tilde{\mathcal{X}}^t),$$

and therefore

$$\mathbb{E}\left[\frac{\mathbb{I}\{\boldsymbol{x} \in \mathcal{X}^t\}}{p^t(\boldsymbol{x} \mid \tilde{\mathcal{X}}^t)} \;\middle|\; \tilde{\mathcal{X}}^t, \hat{A}^t(\cdot \mid \boldsymbol{x})\right] = \mathbb{I}\{\boldsymbol{x} \in \tilde{\mathcal{X}}^t\}.$$

Substituting into the estimator gives

$$\mathbb{E}\Big[\hat{U}^t_{\mathrm{two},\boldsymbol{x}} \;\Big|\; \tilde{\mathcal{X}}^t, \hat{A}^t(\cdot \mid \boldsymbol{x})\Big] = p_{\mathcal{X}}(\boldsymbol{x})\, \frac{\mathbb{I}\{\boldsymbol{x} \in \tilde{\mathcal{X}}^t\}}{q(\boldsymbol{x})}\; \hat{A}^t(\cdot \mid \boldsymbol{x}).$$

Taking expectation over the proposal step yields

$$\mathbb{E}\Big[\hat{U}^t_{\mathrm{two},\boldsymbol{x}} \;\Big|\; \hat{A}^t(\cdot \mid \boldsymbol{x})\Big] = p_{\mathcal{X}}(\boldsymbol{x})\, \frac{\mathbb{E}[\mathbb{I}\{\boldsymbol{x} \in \tilde{\mathcal{X}}^t\}]}{q(\boldsymbol{x})}\; \hat{A}^t(\cdot \mid \boldsymbol{x}) = p_{\mathcal{X}}(\boldsymbol{x})\, \hat{A}^t(\cdot \mid \boldsymbol{x}),$$

since $\mathbb{E}[\mathbb{I}\{\boldsymbol{x} \in \tilde{\mathcal{X}}^t\}] = \Pr(\boldsymbol{x} \in \tilde{\mathcal{X}}^t) = q(\boldsymbol{x})$. Finally, taking expectation over rollout randomness gives

$$\mathbb{E}[\hat{U}^t_{\mathrm{two},\boldsymbol{x}}] = p_{\mathcal{X}}(\boldsymbol{x})\; \mathbb{E}[\hat{A}^t(\cdot \mid \boldsymbol{x})] = u^t_{\boldsymbol{x}},$$

which proves the claim. □

C IDEALIZED OSMD ALGORITHM

Algorithm 2 Sleeping Online Mirror Descent

Require: Number of total arms K, number of available arms k each round, horizon T, step size $\eta > 0$, exploration parameter $\alpha \in (0, 1/k)$.

1: Initialize $\boldsymbol{p}_1 = (1/K, \ldots, 1/K)$.
2: **for** $t = 1, 2, \ldots, T$ **do**
3: Sample available subset $\tilde{\mathcal{X}}_t$. Compute

$$\boldsymbol{p}^t(i \mid \tilde{\mathcal{X}}_t) = \begin{cases} \frac{\boldsymbol{p}_{t,i}}{\sum_{j \in \tilde{\mathcal{X}}_t} \boldsymbol{p}_{t,j}} & \text{if } i \in \tilde{\mathcal{X}}_t \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

4: Sample $a_{t,1}, a_{t,2}, \ldots, a_{t,s} \overset{i.i.d.}{\sim} \boldsymbol{p}^t(\cdot \mid \tilde{\mathcal{X}}_t)$
5: **(Loss Estimator)** For each arm $i \in [K]$, set

$$\widehat{L}_{t,i} = \frac{1}{s} \sum_{r=1}^{s} \mathbf{1}\{a_{t,r} = i\} \frac{l_{t,i}}{\boldsymbol{p}^t(i \mid \tilde{\mathcal{X}}_t)} \tag{26}$$

6: **(OSMD Update)** Update the next distribution by the mirror step

$$\begin{aligned} &\boldsymbol{p}_{t+1} \in \arg\min_{p \in \mathcal{A}} \left\{ \eta \langle p, \widehat{\boldsymbol{L}}_t \rangle + D_F(p, \boldsymbol{p}_t) \right\}, \\ &\text{where} \quad D_F(\boldsymbol{u}, \boldsymbol{v}) = \sum_i u_i \log \frac{u_i}{v_i} \end{aligned} \tag{27}$$

7: **end for**

C.1 SETUP

We formalize the tabular bandit algorithm in Section 3 as Algorithm 2. Under this idealized algorithm. We assume there is a large set of K arms. At each time step t, k arms are uniformly randomly chosen as candidate arms. The settings allows for the pulling of s arms per round, and after each round the loss $l_{t,i}$ is revealed for each chosen arm i, where i indicates the arm's original index in $[K]$. Throughout this section and the next, we denote the vectorized quantities with bold font. For example $\boldsymbol{l}_t = (l_{t,1}, \ldots, l_{t,K})$.

Algorithm 2 departs from traditional Online Stochastic Mirror Descent (OSMD) due to the availability constraint. The action distribution is conditioned on a randomly sampled *available set* at each round, and the loss estimator is modified to remain unbiased under this conditional sampling.

Eqn. equation 25 introduces a two-stage sampling process. First, a random k-subset $\tilde{\mathcal{X}}_t \subseteq [K]$ of available arms is drawn uniformly. The learner then constructs a *conditional distribution*

$$\boldsymbol{p}^t(i \mid \tilde{\mathcal{X}}_t) = \frac{\boldsymbol{p}_{t,i}}{\sum_{j \in \tilde{\mathcal{X}}_t} \boldsymbol{p}_{t,j}} \quad \text{for } i \in \tilde{\mathcal{X}}_t,$$

and assigns zero probability to arms outside $\tilde{\mathcal{X}}_t$. We sometimes also use $\boldsymbol{p}_{|\tilde{\mathcal{X}}_t}(i)$ to denote $\boldsymbol{p}^t(i \mid \tilde{\mathcal{X}}_t)$. This renormalization ensures that the learner only samples from arms that are available at round t, while still using $\boldsymbol{p}_t$ as the global state variable that is updated over time.

The conditional sampling in Eqn. equation 25 invalidates the standard OSMD estimator, since $p_{t,i}$ is no longer the actual probability with which arm i is sampled. Eqn. equation 26 addresses this by defining

$$\widehat{L}_{t,i} = \frac{1}{s} \sum_{r=1}^{s} \mathbf{1}\{a_{t,r} = i\} \frac{l_{t,i}}{\boldsymbol{p}^t(i \mid \tilde{\mathcal{X}}_t)}.$$

This estimator uses the conditional probability $\boldsymbol{p}^t(i \mid \tilde{\mathcal{X}}_t)$ in the denominator, which is the true sampling probability of arm i given the realized availability set. As a result, conditional on $\tilde{\mathcal{X}}_t$, the

estimator is unbiased:

$$\mathbb{E}\Big[\widehat{L}_{t,i} \mid \tilde{\mathcal{X}}_t\Big] = l_{t,i}\,\mathbb{I}\{i \in \tilde{\mathcal{X}}_t\}.$$

The use of s independent samples further reduces variance and corresponds to a semi-bandit feedback model, but does not change the role of the estimator in the mirror update.

D Regret Analysis

This section presents a proof of the regret bound (Theorem 2) for Algorithm 2. Suppose there are K base arms $[K] = \{1, \ldots, K\}$ each corresponding to a problem in the dataset $\mathcal{X} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(K)}\}$. At each step, a subset $\tilde{\mathcal{X}}_t$ of size k is drawn uniformly randomly from $\mathcal{X}$. For every round t, The curator picks s arms $a_{t,1}, a_{t,2}, \ldots, a_{t,s} \in [K]$ from $\tilde{\mathcal{X}}_t$ and the loss for each arm $l_{t,a_{t,i}}$ is revealed. For convenience, write $\boldsymbol{l}_t = (l_{t,1}, \ldots, l_{t,K})$ to be the losses of each arm as a vector. Assume without loss of generality that $l_{t,i} \in [0, 1]$ for all t and i. Define the subset-masked loss vector $\boldsymbol{l}_t^{\tilde{\mathcal{X}}_t} \in \mathbb{R}^K$:

$$\boldsymbol{l}_t^{\tilde{\mathcal{X}}_t}(i) = \begin{cases} \boldsymbol{l}_t(i) & \text{if } i \in \tilde{\mathcal{X}}_t \\ 0 & \text{otherwise} \end{cases} \tag{28}$$

Denote the best available arm at time t as m_t

$$m_t = \arg\min_{i \in \tilde{\mathcal{X}}_t} l_{t,i} \text{ and } l_t^* = l_{t,m_t}. \tag{29}$$

We define the *best-arm regret* to be

$$\mathrm{Reg}_T^{\mathrm{best}} = \mathbb{E}\left[\sum_{t=1}^{T}(\sum_{i=1}^{s} l_{t,a_{t,s}} - l_t^*)\right] \tag{30}$$

Note that the regret can be expressed in the vectorized form.

$$\mathrm{Reg}_n^{\mathrm{best}} = \mathbb{E}\left[\sum_{t=1}^{T} \left\langle \boldsymbol{p}_{\tilde{\mathcal{X}}_t}, \boldsymbol{l}_t \right\rangle - \langle \boldsymbol{e}_{m_t}, \boldsymbol{l}_t \rangle\right], \tag{31}$$

where $\boldsymbol{e}_i$ is a one-hot vector with 1 on index i. Here, we use $\boldsymbol{p}_{|\tilde{\mathcal{X}}_t}$ to represent the vector $(\boldsymbol{p}_{|\tilde{\mathcal{X}}_t}\boldsymbol{x}^{(1)}), \boldsymbol{p}_{|\tilde{\mathcal{X}}_t}(\boldsymbol{x}^{(2)}), \ldots, \boldsymbol{p}_{|\tilde{\mathcal{X}}_t}(\boldsymbol{x}^{(K)}))$. In practice, forcing uniform exploration usually have negligible or even positive effect on the performance. However, it incurs linear regret for theoretical analysis. Thus, to focus on how OSMD manages losses in a non-stationary environment, this regret analysis isolate the loss attributable to factors other than uniform exploration. Towards this end, we set the comparator $\boldsymbol{q}_t$ to be a mixture of the best available arm and uniform distribution over all arms.

$$\boldsymbol{q}_t = (1 - k\alpha)\boldsymbol{e}_{m_t} + \alpha \mathbf{1}_{\tilde{\mathcal{X}}_t} \tag{32}$$

where $\alpha > 0$ and $\mathbf{1}_{\tilde{\mathcal{X}}_t} \in \mathbb{R}^K$ is a binary vector with $\mathbf{1}_{\tilde{\mathcal{X}}_t}(i) = 1$ for all $i \in \tilde{\mathcal{X}}_t$ and zero everywhere else. The *best-available regret* is defined as

$$\mathrm{Reg}_T^{\mathrm{BA}} = \mathbb{E}\left[\sum_{t=1}^{T} \left\langle \boldsymbol{p}_{|\tilde{\mathcal{X}}_t}, \boldsymbol{l}_t \right\rangle - \langle \boldsymbol{q}_t, \boldsymbol{l}_t \rangle\right] \tag{33}$$

Theorem 5 (Restatement of Theorem 2) *Let the drift parameter*

$$V_T = \sum_{t=2}^{T} \Delta_t, \quad \Delta_t = \max_{i \in [K]} |l_{t,i} - l_{t-1,i}| \tag{34}$$

be a measure of how rapid the arm values changes over time t. Assume without loss of generality that $\boldsymbol{l}_{t,i} \in [0, 1]$ for all t and i. Then we have

$$\mathrm{Reg}_n^{\mathrm{BA}} \leq O\left(T^{2/3} V_n^{1/3}\right) \tag{35}$$

for Algorithm 2.

To bound the smoothed regret, the proof follows a block decomposition argument. We partition the horizon into contiguous blocks of length B and restart the algorithm at the beginning of each block. Within a single block, the loss sequence is treated as approximately stationary, allowing us to compare the algorithm against a fixed comparator using standard OSMD analysis. The regret within each block is controlled by a stability–variance tradeoff: the mirror descent inequality yields a term of order $O(\log(1/\alpha)/\eta)$, while the variance of the importance-weighted estimator contributes a term proportional to $O(\eta B)$. Across blocks, non-stationarity is captured through a variation budget V_n, which upper bounds the cumulative discrepancy between the true per-round losses and the frozen losses used in the blockwise analysis, contributing an additive term of order BV_n. The block length B is then optimized to balance the statistical cost of restarting too frequently against the bias induced by treating losses as stationary within each block. This optimization yields a regret bound scaling as $O(V_n^{1/3})$, reflecting the classical tradeoff between adaptivity to non-stationarity and estimation error Besbes et al. (2014).

Let the length of each aforementioned block be L. Denote the start time of each block as $\tau_l = (l-1)L+1$, where $l = 1, 2, \ldots, B$ and $B = T/L$[1]. Denote the time steps of the l-th block as $\mathcal{I}_l = \{\tau_l, \ldots, \tau_l + L - 1\}$.

$$\mathrm{Reg}_n^{\mathrm{BA}} = \mathbb{E}\left[\sum_{t=1}^{n}\left\langle \boldsymbol{p}_{|\tilde{\mathcal{X}}_t}, \boldsymbol{l}_t\right\rangle - \langle \boldsymbol{q}_t, \boldsymbol{l}_t\rangle\right] = \sum_{l=1}^{B}\mathbb{E}\left[\sum_{t\in\mathcal{I}_l}\left\langle \boldsymbol{p}_{|\tilde{\mathcal{X}}_t}, \boldsymbol{y}_t\right\rangle - \langle \boldsymbol{q}_t, \boldsymbol{l}_t\rangle\right] \tag{36}$$

D.1 Time-Frozen Comparator

This part of the proof introduces a stable reference for comparison in the presence of non-stationary losses. Rather than comparing the learner to the best action at every round, which may change arbitrarily over time, we define a comparator based on the losses at a fixed reference τ. This "time-frozen" comparator serves as a proxy for the per-round best action. The construction allows us to relate the learner's loss to this frozen benchmark, while the error incurred by freezing time can be bounded by the amount of variation in the losses.

Define $m_t^\tau \in \arg\min_{i\in\tilde{\mathcal{X}}_t} y_{\tau,i}$ to be the τ-frozen best arm of time t. Denote the value associated with this arm $(y_t^*)^\tau = y_{\tau,m_t^\tau}$. Thus, we can define the time-frozen smooth comparator

$$\boldsymbol{q}_t^\tau = (1-k\alpha)\boldsymbol{e}_{m_t^\tau} + \alpha\mathbf{1}_{\tilde{\mathcal{X}}_t} \tag{37}$$

Lemma 1 *Let $\mathcal{I} = \{\tau, \ldots, \tau+L-1\}$ be an arbitrary block. Suppose $\alpha > 0$. Let V_n be as defined in Theorem 5. For any sequence of random variables $\{X_t\}_{t\in\mathcal{I}}$ with a common support, we have*

$$\mathbb{E}\left[\sum_{t\in\mathcal{I}} X_t - \left\langle \boldsymbol{q}_t, \frac{\boldsymbol{l}_t^{\tilde{\mathcal{X}}_t}}{Z_t}\right\rangle\right] \le \mathbb{E}\left[\sum_{t\in\mathcal{I}} X_t - \left\langle \boldsymbol{q}_t^\tau, \frac{\boldsymbol{l}_t^{\tilde{\mathcal{X}}_t}}{Z_t}\right\rangle\right] + \frac{1-k\alpha}{k\alpha}\sum_{t\in\mathcal{I}}\|\boldsymbol{l}_t - \boldsymbol{l}_\tau\|_1 \tag{38}$$

Proof: By the definition of $\boldsymbol{q}_t^\tau$, we have

$$\begin{aligned}\left\langle \boldsymbol{q}_t^\tau, \boldsymbol{l}_\tau^{\tilde{\mathcal{X}}_t}\right\rangle &\le (1-k\alpha)(y_t^*)^\tau + \alpha\sum_{j\in\tilde{\mathcal{X}}_t}\boldsymbol{l}_{\tau,j} \\ &\le (1-k\alpha(y_t^*)^\tau + \alpha(k(y_t^*)^\tau + k - 1) \\ &= m_t^\tau + \alpha(k-1)\end{aligned} \tag{39}$$

Rearranging (39) and divide by Z_t, we get

$$\frac{m_t^\tau}{Z_t} \ge \left\langle \boldsymbol{q}_t^\tau, \frac{\boldsymbol{l}_\tau^{\tilde{\mathcal{X}}_t}}{Z_t}\right\rangle - \frac{\alpha(k-1)}{Z_t} \ge \left\langle \boldsymbol{q}_t^\tau, \frac{\boldsymbol{l}_\tau^{\tilde{\mathcal{X}}_t}}{Z_t}\right\rangle - \frac{k-1}{k}. \tag{40}$$

Using the fact that $\min$ is a Lipschitz operation, we have

$$|y_t^* - (y_t^*)^\tau| = \left|\min_{i\in\tilde{\mathcal{X}}_t}\boldsymbol{l}_{t,i} - \min_{i\in\tilde{\mathcal{X}}_t}\boldsymbol{l}_{\tau,i}\right| \le \|\boldsymbol{l}_t - \boldsymbol{l}_\tau\|_\infty \le \|\boldsymbol{l}_t - \boldsymbol{l}_\tau\|_1,$$

[1]For convenience, we assume T is divisible by B and L at the same time.

Applying this identity, we have

$$\begin{aligned}\left|\left\langle \boldsymbol{q}_t, \frac{\boldsymbol{l}_t^{\tilde{\mathcal{X}}_t}}{Z_t}\right\rangle - \left\langle \boldsymbol{q}_t^\tau, \frac{\boldsymbol{l}_t^{\tilde{\mathcal{X}}_t}}{Z_t}\right\rangle\right| &\le \frac{1}{k\alpha}\left|\left\langle \boldsymbol{q}_t, \boldsymbol{l}_t^{\tilde{\mathcal{X}}_t}\right\rangle - \left\langle \boldsymbol{q}_t^\tau, \boldsymbol{l}_t^{\tilde{\mathcal{X}}_t}\right\rangle\right| \\ &= \frac{1-k\alpha}{k\alpha}\left|y_t^* - (y_t^*)^\tau\right| \\ &\le \frac{1-k\alpha}{k\alpha}\|\mathbf{y}_t - \mathbf{y}_\tau\|_1\end{aligned} \tag{41}$$

Rearrange (41) and add X_t to both sides while taking expectation, we get

$$\mathbb{E}\left[\sum_{t\in\mathcal{I}} X_t - \left\langle \boldsymbol{q}_t, \frac{\boldsymbol{l}_t^{\tilde{\mathcal{X}}_t}}{Z_t}\right\rangle\right] \le \mathbb{E}\left[\sum_{t\in\mathcal{I}} X_t - \left\langle \mathbf{q}_t^\tau, \frac{\boldsymbol{l}_t^{\tilde{\mathcal{X}}_t}}{Z_t}\right\rangle\right] + \frac{1-k\alpha}{k\alpha}\sum_{t\in\mathcal{I}}\|\boldsymbol{l}_t - \boldsymbol{l}_\tau\|_1 \tag{42}$$

□

D.2 Moment Lemmas

To apply the standard OMD bound (Theorem 5), we require unbiased loss estimates with controlled variance. Since the learner only observes losses on the sampled subset ($\tilde{\mathcal{X}}_t$), we work with an importance-weighted estimator that accounts for both subset sampling and the normalization induced by Z_t.

The following two lemmas establish the properties needed for the regret analysis. The first shows that the estimator is unbiased for the scaled loss, ensuring that the expected update direction matches the true loss. The second provides a bound on the second moment of the estimator, which controls the variance term in the OSMD regret bound. Together, these results justify the use of the estimator in the mirror descent analysis and quantify the cost introduced by partial observation and uniform exploration.

Lemma 2 (Unbiasedness) *Let $\widehat{L}_{t,i}$ be as defined in Algorithm 2. Denote $\widehat{\boldsymbol{L}}_t = (\widehat{L}_{t,1}, \widehat{L}_{t,2}, \ldots, \widehat{L}_{t,K})$. For every $i \in [K]$,*

$$\mathbb{E}[\widehat{L}_{t,i} \mid \tilde{\mathcal{X}}_t] = \boldsymbol{l}_{t,i}\mathbb{I}\{i \in \tilde{\mathcal{X}}_t\} = \boldsymbol{l}_t^{\tilde{\mathcal{X}}_t}(i) \tag{43}$$

Additionally, define the time-frozen estimator

$$\widehat{L}_{t,i}^\tau := \frac{1}{s}\sum_{r=1}^{s}\mathbb{I}\{a_{t,r} = i\}\frac{l_{\tau,i}}{\boldsymbol{p}^t(i \mid \tilde{\mathcal{X}}_t)}, \tag{44}$$

and we have

$$\mathbb{E}[\widehat{L}_{t,i}^\tau \mid \tilde{\mathcal{X}}_t] = l_{\tau,i}\mathbb{I}\{i \in \tilde{\mathcal{X}}_t\} = \boldsymbol{l}_\tau^{\tilde{\mathcal{X}}_t}(i) \tag{45}$$

Proof: If $i \in \tilde{\mathcal{X}}_t$,

$$\mathbb{E}\left[\widehat{L}_{t,i}^\tau \middle| \tilde{\mathcal{X}}_t\right] = \frac{1}{s}\sum_{r=1}^{s}\boldsymbol{p}_t(i \mid \tilde{\mathcal{X}}_t)\frac{l_{t,i}\mathbb{I}\{i \in \tilde{\mathcal{X}}_t\}}{\boldsymbol{p}^t(i \mid \tilde{\mathcal{X}}_t)} = l_{t,i}\mathbb{I}\{i \in \tilde{\mathcal{X}}_t\}$$

Similarly,

$$\mathbb{E}\left[\widehat{L}_{t,i}^\tau \middle| \tilde{\mathcal{X}}_t\right] = \frac{1}{s}\sum_{r=1}^{s}\boldsymbol{p}_t(i \mid \tilde{\mathcal{X}}_t)\frac{l_{\tau,i}\mathbb{I}\{i \in \tilde{\mathcal{X}}_t\}}{\boldsymbol{p}^t(i \mid \tilde{\mathcal{X}}_t)} = l_{\tau,i}\mathbb{I}\{i \in \tilde{\mathcal{X}}_t\}$$

□

Lemma 3 (Bounded Second Moment) *Let $\widehat{L}_{t,j}$ be as defined in Algorithm 2. Suppose we obtain $\boldsymbol{p}^t$ through*

$$\boldsymbol{p}^t \in \arg\min_{\boldsymbol{p}\in\mathcal{A}} \eta\left\langle \boldsymbol{p}, \widehat{\boldsymbol{L}}_t\right\rangle + D_F(\boldsymbol{p}, \boldsymbol{p}^{t-1}),$$

under the constraint $\boldsymbol{p}^t(i) \ge \alpha$ for all i. We have

$$\mathbb{E}\left[\sum_{j=1}^{K}\boldsymbol{p}_j^t\widehat{L}_{t,j}^2 \middle| \tilde{\mathcal{X}}_t\right] \le \frac{k}{s} \tag{46}$$

Proof: For $j \notin \tilde{\mathcal{X}}_t$, $\widehat{L}_{t,j} = 0$, For $j \in \tilde{\mathcal{X}}_t$, let

$$X_{r,j} := \mathbb{I}\{A_{t,r=j}\} \frac{\boldsymbol{l}_{\tau,j}}{\boldsymbol{p}^t(j \mid \tilde{\mathcal{X}}_t)}, \quad \text{consequently } \widehat{L}_{t,j} = \frac{1}{s}\sum_{r=1}^{s} X_{r,j}$$

By construction, $X_{r,j}$'s are IID in r, and since $\boldsymbol{l}_{\tau,j} \in [0,1]$,

$$\mathbb{E}\left[X_{r,j}^2 \mid \tilde{\mathcal{X}}_t\right] = \sum_{j \in \tilde{\mathcal{X}}_t} \boldsymbol{p}^t(j \mid \tilde{\mathcal{X}}_t) \frac{\boldsymbol{l}_{\tau,j}^2}{\boldsymbol{p}^t(j \mid \tilde{\mathcal{X}}_t)^2} \leq \frac{1}{\boldsymbol{p}^t(j \mid \tilde{\mathcal{X}}_t)}$$

By Jensen's inequality, we have

$$\mathbb{E}\left[(\widehat{L}_{t,j}^{\tau})^2 \mid \tilde{\mathcal{X}}_t\right] = \mathbb{E}\left[\left(\frac{1}{s}\sum_{r=1}^{s} X_{r,j}\right)^2 \middle| \tilde{\mathcal{X}}_t\right] \leq \frac{1}{s}\mathbb{E}\left[X_{r,j}^2 \middle| \tilde{\mathcal{X}}_t\right] \leq \frac{1}{s\boldsymbol{p}^t(j \mid \tilde{\mathcal{X}}_t)}$$

Summing with weights $\boldsymbol{p}^t$ over $j \in \tilde{\mathcal{X}}_t$ gives

$$\sum_{j \in \tilde{\mathcal{X}}_t} \left(\boldsymbol{p}^t_{|\tilde{\mathcal{X}}_t}\right)_j \mathbb{E}\left[(\widehat{L}_{t,j}^{\tau})^2 \middle| \tilde{\mathcal{X}}_t\right] \leq \frac{1}{s}\sum_{j \in \tilde{\mathcal{X}}_t} \left(\boldsymbol{p}^t_{|\tilde{\mathcal{X}}_t}\right)_j \frac{Z_t}{\left(\boldsymbol{p}^t_{|\tilde{\mathcal{X}}_t}\right)_j} = \frac{k}{s} Z_t \leq \frac{k}{s}$$

□

D.3 One-Step OMD Bound

The purpose of the following lemma is to provide a local inequality that governs how a single mirror descent update behaves under the importance-weighted loss estimator. When summed over time, the KL terms telescope while the second-moment terms accumulate in a controlled manner. This structure allows the proof to separate the effect of the update rule from issues caused by partial observability and non-stationarity, which are handled in later steps.

Lemma 4 *Suppose $\boldsymbol{p}^t$'s are obtained as in Algorithm 2. Then, for any comparator $\boldsymbol{v}_t \in \Delta_{\tilde{\mathcal{X}}_t}$,*

$$\left\langle \boldsymbol{p}^t_{|\tilde{\mathcal{X}}_t} - \boldsymbol{v}_t, \boldsymbol{l}_t \right\rangle \leq \frac{1}{\eta}\left(\mathrm{KL}(\boldsymbol{v}_t \| \boldsymbol{p}^t_{|\tilde{\mathcal{X}}_t}) - \mathrm{KL}(\boldsymbol{v}_t \| \boldsymbol{p}^{t+1}_{|\tilde{\mathcal{X}}_{t+1}})\right) + \frac{\eta s^2}{2}\sum_{i \in C_t} \boldsymbol{p}^t(i \mid \tilde{\mathcal{X}}_t)\widehat{L}_{t,i}^2 \tag{47}$$

Proof: For the simplicity of the proof, fix t. Write $C = \tilde{\mathcal{X}}_t$. Let $p_i = \boldsymbol{p}_i^t$, $Z = Z_t$, $q_i = \boldsymbol{p}^t(i \mid \tilde{\mathcal{X}}_t)$, and $\widehat{L}_i = \widehat{L}_{t,i}$. Write the updated weights as $p_i^+ = p_i \exp(-\eta \widehat{L}_i)$ and the updated active sum as $Z^+ = \sum_{i \in C} p^+$. Let $q_i^+ = p_i^+ / Z^+$. We denote the vector form of the aforementioned quantities in bold. e.g. $\boldsymbol{p}(i) = p_i$.

Note

$$\log \frac{q_i}{q_i^+} = \log \frac{p_i / Z}{p_i \exp(-\eta \widehat{L}_i)/Z^+} = \log \frac{Z^+}{Z} + \eta \widehat{L}_i \tag{48}$$

Now,

$$\begin{aligned} \mathrm{KL}(\boldsymbol{v} \| \boldsymbol{q}^+) - \mathrm{KL}(\boldsymbol{v} \| \boldsymbol{q}) &= \sum_{i \in C} v_i \log \frac{v_i}{q_i^+} - \sum_{i \in C} v_i \log \frac{v_i}{q_i} \\ &= \sum_{i \in C} v_i \log \frac{q_i}{q_i^+} \\ &= \log \frac{Z^+}{Z} + \eta \left\langle \boldsymbol{v}, \widehat{\boldsymbol{L}} \right\rangle \end{aligned} \tag{49}$$

Rearranging we have

$$\eta \left\langle \boldsymbol{v}, \widehat{\boldsymbol{L}} \right\rangle = \mathrm{KL}(\boldsymbol{v} \| \boldsymbol{q}^+) - \mathrm{KL}(\boldsymbol{v} \| \boldsymbol{q}) - \log \frac{Z^+}{Z} \tag{50}$$

Next, apply the fact that $e^{-x} \le 1 - x + x^2$ for $x \ge 0$ and taking logs, we have

$$\begin{aligned}\log \frac{Z^+}{Z} &\le \log \left(\sum_{i \in C} q_i \left(1 - \eta \widehat{L}_i + \frac{\eta^2}{2} \widehat{L}_i^2 \right) \right) \\ &= \log \left(1 - \eta \left\langle \boldsymbol{q}, \widehat{\boldsymbol{L}} \right\rangle + \frac{\eta^2}{2} \sum_{i \in C} q_i \widehat{L}_i^2 \right) \\ &\le -\eta \left\langle \boldsymbol{q}, \widehat{\boldsymbol{L}} \right\rangle + \frac{\eta^2}{2} \sum_{i \in C} q_i \widehat{L}_i^2 \end{aligned} \tag{51}$$

where the last inequality applies $\log(1 + x) \le x$ for all $x > -1$.

Now plug (51) into (50),

$$\eta \left\langle \boldsymbol{v}, \hat{\boldsymbol{L}} \right\rangle \ge \mathrm{KL}(\boldsymbol{v} \| \boldsymbol{q}^+) - \mathrm{KL}(\boldsymbol{v} \| \boldsymbol{q}) + \eta \left\langle \boldsymbol{q}, \hat{\boldsymbol{y}} \right\rangle + \frac{\eta^2}{2} \sum_{i \in C} q_i \widehat{L}_i^2 \tag{52}$$

Rearranging the inequality and dividing by η, we get

$$\langle \boldsymbol{q} - \boldsymbol{v}, \widehat{\mathbf{L}} \rangle \le \frac{1}{\eta} (\mathrm{KL}(\boldsymbol{v} \| \boldsymbol{q}) - \mathrm{KL}(\boldsymbol{v} \| \boldsymbol{q})) + \frac{\eta}{2} \sum_{i \in C} q_i \widehat{L}_i^2 \tag{53}$$

Substituting the original notation back, we recover the claim. □

D.4 Reduction to Blockwise Fixed Arm

The next two lemmas connect the one-step bound to a blockwise analysis under non-stationarity. Lemma 5 controls the discrepancy between the per-round best available arm and a single arm fixed over a block, showing that this gap is governed by the cumulative variation within the block. Lemma 6 then combines this control with the one-step OMD bound to obtain a regret bound against a fixed comparator over the block. Together, these results allow the per-round inequalities to be aggregated while isolating the effect of non-stationarity.

Lemma 5 *Fix a block $\mathcal{B} = \{t_0, t_0 + 1, \ldots, t_0 + L - 1\}$. Dentoe the blockwise best arm as*

$$m_{\mathcal{B}} \in \arg\min_{i \in [K]} \sum_{i \in \mathcal{B}} \widehat{L}_{t,i} \mathbb{I}\{i \in \tilde{\mathcal{X}}_t\} \tag{54}$$

Let Δ_t be as defined in Theorem 5. Then,

$$\sum_{t \in \mathcal{B}} \left(\widehat{L}_{t, m_{\mathcal{B}}} \mathbb{I}\{m_{\mathcal{B}} \in C_t\} - \widehat{L}_{t, m_t} \right) \le L \sum_{t \in B \setminus \{t_0\}} \Delta_t \tag{55}$$

Proof: Let $b_t(C) = \min_{i \in C} \widehat{L}_{i,t}$. By the Lipschitz property of $\min$, it satisfies

$$|b_t(C) - b_{t-1}(C)| \le \max_{i \in C} \left| \widehat{L}_{t,i} - \widehat{L}_{t-1,i} \right| \le \Delta_t \tag{56}$$

By telescoping (56) from t_0 to t,

$$b_{t_0}(C_t) \le b_t(C_t) + \sum_{\tau = t_0 + 1}^{t} \Delta_\tau = m_t + \sum_{\tau = t_0 + 1}^{t} \Delta_\tau \tag{57}$$

By definition of $m_{\mathcal{B}}$ as the best static arm for the block

$$\sum_{t \in \mathcal{B}} \widehat{L}_{t, m_{\mathcal{B}}} \mathbb{I}\{b_{\mathcal{B}} \in \tilde{\mathcal{X}}_t\} \le \sum_{t \in \mathcal{B}} \widehat{L}_{t, m_t^{t_0}} \mathbb{I}\{m_t^{t_0} \in \tilde{\mathcal{X}}_t\} \tag{58}$$

Also note that by definition,

$$\widehat{L}_{t, m_t^{t_0}} \le \widehat{L}_{t_0, m_t^{t_0}} + \sum_{\tau = t_0 + 1}^{t} \Delta_\tau = b_{t_0}(\tilde{\mathcal{X}}_t) + \sum_{\tau = t_0 + 1}^{t} \Delta_\tau \tag{59}$$

Combine (56) and (59),

$$\sum_{t\in\mathcal{B}} \widehat{L}_{t,m_{\mathcal{B}}}\mathbb{I}\{m_{\mathcal{B}} \in \tilde{\mathcal{X}}_t\} \le \sum_{t\in\mathcal{B}} \left(\widehat{L}_{t,m_t} + \sum_{\tau=t_0+1}^{t} \Delta_\tau\right) \tag{60}$$

Rearranging, we get

$$\sum_{t\in\mathcal{B}} \widehat{L}_{t,m_{\mathcal{B}}}\mathbb{I}\{m_{\mathcal{B}} \in \tilde{\mathcal{X}}_t\} - \widehat{L}_{t,m_t} \le L \sum_{t\in\mathcal{B}\setminus\{t_0\}} \Delta_t \tag{61}$$

□

Lemma 6 *Fix a block $\mathcal{B} = \{t_0, t_0+1, \ldots, t_0+L-1\}$. Run Algorithm 2 with $\eta > 0$ starting from time t_0. Let $\boldsymbol{q}_t \in \Delta_{\tilde{\mathcal{X}}_t}$ be the played distribution at each round $t \in \mathcal{B}$ and let $\widehat{\boldsymbol{L}}_t$ be an unbiased estimator of the masked loss vector*

$$\mathbb{E}[\widehat{\boldsymbol{L}}_t \mid \tilde{\mathcal{X}}_t] = \boldsymbol{L}_t^{\tilde{\mathcal{X}}_t}$$

and that the second moment is bounded by

$$\mathbb{E}\left[\sum_{i\in\tilde{\mathcal{X}}_t} \boldsymbol{q}_{t,i}\widehat{L}_{t,i}^2\right] \le \frac{k}{s} \tag{62}$$

for all t. Then,

$$\mathbb{E}\left[\sum_{t\in\mathcal{B}} \left\langle \boldsymbol{q}_t, \boldsymbol{l}_t^{C_t}\right\rangle - \boldsymbol{l}_{t,m_{\mathcal{B}}}\mathbb{I}\{m_{\mathcal{B}} \in \tilde{\mathcal{X}}_t\}\right] \le \frac{\log(1/\alpha)}{\eta} + \frac{sk\eta L}{2} \tag{63}$$

Proof: For each $t \in \mathcal{B}$ and any comparators $\boldsymbol{v}_t \in \Delta_{\tilde{\mathcal{X}}_t}$, apply Lemma 4

$$\sum_{t\in\mathcal{B}} \langle \boldsymbol{q}_t - \boldsymbol{v}_t, \boldsymbol{L}_t\rangle \le \frac{1}{\eta}\sum_{t\in\mathcal{B}} (\mathrm{KL}(\boldsymbol{v}_t\|\boldsymbol{q}_t) - \mathrm{KL}(\boldsymbol{v}_t\|\boldsymbol{q}_{t+1})) + \frac{\eta s^2}{2}\sum_{t\in\mathcal{B}}\sum_{i\in\tilde{\mathcal{X}}_t} \boldsymbol{q}_{t,i}\widehat{L}_{t,i}^2 \tag{64}$$

By the unbiasedness of $\widehat{\mathbf{y}}_t$,

$$\mathbb{E}[\langle \boldsymbol{q}_t - \boldsymbol{v}_t, \boldsymbol{l}_t\rangle] = \mathbb{E}\left[\left\langle \boldsymbol{q}_t - \boldsymbol{v}_t, \boldsymbol{l}_t^{\tilde{\mathcal{X}}_t}\right\rangle\right]$$

For the KL terms, we have

$$\begin{aligned}\sum_{t\in\mathcal{B}} (\mathrm{KL}(\boldsymbol{v}_t\|\boldsymbol{q}_t) - \mathrm{KL}(\boldsymbol{v}_t\|\boldsymbol{q}_{t+1})) &= \mathrm{KL}(\boldsymbol{v}_t\|\boldsymbol{q}_{t_0}) - \mathrm{KL}(\boldsymbol{v}_t\|\boldsymbol{q}_{t_0+L-1}) \\ &\le \log(1/\alpha)\end{aligned} \tag{65}$$

where the first equality is a result of telescoping with respect to t and the second due to the fact that $\mathbf{q}_{t_0}$ is restarted from uniform.

For the second term on the RHS of (64), we invoke Lemma 3. Taking expectation on both sides, we get (63). □

D.5 Proof of Theorem 5

Note that

$$\sum_{t\in\mathcal{B}} \left\langle \boldsymbol{q}_t, \boldsymbol{l}_t^{\tilde{\mathcal{X}}_t}\right\rangle - \boldsymbol{l}_{t,m_t} = \left(\sum_{t\in\mathcal{B}} \left\langle \boldsymbol{q}_t, \boldsymbol{l}_t^{\tilde{\mathcal{X}}_t}\right\rangle - \boldsymbol{l}_{t,m_{\mathcal{B}}}\mathbb{I}\{m_{\mathcal{B}} \in \tilde{\mathcal{X}}_t\}\right) + \left(\sum_{t\in\mathcal{B}} \boldsymbol{l}_{t,m_{\mathcal{B}}}\mathbb{I}\{m_{\mathcal{B}} \in \tilde{\mathcal{X}}_t\} - \boldsymbol{l}_{t,m_t}\right) \tag{66}$$

We bound the first term using Lemma 6 and the second term using 5. Taking the expectation and summing over blocks, we get

$$\mathrm{Reg}_n^{\mathrm{BA}} = \mathbb{E}\left[\sum_{i=1}^{B}\sum_{t\in\mathcal{B}_i} \left\langle \boldsymbol{q}_t, \boldsymbol{l}_t^{\tilde{\mathcal{X}}_t}\right\rangle - \boldsymbol{l}_{t,m_t}\right] \le \frac{T}{L}\frac{\log(1/\alpha)}{\eta} + \frac{\eta sk}{2}T + LV_n \tag{67}$$

Take

$$L^* = \left(\frac{T}{V_n} \sqrt{\frac{sk \log(1/\alpha)}{2}} \right)^{2/3}, \tag{68}$$

$$\eta^* = 2^{2/3} \left(\log(1/\alpha)\right)^{1/3} (sk)^{-2/3} \left(\frac{V_n}{T} \right)^{1/3}. \tag{69}$$

we get

$$\mathrm{Reg}_n^{\mathrm{BA}} \;\leq\; \frac{3}{2^{1/3}} \left(T^2 \, sk \, \log(1/\alpha) \, V_n \right)^{1/3}. \tag{70}$$

E PERFORMANCE IMPROVEMENT CONTRIBUTION OF EACH PROBLEM

We provide further justification for the per-problem policy improvement contribution defined in Eq. (6). Intuitively, this quantity measures the marginal contribution of including a training problem $\boldsymbol{x}$ in an actor update to the overall improvement in policy performance under the evaluation distribution $p_{\mathcal{X}}$.

This interpretation is exact for a broad class of *tabular policies*, where the policy parameters for different problems (or states) are independent. In such settings, updating the policy using trajectories from a problem $\boldsymbol{x}$ affects only the conditional distribution $\pi(\cdot \mid \boldsymbol{x})$ and leaves the policy unchanged on all other problems.

Example: tabular REINFORCE. Consider tabular REINFORCE, where for each problem $\boldsymbol{x}$ the policy $\pi(\cdot \mid \boldsymbol{x})$ is parameterized independently. At iteration t, suppose we update the policy using rollouts collected *only* from a single problem $\boldsymbol{x}$. By construction, this update modifies $\pi(\cdot \mid \boldsymbol{x})$ but does not change $\pi(\cdot \mid \boldsymbol{x}')$ for any $\boldsymbol{x}' \neq \boldsymbol{x}$.

The overall performance objective is

$$J(\pi) = \sum_{\boldsymbol{x}' \in \mathcal{X}} p_{\mathcal{X}}(\boldsymbol{x}')\, \mathbb{E}_{\boldsymbol{y} \sim \pi(\cdot \mid \boldsymbol{x}')} \big[R(\boldsymbol{y} \mid \boldsymbol{x}') \big].$$

Since only the conditional policy at $\boldsymbol{x}$ is changed, the performance difference $J(\pi^{t+1}) - J(\pi^t)$ depends solely on how the expected reward at $\boldsymbol{x}$ changes.

Applying the standard performance difference identity (Kakade & Langford, 2002) to the single-turn setting yields

$$J(\pi^{t+1}) - J(\pi^t) = p_{\mathcal{X}}(\boldsymbol{x})\, \mathbb{E}_{\boldsymbol{y} \sim \pi^t(\cdot \mid \boldsymbol{x})} \left[\frac{\pi^{t+1}(\boldsymbol{y} \mid \boldsymbol{x})}{\pi^t(\boldsymbol{y} \mid \boldsymbol{x})} A_{\pi^t}(\boldsymbol{y} \mid \boldsymbol{x}) \right],$$

which is exactly the per-problem utility $u^t_{\boldsymbol{x}}$ defined in Eq. (6). Thus, in tabular REINFORCE, updating the policy on a single problem $\boldsymbol{x}$ produces an expected performance improvement of $u^t_{\boldsymbol{x}}$.

The same reasoning extends directly to batch updates: if the policy is updated using a subset of problems $\mathcal{X}^t$, the total performance improvement decomposes additively as $\sum_{\boldsymbol{x} \in \mathcal{X}^t} u^t_{\boldsymbol{x}}$, and including a problem $\boldsymbol{x}$ in the update contributes exactly $u^t_{\boldsymbol{x}}$ to the expected performance gain.

Other tabular methods. This exact additive interpretation applies equally to other tabular reinforcement learning methods, including tabular policy gradient methods, tabular Q-learning, and SARSA, where updates based on a problem or state affect only the corresponding local policy or value parameters. In all such cases, the per-problem utility $u^t_{\boldsymbol{x}}$ captures the true marginal contribution of training on $\boldsymbol{x}$.

Discussion: function approximation. In the presence of function approximation, updating the policy using trajectories from one problem generally affects the policy on other problems as well, breaking the exact additivity described above. Analyzing such cross-problem interference requires strong assumptions on the structure of the function class and the optimization dynamics, and is beyond the scope of this work.

Nevertheless, when policy updates are small—as is typical in modern RL post-training algorithms such as PPO-style methods, GRPO, or GSPO—the per-problem utility $u^t_{\boldsymbol{x}}$ remains a first-order approximation to the marginal contribution of problem $\boldsymbol{x}$ to performance improvement. Our empirical results in Sec. 4.1 indicate that this approximation is sufficiently accurate to drive effective curriculum learning at scale.

F LIMITATIONS

While ACTOR-CURATOR is agnostic to the specific actor update rule and can be combined with a wide range of post-training algorithms, its gains ultimately depend on the stability and quality of the underlying actor optimization: unstable updates can still lead to noisy dynamics or training collapse, a common failure mode in reinforcement learning that curriculum learning alone cannot fully address. In addition, ACTOR-CURATOR inherits standard assumptions from RL-based LLM post-training, most notably access to a reliable reward signal, making it best suited to domains with objectively verifiable rewards such as mathematics, logic puzzles, and code. Finally, learning an explicit curator introduces additional computational overhead beyond standard RL post-training; while modest relative to actor training and rollout generation, this cost is non-negligible, and in our experiments ACTOR-CURATOR increases overall wall-clock training time by approximately 9% (see App. H), though this overhead is small compared to the observed gains in training efficiency.

G EXPERIMENTAL SETUP

G.1 HYPER-PARAMETERS

Unless otherwise specified, all experiments use the same hyper-parameter configuration for ACTOR-CURATOR. The complete set of hyper-parameters is summarized in Table 2. We briefly explain key parameters below, with a focus on those specific to the actor–curator framework.

Parameter	**Value**
Model configuration	
Curator model	Qwen3-0.6B
Actor KL loss	Disabled
Max problem length	1024 tokens
Max solution length	4096 tokens
Sampling and batch sizes	
Candidate batch size $\lvert\tilde{\mathcal{X}}^t\rvert$	2048
Training batch size $\lvert\mathcal{X}^t\rvert$	256
Rollouts per problem $\lvert\mathcal{Y}^t_{\boldsymbol{x}}\rvert$	8
Actor optimization	
Actor temperature	1.0
Actor training top-p	1.0
Actor validation top-p	0.7
Actor learning rate	1×10^{-6}
Actor LR warmup ratio	0.05
Actor weight decay	0.1
Actor gradient clipping	1.0
Actor clip range $\rho_{\min}, \rho_{\max}$	$[3 \times 10^{-4},\ 4 \times 10^{-4}]$
Curator optimization	
Curator dormant steps	20
Curator warm-up steps	5
Curator temperature	1.0
Curator top-p	0.9
Curator learning rate	1×10^{-6}
Curator PPO clip range $\rho_{\max} - \rho_{\min}$	0.2

Table 2: Hyper-parameters used for ACTOR-CURATOR across all experiments unless otherwise specified.

Candidate and training batch sizes. At each training iteration, a candidate batch $\tilde{\mathcal{X}}^t$ of size 2048 is first sampled from the proposal distribution q. The curator then reweights this candidate set and samples a smaller training batch $\mathcal{X}^t$ of size 256, which is used for actor rollouts and updates. This two-stage sampling scheme follows Section 3.5 and allows scalable curation over large problem banks.

Rollouts per problem. For each selected problem $x \in \mathcal{X}^t$, we generate $|\mathcal{Y}^t_x| = 8$ on-policy rollouts from the current actor. These rollouts are used both for the actor update and for estimating per-problem policy improvement signals used to train the curator.

Curator dormant and warm-up steps. During the first *curator dormant steps* (20 iterations), problems are sampled uniformly from the candidate batch, and curator outputs are ignored. This stabilizes early actor learning before meaningful policy-improvement estimates can be obtained. During the subsequent *curator warm-up steps* (5 iterations), the curator begins to influence sampling, but its parameters are updated conservatively. After warm-up, the curator is fully active and trained online using bandit feedback.

Sampling prior. When `use sampling prior` is enabled, curator-assigned weights are multiplied by the proposal-induced marginal inclusion probability $q(x)$ before normalization. This

encourages coverage of the full dataset and prevents the curator from collapsing onto a narrow subset of problems early in training, consistent with the two-stage unbiased estimator in Equation (12).

Proximal curator clipping. Curator updates use the PPO-style clipped OSMD objective described in Section 3.6. The clipping range $\rho_{\min}, \rho_{\max}$ constrains the importance ratio between consecutive curator policies, stabilizing learning under function approximation. The additional clip range parameter controls the maximum allowed deviation between these bounds.

G.2 HARDWARE

All experiments were conducted on NVIDIA A100 and H200 GPUs. Actor rollout generation and optimization dominate overall runtime; curator training introduces approximately 14% additional wall-clock cost relative to uniform sampling, as discussed in Appendix E.

H ADDITIONAL ANALYSIS

H.1 OVERHEAD

We found that ACTOR-CURATOR adds about 9% training wall time overhead. We present the overhead by dataset and model in Tab. 3. However, as shown in Fig. 11, this is relatively the compared to the efficiency gains.

Table 3: **Average wall-time overhead percentage** by model and datasets, averaged over training steps 500 steps.

	COUNTDOWN	**ZEBRA**	**ARC-1D**	**MATH**
QWEN2.5-3B-BASE	11.86	16.71	17.12	9.56
LLAMA3.2-3B-IT	9.34	13.63	19.40	9.82

I ADDITIONAL ABLATION

Candidate batch size $|\tilde{\mathcal{X}}|$. As shown in Fig. 8, candidate sizes of 512 and 2048 yield similar final performance. Larger batches (e.g., 8192) lead to unstable training, likely due to reduced exploration and overfitting.

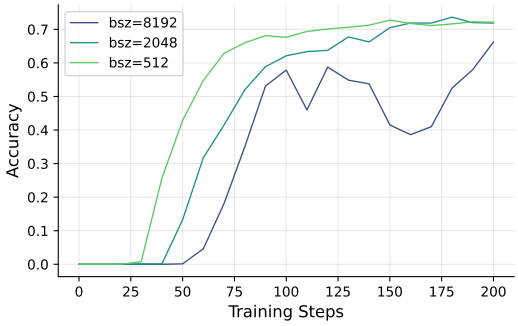
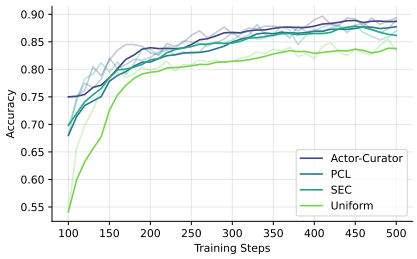
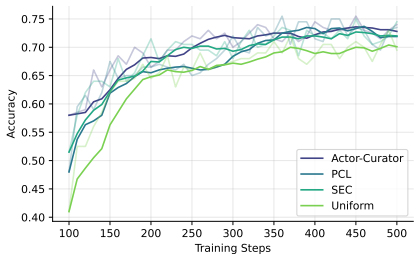


Figure 8: **Effect of candidate batch size on test performance**. While `bsz=2048` and `bsz=512` converge to similar performance at step 200, `bsz=8192` suffers from instability. Selected batch size is held constant at 256.

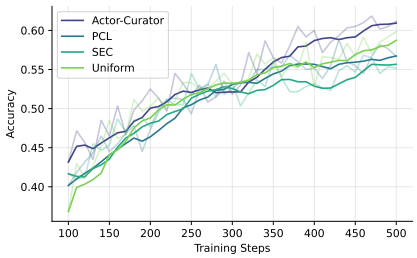
Figure 9: Test set performance across training steps (within 100 training steps) for **Qwen2.5-3B-Base** across benchmarks and methods.



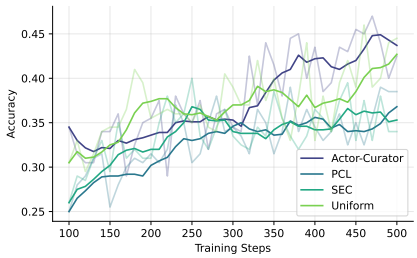
(a) **Countdown**



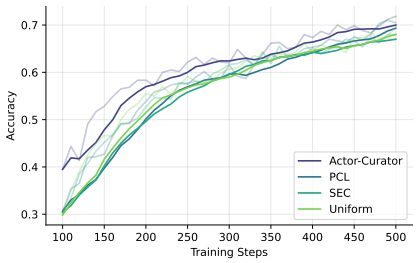
(b) **Countdown-hard**



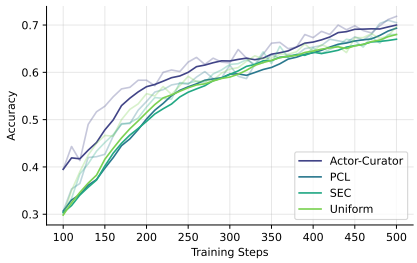
(c) **Zebra**



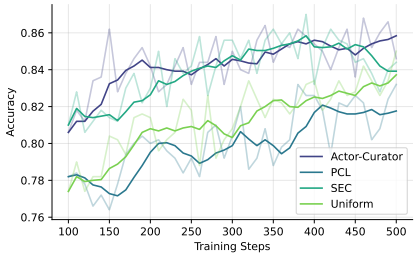
(d) **Zebra-hard**



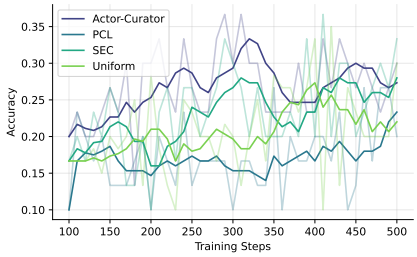
(e) **ARC-1D**



(f) **ARC-hard**



(g) **MATH500**



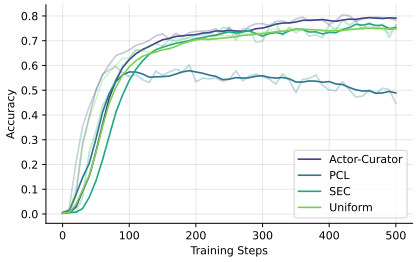
(h) **AIME24**

J MAIN RESULTS (EXTENDED)

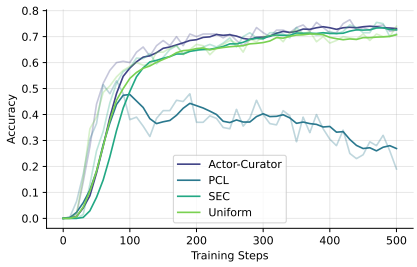
This appendix provides extended empirical results complementing the main paper. We report additional quantitative comparisons across benchmarks, models, and curriculum learning methods, as well as detailed training dynamics over time. These results further substantiate the robustness, efficiency, and stability of AC across diverse problem domains and model backbones.

Extended performance comparison. Tab. 4 reports peak validation performance within the first 100 training steps across all benchmarks and models. Compared to uniform sampling, heuristic curricula (SEC), and learning-based baselines (PCL), AC consistently achieves higher peak performance

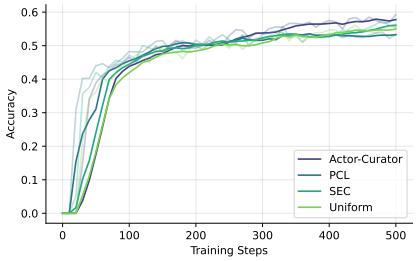
Figure 10: Test set performance across training steps (within 100 training steps) for **Llama3.2-3B-it** across benchmarks and methods.



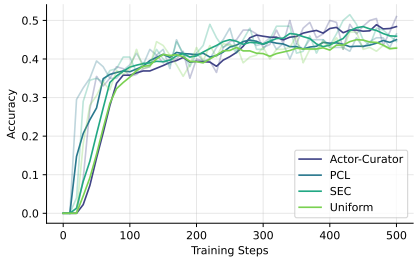
(a) **Countdown**



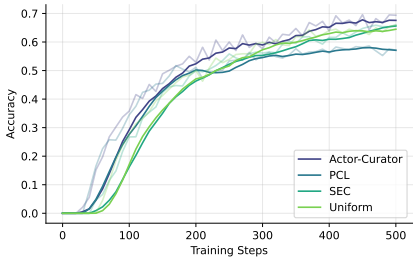
(b) **Countdown-hard**



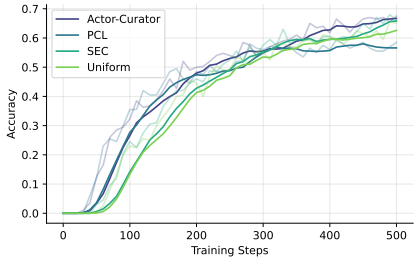
(c) **Zebra**



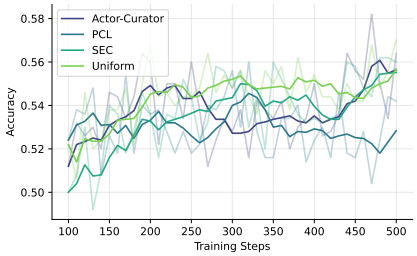
(d) **Zebra-hard**



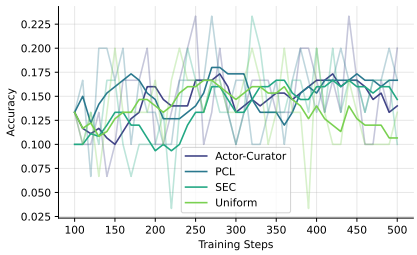
(e) **ARC-1D**



(f) **ARC-hard**



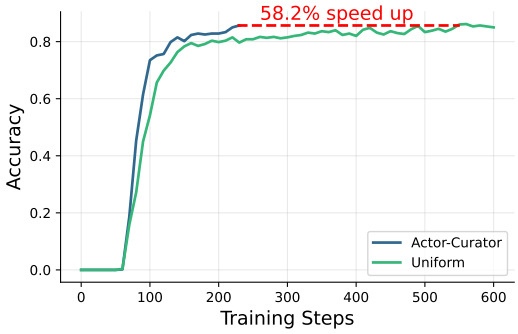
(g) **MATH500**



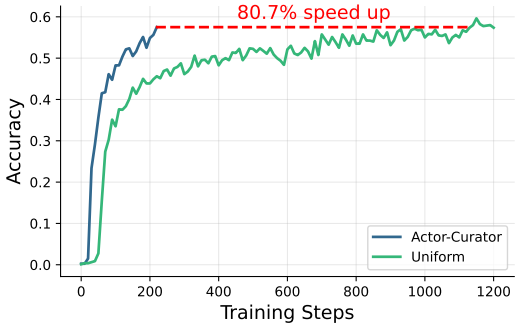
(h) **AIME24**

on most benchmarks. The gains are particularly pronounced on harder subsets (e.g., Countdown-hard, Zebra-hard, ARC-hard, and AIME24), highlighting the effectiveness of directly optimizing for expected policy improvement when problem difficulty and utility are highly non-uniform. While performance on MATH500 is largely saturated for some model configurations, AC remains competitive and avoids degradation relative to strong baselines.

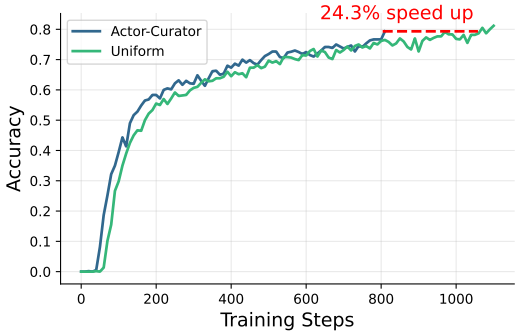
Figure 11: **Training efficiency:** Actor-Curator attains significant efficiency increase with relatively low overhead on Countdown, Zebra, and ARC.



(a) **Countdown**



(b) **Zebra**



(c) **ARC-1D**

Training dynamics and stability. Figs. 9 and 10 visualize test performance as a function of training steps for Qwen2.5-3B-Base and Llama3.2-3B-it, respectively. Across benchmarks, AC not only reaches higher peak performance but also exhibits faster convergence and more stable learning dynamics. In many cases, competing methods plateau early or exhibit higher variance, whereas AC continues to make steady progress, effectively raising the performance ceiling.

Model-agnostic behavior. The trends observed in Figs. 9 and 10 are consistent across both base and instruction-tuned models, indicating that the benefits of AC are not tied to a specific initialization or training regime. This supports the claim that learning curricula via policy-improvement-driven signals provides a generally applicable mechanism for improving RL post-training efficiency and robustness.

Table 4: Peak validation performance on problems within 100 training steps across models and methods. ACTOR-CURATOR outperforms both other learning based methods (PCL) and methods that rely on human heuristics (SEC). **Models:** Qwen2.5 refers to Qwen2.5-3B-Base; Llama3.2 refers to Llama3.2-3B-it.

BENCHMARK	MODEL	METHOD					IMPROVEMENT	
		$\pi_{\mathbf{ref}}$	UNIFORM	SEC	PCL	AC (OURS)	$+\Delta$	$+\Delta\%$
COUNTDOWN	QWEN2.5	0.00	44.74	58.87	57.24	62.12	+3.25	+5.52
	LLAMA3.2	0.00	63.12	62.78	59.62	66.25	+3.13	+4.96
CD-HARD	QWEN2.5	0.00	41.00	51.50	48.00	58.00	+6.50	+12.62
	LLAMA3.2	0.00	58.50	58.00	53.00	60.50	+2.00	+3.42
ZEBRA	QWEN2.5	0.00	35.12	36.00	34.12	37.62	+1.62	+4.50
	LLAMA3.2	0.00	44.50	46.50	48.25	47.12	-1.13	-2.34
ZEBRA-HARD	QWEN2.5	0.00	30.50	27.50	26.00	34.50	+4.00	+13.11
	LLAMA3.2	0.00	37.50	38.00	39.50	40.50	+1.00	+2.53
ARC-1D	QWEN2.5	0.00	26.74	27.87	26.37	36.37	+8.50	+30.51
	LLAMA3.2	0.00	27.62	26.75	34.50	35.25	+0.75	+2.17
ARC-HARD	QWEN2.5	0.00	19.50	18.50	18.50	31.00	+11.50	+58.97
	LLAMA3.2	0.00	23.00	24.50	24.00	31.50	+7.00	+28.57
MATH500	QWEN2.5	61.80	83.00	81.00	79.79	81.00	-2.00	-2.41
	LLAMA3.2	41.00	52.20	52.00	52.40	53.60	+1.20	+2.29
AIME24	QWEN2.5	3.33	23.33	20.00	23.33	30.00	+6.67	+28.57
	LLAMA3.2	0.00	13.33	13.33	13.33	16.67	+3.34	+25.06

Training efficiency. Fig. 11 compares learning curves of Actor-Curator against uniform sampling on Countdown, Zebra, and ARC. Across all three benchmarks, Actor-Curator reaches the same target accuracy substantially earlier, yielding step-level speedups of 58.2% on Countdown, 80.7% on Zebra, and 24.3% on ARC. This indicates that policy-improvement-driven data selection primarily accelerates optimization by prioritizing high-impact problems.