BANDITSCHED: ADAPTIVE DATA SCHEDULING WITH CONTEXTUAL BANDITS FOR EFFICIENT REASONING REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) has emerged as a powerful paradigm for eliciting reasoning capabilities in large language models. As the model's competence evolves during training, so does the value of each training problem, and data selection must adapt to this non-stationary process. Existing methods rely on offline profiling, coarse-grained curricula, or filtering after rollouts have already been generated. We observe that the reward-based learning signal of a problem is governed by its current pass rate: groups with intermediate success rates yield nonzero mean absolute advantages, while groups with uniformly correct or incorrect responses yield zero advantages. Motivated by this observation, we formulate RLVR data selection as a *non-stationary contextual bandit* problem, where each training problem is an arm and its context encodes competence statistics, difficulty, and reasoning-pattern information. We propose BANDITSCHED, a Thompson Sampling-based framework that (1) predicts prompt utility before allocating rollouts, (2) periodically probes previously attempted low-success problems to refresh estimates near the model's competence frontier without policy updates, and (3) combines advantage magnitude, competence-edge membership, and question-space diversity into a composite reward. A sliding-window reward model tracks the changing utilities, with its window length guided by an estimation–drift trade-off. Experiments on mathematical reasoning, code generation, and logical puzzles with Qwen2.5 and Qwen3 models show that BANDITSCHED improves rollout efficiency while matching or exceeding the final accuracy of strong baselines, with the largest gains on hard and out-of-distribution tasks. Specifically, BANDITSCHED reaches 95% of Full-GRPO's final accuracy using 30–35% of its online rollouts and completes full training runs with an approximately $1.8\times$ wall-clock speedup. Controlled comparisons further show that contextual scheduling improves efficiency beyond pass-rate band filtering alone.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) has recently become the standard recipe for post-training large language models (LLMs) to reason (Guo et al., 2025; Jaech et al., 2024). By optimizing a policy against binary outcome rewards, i.e., correct or incorrect answers verified by rule-based checkers, RLVR induces chain-of-thought reasoning behaviors that improve mathematical, coding, and logical problem-solving capabilities (Yu et al., 2025; Zeng et al., 2025; Shao et al., 2024). Despite impressive progress, a critical bottleneck remains under-explored: *which training problems should the model practice on, and when?* The answer is dynamic. As illustrated in Figure 1, problems near the model's *edge of competence* (Huang et al., 2026) tend to produce larger mean absolute group-relative advantages, which measure the reward-based learning signal they provide, and this set of problems shifts as training progresses.

The current landscape of RLVR data selection can be organized along two axes: *when* selection happens and *what* criteria guide it. Methods based on difficulty (Tang et al., 2025) or influence scores (Zhu et al., 2025) combine profiling with online filtering, replay, or staged selection, but the profiling step incurs additional computation. DAPO's dynamic sampling (Yu et al., 2025) filters prompts whose rollout groups have pass rates of exactly 0 or 1; these decisions are adaptive, but

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

the discarded rollouts have already been generated. Curriculum-based approaches adapt training through easy-to-hard schedules or bandit decisions over problem categories (Parashar et al., 2025; Chen et al., 2025b). Because they operate on coarse categories, they cannot distinguish problems within the same category whose learning value differs. These limitations motivate learning a predictive utility model for individual problems.

We develop a framework that treats each training problem as a distinct option whose learning value is uncertain and evolving. This motivates a non-stationary contextual bandit formulation (Lattimore & Szepesvári, 2020). The context encodes problem features and cached competence statistics; the reward is an observed proxy for learning value; and non-stationarity arises as the policy and its rollout outcomes change. Estimating utility before generating responses allows the scheduler to allocate rollout computation using information shared across problems.

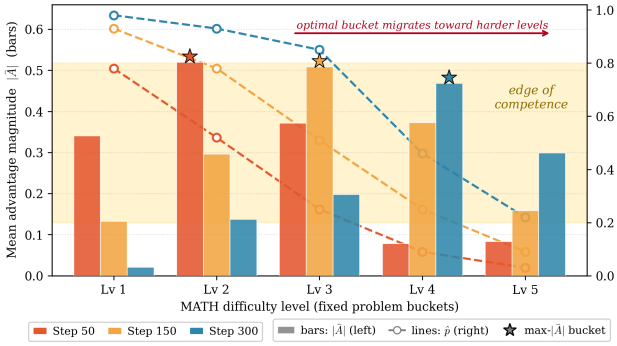

Figure 1: Per-problem mean absolute advantage $|\bar{A}|$ during GRPO training of Qwen2.5-3B on MATH. Problems are binned by the given difficulty levels for simplicity. The edge-of-competence band (pass rate in $[0.2, 0.8]$) consistently exhibits high $|\bar{A}|$, i.e., strong reward-based learning signals.

In this paper, we propose BANDITSCHED (**Bandit**-guided **Sched**uling for Reasoning RL), a framework that casts RLVR data selection as a non-stationary contextual bandit and uses Thompson Sampling for prompt selection. Our approach makes three contributions: **(1) Contextual bandit formulation (§3.2).** We combine pass-rate dynamics, difficulty metadata, and reasoning-pattern embeddings to predict prompt utility before rollout. A composite reward uses advantage magnitude, edge-of-competence membership, and diversity in question space. **(2) Dual-layer exploration with frontier probing (§3.4).** In addition to posterior sampling, periodic probe rollouts revisit previously attempted low-success problems that lie near higher-success ones in embedding space, refreshing their competence estimates without policy-gradient updates. **(3) Design analysis and empirical evaluation (§3.5, §4).** We analyze the estimation–drift trade-off that guides the window length, and evaluate the contributions of context, probing, reward modeling, and initialization against strong filtering baselines. Experiments with Qwen2.5 (3B, 7B) and Qwen3 (4B, 8B) across math, code, and logic show that BANDITSCHED substantially lowers rollout cost relative to Full-GRPO with DAPO-style dynamic sampling while achieving competitive final accuracy. On math, BANDITSCHED reaches the common accuracy target with 30–35% of Full-GRPO's online rollouts and achieves an approximately $1.8\times$ full-run wall-clock speedup.

2 RELATED WORK

Reinforcement learning for LLM reasoning. The paradigm of post-training LLMs with RL to elicit reasoning traces was popularized by OpenAI o1 (Jaech et al., 2024) and DeepSeek-R1 (Guo et al., 2025). The standard algorithm is Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which eliminates the value network by computing advantages relative to a group of sampled responses. Subsequent works have improved GRPO's stability through decoupled clipping and dynamic sampling (DAPO; Yu et al. 2025), difficulty-aware advantage shaping (GRPO-LEAD; Zhang & Zuo 2025), and token-level reward shaping (Yue et al., 2025b). SimpleRL-Zoo (Zeng et al., 2025) systematically investigates zero RL training across diverse base models and highlights the importance of query difficulty control.

Data scheduling and selection for RLVR. Prior work on RLVR training data follows two main lines. Curriculum methods adapt training difficulty over time. E2H-Reasoner (Parashar et al., 2025) uses a probabilistic easy-to-hard scheduler with finite-sample guarantees, and SEC (Chen et al., 2025b) runs a non-stationary bandit over problem categories. SATURN (Liu et al., 2025a) controls difficulty with SAT problems, and ProRL (Liu et al., 2025b) trains on progressively harder data. Data-efficiency methods reduce training cost. DEPO (Tang et al., 2025) combines offline pass-rate filtering with online explorability and replay, reaching a $1.85\times$ speedup with 20% of the data.

CROPI (Zhu et al., 2025) selects data with off-policy influence estimates for a 2.66× acceleration. LIMO (Ye et al., 2025) and 1-shot RLVR (Wang et al., 2025) further show that very small subsets can elicit strong reasoning. Other studies analyze the underlying training dynamics. Huang et al. (2026) find difficulty-driven phase transitions, and Chen et al. (2025a) study clipping and entropy in exploration. UCAS (Xie et al., 2025) and ReLaX (Zhang et al., 2025) encourage exploration through uncertainty-aware advantage shaping and latent dynamics, respectively, and Tang et al. (2026) show that positive and negative samples play asymmetric roles. Whether RLVR expands the base model's reasoning boundary also remains debated (Yue et al., 2025a; Chen et al., 2025c). Our work casts data selection as a *contextual* bandit over individual problems, combining curriculum adaptation, efficiency-oriented selection, and exploration in one predictive scheduler.

Contextual bandits. Thompson Sampling (TS) (Thompson, 1933; Agrawal & Goyal, 2013) and Upper Confidence Bound (UCB) (Auer et al., 2002; Li et al., 2010) are classical approaches to the exploration–exploitation trade-off in bandits. For non-stationary environments, sliding-window and discounted variants (Garivier & Moulines, 2011) have been developed. We apply sliding-window contextual Thompson Sampling to prompt-level RLVR data selection with reasoning-pattern features and frontier probing.

3 METHODOLOGY

We present BANDITSCHED, which formulates RLVR data selection as a non-stationary contextual bandit and solves it using Thompson Sampling with a dual-layer exploration mechanism.

3.1 PRELIMINARIES: GRPO FOR REASONING

Given a training prompt set $\mathcal{D} = \{x_1, \ldots, x_N\}$ with verifiable answers, GRPO (Shao et al., 2024) optimizes the policy π_θ by sampling a group of G responses $\{y_i^{(g)}\}_{g=1}^{G}$ for each prompt x_i, computing binary rewards $r_i^{(g)} \in \{0, 1\}$ via a verifier, and estimating group-relative advantages:

$$\hat{A}_i^{(g)} = \frac{r_i^{(g)} - \bar{r}_i}{\max(\sigma(r_i), \epsilon)}, \quad \bar{r}_i = \frac{1}{G}\sum_{g=1}^{G} r_i^{(g)}, \quad \sigma(r_i) = \sqrt{\frac{1}{G}\sum_{g=1}^{G}(r_i^{(g)} - \bar{r}_i)^2}. \tag{1}$$

The clipped surrogate objective is then:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\mathbb{E}_{x_i, y_i^{(g)}}\left[\min\left(\rho_i^{(g)}\hat{A}_i^{(g)},\ \text{clip}(\rho_i^{(g)}, 1-\epsilon_l, 1+\epsilon_h)\hat{A}_i^{(g)}\right)\right], \tag{2}$$

where $\rho_i^{(g)} = \pi_\theta(y_i^{(g)}|x_i)/\pi_{\theta_{\text{old}}}(y_i^{(g)}|x_i)$ is the importance sampling ratio. We summarize the reward-based learning signal of prompt x_i at step t by

$$\bar{A}_{i,t} := \frac{1}{G}\sum_{g=1}^{G}|\hat{A}_{i,t}^{(g)}|, \qquad q_{i,t} := \frac{1}{G}\sum_{g=1}^{G} r_{i,t}^{(g)}, \tag{3}$$

where $q_{i,t}$ is the success rate of the current rollout group. Because the advantages are centered within each group, their signed mean is always zero; the mean absolute value instead measures how much the rewards in the group differ. With binary rewards and an inactive numerical stabilizer, $\bar{A}_{i,t} = 2\sqrt{q_{i,t}(1-q_{i,t})}$ (Appendix A.1). This signal vanishes when all G responses are correct or all are incorrect, and it peaks at $q_{i,t} = 0.5$. In other words, a prompt yields a nonzero policy gradient only if the model sometimes solves it and sometimes fails, which explains why the learning signal concentrates at the edge of competence (Figure 1). We therefore use $\bar{A}_{i,t}$, which can be read directly from verifier outcomes, as the core utility signal for data selection.

3.2 RLVR DATA SELECTION AS A NON-STATIONARY CONTEXTUAL BANDIT

Problem setup. At each training step $t = 1, \ldots, T$, the scheduler selects a batch $\mathcal{B}_t \subset \mathcal{D}$ of B prompts, the policy generates G responses for each of them, and GRPO updates the policy on the resulting groups. We treat each prompt as an arm. Selecting a prompt pulls its arm, and its rollout outcomes yield a composite reward $r_t(x_i)$ that serves as a proxy for its learning value. Rewards are

observed only for the selected prompts, and they are non-stationary because the policy π_{θ_t} changes after every update. The number of arms is also large relative to the selection budget: uniform selection visits each prompt only BT/N times on average, just a few in practice (§4.1), so estimating each prompt's utility independently is unreliable. We therefore describe each prompt by a context vector and learn a single reward model shared across prompts. Observations on one prompt then inform predictions for related ones, including problems within the same difficulty bucket (Appendix B).

Context features. For prompt x_i at step t, we construct

$$\mathbf{c}_{i,t} = \left[\underbrace{\hat{p}_{i,t}, \Delta\hat{p}_{i,t}, 4\hat{\sigma}^2_{i,t}}_{\text{competence features}}, \underbrace{\widetilde{d}_i, \widetilde{\mathbf{e}}_i^\top}_{\text{problem features}}, 1 \right]^\top. \tag{4}$$

The competence features describe the model's current standing on the prompt. The pass-rate exponential moving average (EMA) $\hat{p}_{i,t}$ locates the prompt relative to the edge of competence, and its change $\Delta\hat{p}_{i,t} = \hat{p}_{i,t} - \hat{p}_{i,t-\tau}$ over the last τ steps indicates whether the model is making progress on it. The variance of the pass-rate estimates, $\hat{\sigma}^2_{i,t}$, reflects the uncertainty of the current estimate; the factor of 4 maps it to $[0, 1]$. Among the static problem features, difficulty metadata $\widetilde{d}_i$ is scaled to $[0, 1]$ when available. The reasoning-pattern embedding is computed from the question alone: we mean-pool the base model's final-layer hidden states over the problem statement, project them to a d_e-dimensional vector $\mathbf{e}_i$, and normalize it as $\widetilde{\mathbf{e}}_i = \mathbf{e}_i / \|\mathbf{e}_i\|_2$. The encoder and projection are frozen before RLVR training. Problems with similar reasoning patterns often become learnable together, so the embedding lets the reward model generalize from frequently observed prompts to rarely observed ones. Including the intercept, the context has dimension $d = d_e + 5$ (Appendix D.1). Competence statistics are cached, so constructing contexts requires no additional rollouts.

Composite reward. For each selected prompt, we use

$$r_t(x_i) = \alpha \frac{\bar{A}_{i,t}}{\max_{x_j \in \mathcal{B}_t} \bar{A}_{j,t}} + \beta \mathbb{I}[\hat{p}_{i,t} \in (p_l, p_h)] + \gamma \, \mathrm{Div}(x_i, t). \tag{5}$$

where $0 < p_l < p_h < 1$ delimit the competence band and $\alpha, \beta, \gamma \geq 0$ weight the three terms, which capture complementary aspects of learning value. The first term measures the learning signal realized by the current rollout group. It is normalized by the batch maximum so that its scale is comparable across steps, and it is set to zero when all batch advantages vanish. The second term rewards prompts whose smoothed pass-rate estimate lies inside the competence band. A single group of G responses gives a noisy success rate, so this EMA-based indicator adds a more stable signal of competence-edge membership. The third term, $\mathrm{Div}(x_i, t)$, is the cosine distance between the question embedding of x_i and the centroid of question embeddings selected in the current epoch, which favors questions from under-covered regions.

Reward model. To predict rewards before any rollout, we model the expected reward as a time-varying linear function of the context,

$$\mathbb{E}[r_t(x_i) \mid \mathbf{c}_{i,t}] \approx \mathbf{c}_{i,t}^\top \boldsymbol{\theta}_t^*, \tag{6}$$

where $\boldsymbol{\theta}_t^*$ captures the current relationship between contexts and utility. The linear form yields a closed-form posterior and makes scoring all N prompts at every step inexpensive. Because $\boldsymbol{\theta}_t^*$ drifts as the policy improves, we fit it only on observations from a sliding window of recent steps (§3.3); §3.5 discusses the choice of the window length.

3.3 Thompson Sampling for Prompt Selection

We maintain a Gaussian posterior over the reward parameter using sliding-window Bayesian linear regression. Let $s_t = \max(1, t - W)$. Observations from steps s_t through $t - 1$ define

$$\begin{aligned} \boldsymbol{\theta}_t \mid \mathcal{H}_{t-1} &\sim \mathcal{N}(\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t), \\ \hat{\boldsymbol{\Sigma}}_t &= \left(\lambda \mathbf{I} + \sum_{s=s_t}^{t-1} \sum_{x_i \in \mathcal{B}_s} \mathbf{c}_{i,s} \mathbf{c}_{i,s}^\top \right)^{-1}, \end{aligned} \tag{7}$$

$$\hat{\boldsymbol{\mu}}_t = \hat{\boldsymbol{\Sigma}}_t \sum_{s=s_t}^{t-1} \sum_{x_i \in \mathcal{B}_s} r_s(x_i)\mathbf{c}_{i,s}, \tag{8}$$

where $\lambda > 0$ is the regularization parameter and $\mathcal{H}_{t-1}$ denotes the observed history. Each observation pairs the pre-rollout context $\mathbf{c}_{i,s}$ with the realized reward $r_s(x_i)$.

Batch selection. At step t, we draw $\tilde{\boldsymbol{\theta}}_t \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t)$, compute $\tilde{r}_t(x_i) = \mathbf{c}_{i,t}^\top \tilde{\boldsymbol{\theta}}_t$ for every $x_i \in \mathcal{D}$, and select the top-B prompts. The sampled parameter drives exploration. Where recent observations cover the context space well, the posterior is concentrated and selection follows the estimated utility. Elsewhere, the sampled scores vary across steps, so prompts of uncertain value are occasionally selected and their utility can be learned. Ranking all prompts under a single sample yields a full batch from one posterior draw.

Statistic updates and initialization. After the rollouts, the pass-rate EMA of each observed prompt is updated with its group success rate,

$$\hat{p}_{i,t+1} = (1 - \eta_p)\, \hat{p}_{i,t} + \eta_p\, q_{i,t}, \tag{9}$$

while unobserved prompts keep their previous statistics. Training starts without a dataset-wide rollout sweep: all EMAs start from the same neutral value, and the prior is $\mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I})$ (Appendix D.2). At the first step, the competence features are identical across prompts, so the ranking is determined by a random linear function of the problem features; later observations differentiate prompts through their competence features. Scoring costs $O(Nd)$ operations per step, and the posterior involves only a $d \times d$ matrix (Appendix D.4).

3.4 Frontier Probing: Exploring the Reasoning Boundary

A prompt's statistics are refreshed only when it is selected. A problem that the model fails early in training receives a low pass-rate EMA and a low predicted utility, so it is rarely selected again, and its estimate can remain stale after related skills have improved. Posterior sampling explores uncertainty in the reward parameters rather than in the cached statistics of individual prompts. Frontier probing therefore adds a second layer of exploration that periodically re-evaluates such problems.

Frontier identification. Every K steps, let $n_{j,t}$ count the rollout groups previously observed for question x_j. We identify

$$\mathcal{F}_t = \Big\{ x_j \in \mathcal{D} :\ n_{j,t} \geq 1, \quad \hat{p}_{j,t} < p_l, \max_{x_i : \hat{p}_{i,t} \geq p_l} \cos(\mathbf{e}_j, \mathbf{e}_i) > \tau_{\text{sim}} \Big\}. \tag{10}$$

The first two conditions select attempted problems with low estimated success. The third requires each of them to be close, in reasoning-pattern space, to a problem whose estimated pass rate has reached p_l. Probing thus targets failed problems whose related skills the model has begun to acquire.

Probe rollouts. We sample $\mathcal{F}_t^{\text{probe}} \subset \mathcal{F}_t$ with $B_{\text{probe}} \ll B$ questions and generate G responses for each of them without a policy-gradient update. The probe outcomes update the pass-rate statistics through Eq. (9), and hence the contexts at subsequent steps. When a probe reveals that a problem has become partially solvable, its EMA and predicted utility rise, so Thompson Sampling can select it for training. Every problem in $\mathcal{D}$ remains a candidate for the main batch.

Cost analysis. Probing adds $B_{\text{probe}}G$ responses every K steps, a fraction $B_{\text{probe}}/(KB)$ of the nominal training responses. Since $B_{\text{probe}} \ll B$ and probes skip the policy update, this overhead is small. Appendix E.2 reports the measured probe time alongside generation and total time.

Algorithm 1 summarizes training-batch selection, GRPO updates, periodic probing, and posterior/statistic updates. Probe outcomes update only the cached statistics, while the posterior uses training observations alone.

3.5 Design Rationale for the Sliding Window

The window length W balances two sources of error in the reward model. A longer window provides more observations for estimating $\boldsymbol{\theta}_t^*$, but it also retains stale observations collected under older policies. Building on analyses of linear Thompson Sampling and sliding-window bandits (Agrawal & Goyal, 2013; Garivier & Moulines, 2011), we use a simple proxy to make this trade-off explicit.

Algorithm 1 BANDITSCHED: Contextual Bandit-Guided Data Selection for RLVR

Require: Training set $\mathcal{D}$, batch size B, group size G, window W, probe interval K, probe size B_{probe}, thresholds $p_l, p_h, \tau_{\text{sim}}$, reward weights α, β, γ

Initialize the prior and cached statistics as in Appendix D.2; set all observation counts to zero
for $t = 1, 2, \ldots, T$ **do**
  Construct and cache $\mathbf{c}_{i,t}$ for all $x_i \in \mathcal{D}$ using Eq. (4)
  Sample $\tilde{\boldsymbol{\theta}}_t \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t)$; score $\tilde{r}_t(x_i) = \mathbf{c}_{i,t}^\top \tilde{\boldsymbol{\theta}}_t$
  Select $\mathcal{B}_t$ as the top-B questions from $\mathcal{D}$
  Generate G responses per $x_i \in \mathcal{B}_t$; compute $q_{i,t}$ and $\hat{A}_{i,t}^{(g)}$
  Update the policy using $\mathcal{L}_{\text{GRPO}}$ on $\mathcal{B}_t$
  Compute $r_t(x_i)$ using Eq. (5) and retain the cached contexts for regression
  if $t \bmod K = 0$ **then**
    Identify $\mathcal{F}_t$ using Eq. (10) and sample $\mathcal{F}_t^{\text{probe}}$
    Execute probe rollouts without a policy-gradient update
  end if
  Update $(\hat{\boldsymbol{\mu}}_{t+1}, \hat{\boldsymbol{\Sigma}}_{t+1})$ from training observations using Eqs. (7)–(8)
  Update cached pass-rate statistics (Eq. (9)) and observation counts from training and probe groups in observation order
end for

An estimation–drift proxy. Suppose the reward parameter drifts by at most δ per step, $\|\boldsymbol{\theta}^*_{t+1} - \boldsymbol{\theta}^*_t\|_2 \le \delta$, and contexts have norm at most L. Over W steps, drift then changes a linear prediction by at most $L\delta W$. Combining this staleness cost with a standard estimation term gives

$$\mathcal{J}(W) = adT\sqrt{\frac{\log(NT)}{W}} + bL\delta WT, \qquad 1 \le W \le T, \tag{11}$$

where $a, b > 0$ are problem-dependent constants. The first term decreases as the window provides more observations, while the second grows with staleness. For $\delta > 0$, the unconstrained minimizer is

$$W_0 = \left(\frac{ad\sqrt{\log(NT)}}{2bL\delta}\right)^{2/3}, \tag{12}$$

which we clip to $[1, T]$ and round in practice. For an interior optimum, the proxy scales as

$$\mathcal{J}(W_0) = O\Big(d^{2/3}(L\delta)^{1/3}T[\log(NT)]^{1/3}\Big) \tag{13}$$

(Appendix A.2). The key implication is $W_0 \propto \delta^{-2/3}$: the faster the utility relationship drifts, the shorter the window should be. Because the constants and the drift rate are unknown in practice, we treat W as a hyperparameter and examine this trade-off empirically in §4.4.

4 EXPERIMENTS

4.1 EXPERIMENTS SETUP

Datasets and Benchmarks. For *training data*, we adopt the DAPO-Math-17K dataset (Yu et al., 2025) as our primary training set, which contains 17K curated mathematical problems with difficulty annotations. For code reasoning, we use the training split of TACO (Li et al., 2023) (5K problems). For logical puzzles, we generate Countdown and Zebra problems using the Open-Thought framework (Guha et al., 2026) at three difficulty levels (10K each). For *evaluation*, we use the following benchmarks: (1) **MATH-500** (Hendrycks et al., 2021): 500 competition-level math problems, stratified by difficulty levels 1–5. (2) **AIME 2024** (Mathematical Association of America, 2024): 30 problems from the American Invitational Mathematics Examination (reported as Avg@32). (3) **Minerva Math** (Lewkowycz et al., 2022): 272 undergraduate-level STEM problems (in Appendix E.1). (4) **LiveCodeBench** (Jain et al., 2025): recent coding problems with automated test-case verification (Pass@1, problems after October 2024). (5) **Countdown** and **Zebra Puzzles** (Gandhi et al., 2025): procedurally generated logic tasks with OOD difficulty evaluation.

Table 1: **Mathematical reasoning results.** Pass@1 on MATH-500 and Avg@32 on AIME 2024. **Eff.** is the online rollout count to 95% of Full-GRPO's final MATH-500 accuracy, normalized by Full-GRPO's count to the same target; ↓ denotes lower is better. Best in **bold**, second underlined.

Method	Qwen2.5-3B			Qwen2.5-7B			Qwen3-4B			Qwen3-8B		
	MATH	AIME	Eff.↓	MATH	AIME	Eff.↓	MATH	AIME	Eff.↓	MATH	AIME	Eff.↓
Base Model	48.4	12.3	–	58.2	18.7	–	59.6	20.1	–	66.8	26.4	–
Full-GRPO	63.8±0.6	26.8±1.4	1.00	74.6±0.5	36.2±1.3	1.00	75.2±0.5	37.4±1.2	1.00	81.4±0.4	44.6±1.1	1.00
Random-20%	59.2±1.0	22.1±1.8	0.82	70.3±0.9	31.4±1.6	0.80	71.0±0.8	32.8±1.5	0.81	77.2±0.7	39.5±1.4	0.79
DEPO	63.1±0.7	26.2±1.5	0.48	73.9±0.6	35.6±1.4	0.45	74.5±0.5	36.8±1.3	0.46	80.8±0.4	43.9±1.2	0.44
SEC	63.5±0.6	27.1±1.3	0.55	73.1±0.7	35.8±1.2	0.52	74.8±0.5	37.0±1.2	0.53	81.0±0.4	44.2±1.1	0.50
E2H	62.9±0.8	25.8±1.6	0.58	73.4±0.6	35.2±1.5	0.55	74.1±0.6	36.4±1.4	0.56	80.5±0.5	43.5±1.3	0.53
CROPI	64.0±0.5	27.4±1.2	0.42	74.2±0.5	36.4±1.1	0.40	75.0±0.4	37.6±1.1	0.41	81.2±0.3	44.8±1.0	0.39
BANDITSCHED	**64.8**±0.5	**28.3**±1.1	**0.35**	**75.8**±0.4	**38.7**±1.0	**0.32**	**76.4**±0.4	**39.5**±0.9	**0.33**	**82.6**±0.3	**46.8**±0.9	**0.30**

Table 2: **Code and logic reasoning results** (Pass@1). CD-OOD and Zebra-OOD denote the hardest difficulty levels held out during training. We report results for Qwen2.5-7B and Qwen3-8B; results on the smaller models follow the same trends, as shown in Appendix E.

Method	Qwen2.5-7B					Qwen3-8B				
	LCB	CD-ID	CD-OOD	Zeb-ID	Zeb-OOD	LCB	CD-ID	CD-OOD	Zeb-ID	Zeb-OOD
Full-GRPO	22.4±0.8	68.3±1.2	29.1±2.0	54.6±1.5	21.8±1.9	28.6±0.7	74.2±1.0	35.4±1.8	61.3±1.3	27.5±1.7
DEPO	22.1±0.9	67.8±1.3	28.5±2.2	53.9±1.6	21.2±2.1	28.2±0.8	73.6±1.1	34.8±1.9	60.7±1.4	26.9±1.8
SEC	22.6±0.7	68.9±1.1	30.4±1.8	55.2±1.4	22.6±1.8	28.9±0.6	74.8±0.9	36.2±1.6	62.0±1.2	28.3±1.6
E2H	22.2±0.8	68.5±1.2	29.8±1.9	54.1±1.5	21.5±2.0	28.4±0.7	74.0±1.0	35.6±1.7	61.0±1.3	27.2±1.7
CROPI	23.0±0.7	69.2±1.0	31.2±1.7	55.8±1.3	23.4±1.7	29.4±0.6	75.4±0.9	37.0±1.5	62.8±1.1	29.1±1.5
BANDITSCHED	**24.1**±0.6	**70.6**±0.9	**34.2**±1.5	**57.4**±1.1	**25.7**±1.5	**30.8**±0.5	**77.1**±0.8	**40.6**±1.3	**64.9**±1.0	**31.8**±1.3

Baselines. We compare the following methods: (1) **Full-GRPO**: training on the full dataset with dynamic sampling (Yu et al., 2025), which filters groups with pass rate 0 or 1. (2) **Random-20%**: uniform random selection of 20% of prompts per step. (3) **DEPO** (Tang et al., 2025): offline pass-rate filtering with online explorability-based filtering and replay. (4) **SEC** (Chen et al., 2025b): Self-Evolving Curriculum using sliding-window UCB over problem categories. (5) **E2H-Reasoner** (Parashar et al., 2025): a probabilistic easy-to-hard curriculum that fades out easy tasks. (6) **CROPI** (Zhu et al., 2025): multi-stage RL with off-policy influence-guided data selection. Appendix E.3 additionally compares **Band-Filter** and **Band-Filter + greedy value**, which use the same EMA pass-rate band without a contextual posterior.

Evaluation Metrics. We report: (1) **Pass@1**: greedy-decoded accuracy. (2) **Avg@32**: average accuracy over 32 sampled responses (temperature $= 1.0$, top-$p = 0.95$). (3) **Pass@32**: the fraction of problems with at least one correct response among 32 samples, measuring success coverage at a fixed sampling budget. (4) **Rollout efficiency**: online rollouts to a common target of 95% of Full-GRPO's final accuracy, divided by Full-GRPO's rollouts to that same target; lower is better. (5) **Reasoning diversity**: entropy over reasoning-pattern clusters in correct solutions, distinct from the question-space diversity bonus. We also report wall-clock time cost results (Appendix E.2).

Implementation Details. We use Qwen2.5-{3B, 7B} (Yang et al., 2024) and Qwen3-{4B, 8B} (Yang et al., 2025) as base models. Training uses GRPO with DAPO-style decoupled clipping ($\epsilon_l = 0.2$, $\epsilon_h = 0.28$) and token-level loss (Yu et al., 2025). We sample $G = 16$ responses per prompt with a maximum generation length of 8192 tokens, a training batch of $B = 128$ prompts, and $T = 400$ gradient steps. Under this budget, uniform selection visits each DAPO-Math-17K prompt about three times on average. We use AdamW (Loshchilov & Hutter, 2019) with learning rate 10^{-6} and linear warmup over 20 steps. For BANDITSCHED, we use $(p_l, p_h) = (0.2, 0.8)$, $(\alpha, \beta, \gamma) = (0.5, 0.3, 0.2)$, $W = 50$, $\eta_p = 0.3$, $\lambda = 1$, and $d_e = 64$, and frontier probing uses $K = 10$, $B_{\text{probe}} = 8$, and $\tau_{\text{sim}} = 0.7$. These defaults are selected on Qwen2.5-3B/MATH and reused across all models and domains. Appendix D provides the remaining details on initialization, feature scaling, the embedding projection, and probe accounting. Experiments use 8×H100-80G GPUs and verl (Sheng et al., 2025). Accuracy results are mean ± standard deviation over 3 seeds.

4.2 MAIN RESULTS

Mathematical reasoning. Table 1 reports final accuracy and target rollout ratios across four models. BANDITSCHED achieves the highest mean accuracy in every setting while reaching the common

target with 30–35% of Full-GRPO's online rollouts, compared with 39–48% for the most efficient baselines, CROPI and DEPO. On MATH-500, Qwen3-8B obtains 82.6% Pass@1, compared with 81.4% for Full-GRPO and 81.2% for CROPI. On AIME 2024, Qwen2.5-7B obtains 38.7 Avg@32 versus 36.2 for Full-GRPO. The MATH-500 improvements over both Full-GRPO and CROPI are significant at the 0.05 level for the three larger models, while the Qwen2.5-3B and AIME differences are smaller relative to seed variation (Appendix E.8). Full math runs achieve approximately $2.1\times$ generation-time and $1.8\times$ total-time speedups (Appendix E.2).

Code and logic reasoning. Table 2 reports LiveCodeBench, Countdown, and Zebra results for Qwen2.5-7B and Qwen3-8B. BANDITSCHED achieves the highest mean accuracy across all these settings, showing that its benefits extend beyond mathematics. The gains are largest on the held-out difficulty levels. On Countdown OOD, BANDITSCHED improves over Full-GRPO by 5.2 points with Qwen3-8B and 5.1 points with Qwen2.5-7B, compared with 2.9 and 2.3 points on the in-distribution split. This pattern suggests that tracking the evolving competence frontier is particularly beneficial when test problems are harder than those seen during training. The corresponding rollout ratios on code and logic range from 0.30 to 0.37 (Appendix E.2).

4.3 ABLATION STUDIES

Table 3 examines the scheduler's components. Removing contextual features causes the largest drop (-2.4 on both MATH and AIME), highlighting the value of sharing information across prompts. Removing the reasoning-pattern embedding alone reduces MATH by 1.7 points, more than removing the competence block (1.2 points; Appendix E.6). Disabling frontier probing reduces MATH/AIME by 1.2/1.5 points, and removing the diversity bonus reduces them by 0.8/1.2 points. Using the full history instead of a sliding window lowers accuracy by 1.5/1.9 points, confirming the benefit of forgetting stale observations. Replacing Thompson Sampling with ϵ-greedy or UCB also lowers accuracy. The pass-rate filtering controls in Appendix E.3 provide a closer comparison: Band-Filter and greedy band filtering reach the target with rollout ratios of 0.44 and 0.41, versus 0.35 for BANDITSCHED. Pass-rate filtering thus accounts for a substantial part of the efficiency gain, and contextual scheduling reduces the rollout count by a further 14.6% relative to greedy filtering while improving accuracy by 0.9/1.2 points. Initialization, simplified configurations, embedding alternatives, and reward models are analyzed in Appendix E.

Table 3: **Ablation study** on Qwen2.5-3B, MATH-500 (Pass@1) and AIME (Avg@32). Rows remove individual components or replace Thompson Sampling with another rule. Results are mean $\pm$ standard deviation over 3 seeds.

Variant	MATH-500	AIME
BANDITSCHED (full)	**64.8**$\pm$0.5	**28.3**$\pm$1.1
w/o contextual features (vanilla TS)	62.4$\pm$0.7	25.9$\pm$1.4
w/o reasoning-pattern embedding	63.1$\pm$0.6	26.6$\pm$1.3
w/o frontier probing	63.6$\pm$0.6	26.8$\pm$1.2
w/o diversity bonus ($\gamma = 0$)	64.0$\pm$0.5	27.1$\pm$1.2
w/o sliding window (full history)	63.3$\pm$0.7	26.4$\pm$1.3
Replace TS with ϵ-greedy	63.5$\pm$0.6	26.7$\pm$1.2
Replace TS with UCB	63.9$\pm$0.5	27.0$\pm$1.1

4.4 FURTHER ANALYSIS

Training dynamics. Figure 2(a) shows MATH-500 learning curves for Qwen2.5-3B and Qwen3-4B. Full-GRPO and E2H plateau around steps 150–250, while BANDITSCHED continues improving as its selection keeps adapting to the model's changing competence. A small late-training dip, also observed for Full-GRPO and E2H, coincides with more generations reaching the 8192-token cap and being truncated (Appendix E.9).

Table 4: **Pass@k and reasoning diversity** on MATH-500.

	Qwen2.5-7B				Qwen3-8B			
Method	P@1	P@8	P@32	Div.↑	P@1	P@8	P@32	Div.↑
Full-GRPO	74.6	84.2	89.8	2.31	81.4	89.6	93.4	2.28
DEPO	73.9	83.8	89.4	2.24	80.8	89.1	93.0	2.21
SEC	73.1	84.6	90.2	2.38	81.0	89.8	93.6	2.35
CROPI	74.2	84.4	90.0	2.29	81.2	89.5	93.3	2.26
BANDITSCHED	**75.8**	**86.1**	**91.6**	**2.52**	**82.6**	**91.2**	**95.1**	**2.49**

Data utilization. Figure 2(b) shows that selection gradually shifts from easier to harder questions while retaining some easy problems. Removing the diversity bonus reduces Pass@1 by 0.8 points and Pass@32 by 1.8 points, indicating that broader question coverage benefits both accuracy and sampling coverage. Selection coverage and initialization are analyzed in Appendix E.5.

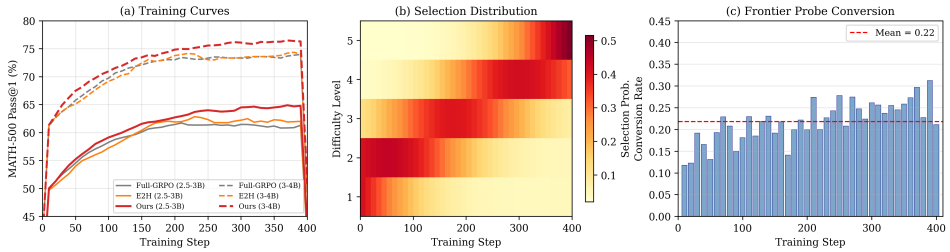

Figure 2: **Training dynamics analysis** with Qwen2.5-3B (solid) and Qwen3-4B (dashed). **(a)** MATH-500 accuracy over steps. **(b)** Distribution of selected problem difficulty for Qwen2.5-3B, showing a shift toward harder questions while retaining some easy problems. **(c)** Frontier conversion rate: fraction of probed questions whose estimated pass rate exceeds p_l within 20 steps.

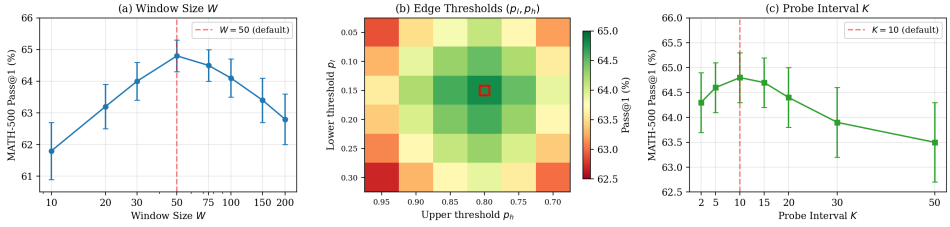

Figure 3: **Sensitivity analysis** on Qwen2.5-3B, MATH-500 Pass@1. The panels show MATH-500 Pass@1 as a function of **(a)** the sliding window size W, **(b)** the competence-edge thresholds (p_l, p_h), and **(c)** the frontier probe interval K. Results on other models show consistent trends.

Frontier probing effectiveness. Figure 2(c) reports the fraction of probed questions whose estimated pass rate exceeds p_l within 20 steps. This conversion rate averages 23% and rises from about 15% at step 50 to 28% at step 350, suggesting that more frontier problems become reachable as training progresses. Probing randomly chosen low-success questions yields only 8% conversion (Appendix E.9), indicating that embedding similarity identifies problems about to become learnable.

Pass@k and reasoning diversity. Table 4 shows higher Pass@32 for BANDITSCHED: 91.6 versus 89.8 for Qwen2.5-7B, and 95.1 versus 93.4 for Qwen3-8B, compared with Full-GRPO. BANDITSCHED also attains higher entropy over reasoning-pattern clusters in correct solutions. Together, these results show that BANDITSCHED improves not only single-sample accuracy but also the coverage and diversity of correct solutions on MATH-500 dataset.

Sensitivity analysis. Figure 3 examines the main hyperparameters. Performance peaks near $W = 50$ with a broad plateau over $W \in [30, 100]$, and both smaller and larger windows perform worse, consistent with the estimation–drift trade-off in §3.5. The fitted ridge coefficients keep changing throughout training, with a mean change of 0.087 between consecutive steps (Appendix E.7), confirming that the utility relationship drifts. Competence thresholds are robust within $p_l \in [0.15, 0.25]$ and $p_h \in [0.75, 0.85]$, and probe intervals in $K \in [5, 20]$ also give a broad plateau. The shared configuration and simplified variants are described in Appendices D.3 and E.4.

5 CONCLUSION AND FUTURE WORK

We presented BANDITSCHED, a framework that casts RLVR data selection as a non-stationary contextual bandit and solves it with Thompson Sampling and frontier probing. By predicting prompt utility before rollout and keeping competence estimates up to date, BANDITSCHED allocates rollouts to the problems that currently provide the most learning signal. Experiments with four models from 3B to 8B across math, code, and logic show substantially lower online rollout costs and competitive final accuracy, with the largest improvements on hard and OOD tasks. On math reasoning, BANDITSCHED reaches the common accuracy target with 30–35% of Full-GRPO's rollouts and achieves an approximately $1.8\times$ full-run wall-clock speedup. Future work includes mixed-domain training, process reward models, richer utility representations, and larger-scale evaluation.

AI USE STATEMENT

In this work, we used generative AI tools only to edit and polish the writing of this paper in order to improve its readability. Specifically, after the authors had drafted the complete manuscript, we used an LLM to suggest local improvements to grammar, word choice, and sentence flow, and to flag passages that read awkwardly or ambiguously. We have not used generative AI tools for any of the tasks that require disclosure under the ICLR policy: the contextual bandit formulation, the design of the context features and the composite reward, the implementation of BanditSched and of all baselines, the experimental design, and the interpretation of the results were produced entirely by the authors. The synthetic Countdown and Zebra puzzle sets were produced by rule-based procedural generators rather than by an LLM, and no dataset cleaning, reformatting, or translation was carried out with AI assistance. Beyond the language editing described above, we also did not use generative AI tools for tasks with recommended disclosure, such as writing code, creating figures, or identifying related literature. We have reviewed every AI-assisted edit and verified that no technical claim, numerical result, or citation was introduced, removed, or altered in the process. We take full responsibility for the final content of this work, including all text, claims, and artifacts.

ETHICS STATEMENT

The authors have read and adhere to the ICLR Code of Ethics. This work studies data scheduling for reinforcement learning with verifiable rewards and does not involve human subjects, user studies, or the collection of any personal or sensitive information, so no IRB review was required. All training and evaluation data come from publicly available sources released for research use, namely DAPO-Math-17K, TACO, MATH-500, AIME 2024, Minerva Math, and LiveCodeBench, together with Countdown and Zebra puzzles generated procedurally by rule-based scripts; we use these datasets in accordance with their licenses and terms of use, and we release no new data containing personal information. The base models we train are openly released Qwen2.5 and Qwen3 checkpoints, and reported numbers are averaged over three random seeds with standard deviations to support faithful reporting. We note that BanditSched lowers the compute required to strengthen reasoning capabilities, which carries the usual dual-use risk that more capable reasoning models could be misused, and that the scheduler optimizes for learning value rather than social desirability, so it may amplify biases present in the training corpus or the reward function if these are left unexamined. We discuss these considerations, along with privacy and security implications and recommended mitigations such as dataset curation, subgroup evaluation, and red-teaming, in Appendix H. The authors declare no conflicts of interest or sponsorship that could have influenced this work.

REPRODUCIBILITY STATEMENT

We have made the following efforts to ensure that the results in this paper can be reproduced. The complete BanditSched procedure is given in Algorithm 1, with the context features, composite reward, posterior update, and pass-rate EMA update specified in Equations (4), (5), (7), (8), and (9), and the frontier probing rule in Equation (10); Appendix C additionally provides a data flow diagram that clarifies how the candidate pool, frontier set, and training batch interact during training. Section 4.1 lists the training sets, evaluation benchmarks, baselines, evaluation metrics, base models, and the full set of training and BanditSched hyperparameters, and Appendix D documents the reasoning-pattern embedding extraction procedure, feature scaling, initialization, and the computational cost of the scheduler. Section 4.1 also reports the hardware and software stack used for all runs, and every experimental number is reported as the mean and standard deviation over three random seeds. Anonymized source code for BanditSched is available at `https://anonymous.4open.science/r/BanditSched-B362`.

REFERENCES

Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International conference on machine learning*, pp. 127–135. PMLR, 2013.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

Peter Chen, Xiaopeng Li, Ziniu Li, Wotao Yin, Xi Chen, and Tianyi Lin. Exploration vs exploitation: Rethinking rlvr through clipping, entropy, and spurious reward. In *The 5th Workshop on Mathematical Reasoning and AI at NeurIPS 2025*, 2025a.

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

Xingwu Chen, Tianle Li, and Difan Zou. Reshaping reasoning in llms: A theoretical analysis of rl training dynamics through pattern selection. *arXiv preprint arXiv:2506.04695*, 2025c.

Kanishk Gandhi, Ayush K Chakravarthy, Anikait Singh, Nathan Lile, and Noah Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. In *Second Conference on Language Modeling*, 2025.

Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for switching bandit problems. In *International conference on algorithmic learning theory*, pp. 174–188. Springer, 2011.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, et al. Openthoughts: Data recipes for reasoning models. In *International Conference on Learning Representations*, volume 2026, pp. 108059–108130, 2026.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

Yu Huang, Zixin Wen, Yuejie Chi, Yuting Wei, Aarti Singh, Yingbin Liang, and Yuxin Chen. On the learning dynamics of rlvr at the edge of competence. *arXiv preprint arXiv:2602.14872*, 2026.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025.

Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pp. 661–670, 2010.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. Taco: Topics in algorithmic code generation dataset. *arXiv preprint arXiv:2312.14852*, 2023.

Huanyu Liu, Ge Li, Jia Li, Hao Zhu, Kechi Zhang, and Yihong Dong. Saturn: Sat-based reinforcement learning to unleash llms reasoning. *arXiv preprint arXiv:2505.16368*, 2025a.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Mathematical Association of America. Aime 2024 problems. `https://artofproblemsolving.com/wiki/index.php/2024_AIME_I`, 2024.

Shubham Parashar, Shurui Gui, Xiner Li, Hongyi Ling, Sushil Vemuri, Blake Olson, Eric Li, Yu Zhang, James Caverlee, Dileep Kalathil, et al. Curriculum reinforcement learning from easy to hard tasks improves llm reasoning. *arXiv preprint arXiv:2506.06632*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, 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. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Xinyu Tang, Zhenduo Zhang, Yurou Liu, Wayne Xin Zhao, Zujie Wen, Zhiqiang Zhang, and Jun Zhou. Towards high data efficiency in reinforcement learning with verifiable reward. *arXiv preprint arXiv:2509.01321*, 2025.

Xinyu Tang, Yuliang Zhan, Zhixun Li, Wayne Xin Zhao, Zhenduo Zhang, Zujie Wen, Zhiqiang Zhang, and Jun Zhou. Rethinking sample polarity in reinforcement learning with verifiable rewards. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2928–2954, 2026.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, et al. Reinforcement learning for reasoning in large language models with one training example. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Can Xie, Ruotong Pan, Xiangyu Wu, Yunfei Zhang, Jiayi Fu, Tingting Gao, and Guorui Zhou. Unlocking exploration in rlvr: Uncertainty-aware advantage shaping for deeper reasoning. *arXiv preprint arXiv:2510.10649*, 2025.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

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.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. In *Second Conference on Language Modeling*, 2025.

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.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a.

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*, 2025b.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun MA, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. In *Second Conference on Language Modeling*, 2025.

Jixiao Zhang and Chunsheng Zuo. Grpo-lead: A difficulty-aware reinforcement learning approach for concise mathematical reasoning in language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5642–5665, 2025.

Shimin Zhang, Xianwei Chen, Yufan Shen, Ziyuan Ye, and Jibin Wu. Relax: Reasoning with latent exploration for large reasoning models. *arXiv preprint arXiv:2512.07558*, 2025.

Erle Zhu, Dazhi Jiang, Yuan Wang, Xujun Li, Jiale Cheng, Yuxian Gu, Yilin Niu, Aohan Zeng, Jie Tang, Minlie Huang, et al. Data-efficient rlvr via off-policy influence guidance. *arXiv preprint arXiv:2510.26491*, 2025.

CONTENTS

1 Introduction **1**

2 Related Work **2**

3 Methodology **3**

3.1 Preliminaries: GRPO for Reasoning 3

3.2 RLVR Data Selection as a Non-Stationary Contextual Bandit 3

3.3 Thompson Sampling for Prompt Selection 4

3.4 Frontier Probing: Exploring the Reasoning Boundary 5

3.5 Design Rationale for the Sliding Window 5

4 Experiments **6**

4.1 Experiments Setup 6

4.2 Main Results 7

4.3 Ablation Studies 8

4.4 Further Analysis 8

5 Conclusion and Future Work **9**

A Reward-Signal Analysis and Sliding-Window Design **16**

A.1 Mean Absolute Group-Relative Advantage 16

A.2 A Heuristic Window-Length Trade-off 16

B Comparison with Filtering and Bucket-Based Scheduling **17**

C BANDITSCHED Data Flow Diagram **17**

D Additional Experimental Details **18**

D.1 Question Features and Context Scaling 18

D.2 Initialization, Statistic Updates, and Frontier Selection 18

D.3 Shared Hyperparameters and Probe Accounting 19

D.4 Analytical Cost Scaling 19

E Additional Experimental Results **19**

E.1 Extended Reasoning Accuracy Performance 19

E.2 Efficiency Metrics and End-to-End Accounting 21

E.3 Contextual Scheduling versus Strong Pass-Rate Filtering 22

E.4 Simplified Configurations 23

E.5 Initialization and Training-Set Coverage 23

E.6 Context Contributions and Embedding Alternatives 23

E.7 Reward-Model Fit and Fitted-Coefficient Drift 24
E.8 Statistical Analysis of Accuracy Differences 25
E.9 Frontier-Probe and Truncation Diagnostics 25

F Case Study **26**

G Limitation Analysis **26**

H Broader Impacts **27**

A Reward-Signal Analysis and Sliding-Window Design

A.1 Mean Absolute Group-Relative Advantage

The prompt-level statistic used by the scheduler is the mean *absolute* advantage,

$$\bar{A}_{i,t} := \frac{1}{G}\sum_{g=1}^{G}\left|\hat{A}_{i,t}^{(g)}\right|. \tag{14}$$

Taking absolute values before averaging is necessary, since the signed mean of the centered advantages in Eq. (1) is identically zero.

Let $q_{i,t} = G^{-1}\sum_g r_{i,t}^{(g)}$ denote the empirical success rate in the current rollout group, as distinct from the historical EMA $\hat{p}_{i,t}$. For binary verifier rewards,

$$\frac{1}{G}\sum_{g}|r_{i,t}^{(g)} - q_{i,t}| = 2q_{i,t}(1-q_{i,t}), \qquad \sigma(r_{i,t}) = \sqrt{q_{i,t}(1-q_{i,t})}. \tag{15}$$

Consequently, including the numerical stabilizer in Eq. (1),

$$\bar{A}_{i,t} = \frac{2q_{i,t}(1-q_{i,t})}{\max\{\sqrt{q_{i,t}(1-q_{i,t})},\epsilon\}}. \tag{16}$$

When the stabilizer is inactive, this reduces to $2\sqrt{q_{i,t}(1-q_{i,t})}$, which peaks at $q_{i,t} = 0.5$. At $q_{i,t} \in \{0,1\}$ it is zero. For $G = 16$, the values at $q = 1/16,\ 2/16,\ 4/16,\ 8/16,\ 12/16$, and $15/16$ are approximately 0.484, 0.661, 0.866, 1.000, 0.866, and 0.484, respectively.

This identity shows that the advantage term and the competence-band indicator in Eq. (5) are closely related, although they carry different information in practice. The advantage term uses the current group's success rate $q_{i,t}$, which is recent but noisy, whereas the band indicator uses the EMA $\hat{p}_{i,t}$, which is smoother but lags behind the current policy. Batch normalization further makes the advantage term relative to the other selected prompts. The two terms therefore provide complementary views of the same correctness statistics, and the diversity term adds information about question coverage.

The mean absolute advantage summarizes the reward-based learning signal available from a prompt. The policy-gradient norm additionally depends on score-function vectors, clipping, and token aggregation, and becomes available only during the update, whereas $\bar{A}_{i,t}$ can be computed as soon as the rollouts are verified.

A.2 A Heuristic Window-Length Trade-off

A shorter window limits the influence of stale observations, while a longer window provides more observations for estimating the reward model. We describe this trade-off with the proxy

$$\mathcal{J}(W) = adT\sqrt{\frac{\log(NT)}{W}} + bL\delta WT, \qquad 1 \le W \le T, \tag{17}$$

where $a, b > 0$ are problem-dependent constants. The first term represents estimation uncertainty and the second represents sensitivity to parameter drift. We use this expression as a design heuristic for the sliding-window, batched scheduler.

Treating $\log(NT)$ as independent of W, the unconstrained minimizer for $\delta > 0$ is

$$W_0 = \left(\frac{ad\sqrt{\log(NT)}}{2bL\delta}\right)^{2/3}. \tag{18}$$

The feasible choice clips W_0 to $[1, T]$ and rounds it to an integer. For an interior minimizer, substitution gives

$$\mathcal{J}(W_0) = O\Big(d^{2/3}(L\delta)^{1/3}T[\log(NT)]^{1/3}\Big). \tag{19}$$

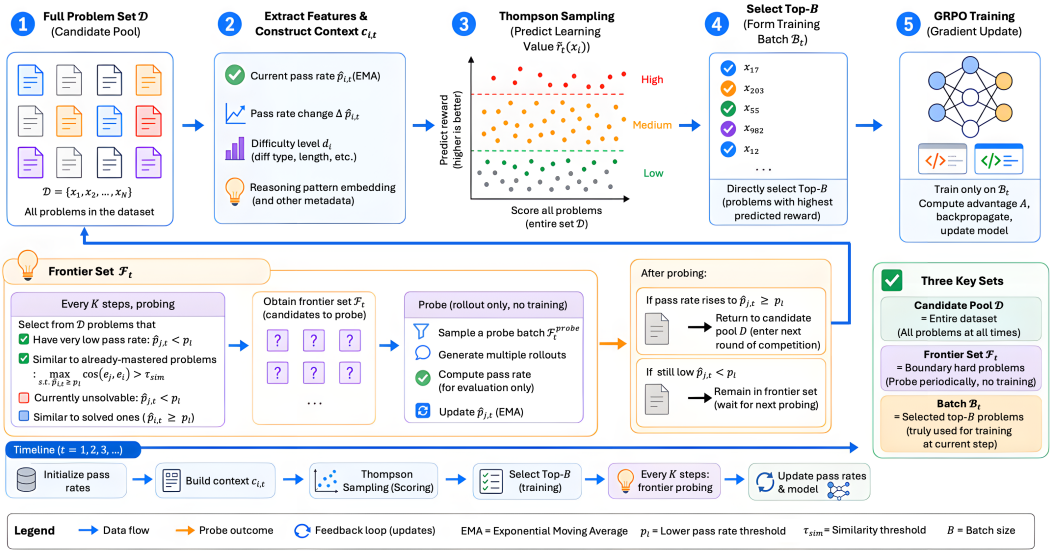

Figure 4: Data flow diagram of BANDITSCHED. The diagram highlights the roles of the candidate pool, frontier set, and training batch, and how they interact throughout training.

For fixed positive δ, the proxy is linear in T. A per-step bound $\|\boldsymbol{\theta}^*_{t+1} - \boldsymbol{\theta}^*_t\|_2 \leq \delta$ allows the reward parameter to change by $O(T)$ over training, so a sublinear dynamic-regret guarantee would require additional assumptions on the total variation of the parameter sequence. For $\delta = 0$, the proxy is minimized at $W = T$, recovering the stationary setting in which all observations should be retained.

The scaling $W_0 \propto \delta^{-2/3}$ motivates reducing the window as drift increases. Since its multiplicative constant is unknown, we select the default $W = 50$ empirically from the sensitivity study. The fitted-coefficient diagnostic in Appendix E.7 provides a complementary view of how the reward model changes over training.

B COMPARISON WITH FILTERING AND BUCKET-BASED SCHEDULING

Selection before rollout. Full-GRPO already includes DAPO-style dynamic sampling, which observes a rollout group's outcomes before discarding a prompt with uniform rewards. By that point, the generation cost has already been paid. BANDITSCHED instead predicts a prompt's utility from cached statistics and question features before generating the next rollout group, which reduces the number of uninformative groups generated in the first place. Frontier probes periodically refresh the statistics of previously attempted low-success prompts, so that predictions stay informative for problems that are rarely selected.

Within-bucket heterogeneity. Prompt-level context allows the scheduler to distinguish questions within a coarse difficulty or topic bucket. In the motivating analysis, the standard deviation of mean absolute advantage is 0.21 within difficulty buckets, compared with 0.09 between bucket means. Most of the variation in learning signal therefore lies within buckets, where a scheduler that samples uniformly inside a chosen bucket cannot exploit it. The controlled filtering comparison in Appendix E.3 quantifies the practical benefit of this finer granularity.

C BANDITSCHED DATA FLOW DIAGRAM

In this appendix, we provide a detailed data flow diagram of BANDITSCHED to clarify how different components interact during training and how data flows through the system.

Figure 4 presents a complete view of the pipeline. At a high level, BANDITSCHED operates over the entire dataset at every step and performs dynamic selection through a contextual bandit mechanism, while periodically probing a separate frontier set to anticipate future learning opportunities.

The main training process is illustrated in the top row of the figure. At each training step t, the method considers the full problem set $\mathcal{D} = \{x_1, \ldots, x_N\}$ as the candidate pool. For every problem x_i, a context vector $\mathbf{c}_{i,t}$ is constructed, which includes the current pass rate estimate $\hat{p}_{i,t}$, its temporal change, difficulty-related features, and a reasoning-pattern embedding. Based on these features, Thompson Sampling is used to predict a learning value $\tilde{r}_t(x_i)$ for each problem. All problems are scored in this way, and the top-B problems with the highest predicted values are selected to form the training batch $\mathcal{B}_t$. The model is then updated using GRPO on this batch only, where advantage signals are computed from rollout outcomes and used to perform gradient updates.

The bottom portion of the figure shows the frontier probing mechanism. Every K training steps, BANDITSCHED probes a small set of previously attempted questions with low estimated success rates that are close to currently learnable questions in the fixed embedding space. These rollouts update the corresponding pass-rate statistics without a policy-gradient update, and the refreshed statistics enter the context vectors and scores at subsequent steps.

A key design principle illustrated by the diagram is that all problems remain in the candidate pool throughout training. There is no hard filtering or removal of data. Instead, selection is entirely driven by predicted learning value, meaning that problems with low utility are implicitly deprioritized rather than explicitly excluded. This soft selection mechanism allows BANDITSCHED to continuously adapt to the evolving competence of the model and smoothly shift its focus toward the most informative training signals.

D Additional Experimental Details

D.1 Question Features and Context Scaling

We compute each question embedding before RLVR training by mean-pooling the base model's final-layer hidden states for the problem statement. A linear projection to $d_e = 64$ dimensions is trained on a proxy task using 2K labeled training problems with difficulty and topic annotations. The encoder and projection are then frozen. The embedding depends only on the question, so it is available before any rollout and uses neither generated responses nor reference solutions. The frontier rule in Eq. (10) uses the projected embedding $\mathbf{e}_i$, whose cosine similarities are unchanged by normalization.

Including an intercept, the context has dimension $d = 3 + 1 + d_e + 1 = 69$. We use the scaled representation

$$\mathbf{c}_{i,t} = \left[\hat{p}_{i,t}, \Delta\hat{p}_{i,t}, 4\hat{\sigma}^2_{i,t}, \widetilde{d}_i, \widetilde{\mathbf{e}}_i^\top, 1\right]^\top, \tag{20}$$

where $\hat{p}_{i,t} \in [0,1]$, $\Delta\hat{p}_{i,t} \in [-1,1]$, $\hat{\sigma}^2_{i,t} \in [0, 0.25]$, $\widetilde{d}_i \in [0,1]$ is normalized difficulty metadata when available, and $\widetilde{\mathbf{e}}_i$ has unit norm. These ranges imply $\|\mathbf{c}_{i,t}\|_2 \le \sqrt{6}$. Normalizing the embedding keeps its contribution to the context norm independent of its dimension.

The diversity bonus uses the cosine distance between a question embedding and the centroid of question embeddings selected in the current epoch, and thus encourages coverage of the question space.

D.2 Initialization, Statistic Updates, and Frontier Selection

We initialize the regression prior and prompt statistics as

$$\hat{\boldsymbol{\mu}}_0 = \mathbf{0}, \qquad \hat{\boldsymbol{\Sigma}}_0 = \lambda^{-1}\mathbf{I}, \qquad \hat{p}_{i,0} = 0.5, \qquad \Delta\hat{p}_{i,0} = 0, \qquad \hat{\sigma}^2_{i,0} = 0, \tag{21}$$

with $\lambda = 1$. No initial rollout sweep of the training set is performed. At the first step, the competence features are identical, so the relative ranking is induced by a random linear functional of the question and difficulty features; the shared intercept does not affect the relative order.

For every observed prompt, whether in a training batch or a probe batch, the pass-rate EMA is updated with the observed group success rate according to Eq. (9), with $\eta_p = 0.3$. Unobserved prompts retain their previous statistics, and the initial value 0.5 serves as a neutral starting point before any observation.

Table 5: Analytical arithmetic and storage estimates for the linear scoring stage with $d = 69$. Memory is for one FP32 context matrix, in decimal MB.

Number of questions	Scoring FLOPs ($\approx 2Nd$)	Context memory
17K	2.35×10^6	4.7 MB
100K	1.38×10^7	27.6 MB
1M	1.38×10^8	276.0 MB

The frontier rule in Eq. (10) requires at least one previous observation, $n_{j,t} \geq 1$, where $n_{j,t}$ counts the rollout groups observed for question x_j before frontier identification. This rule governs probe selection only, and main-batch Thompson Sampling continues to score the entire training set. Coverage of the training set arises from shared context, posterior uncertainty, and the diversity bonus; Appendix E.5 reports the resulting fraction of questions selected over training.

D.3 Shared Hyperparameters and Probe Accounting

The default configuration is selected on Qwen2.5-3B/MATH and reused across the four models and three domains:

$$\begin{gathered} W = 50, \quad K = 10, \quad B_{\text{probe}} = 8, \quad \tau_{\text{sim}} = 0.7, \\ (p_l, p_h) = (0.2, 0.8), \quad (\alpha, \beta, \gamma) = (0.5, 0.3, 0.2), \quad \lambda = 1, \quad \eta_p = 0.3. \end{gathered} \tag{22}$$

The sensitivity study shows broad plateaus, with less than 0.5 percentage-point variation over $W \in [30, 100]$ and $K \in [5, 20]$.

For the implemented $B = 128$, $B_{\text{probe}} = 8$, and $K = 10$, the extra number of probe responses relative to the nominal training-response count is

$$\frac{B_{\text{probe}} G}{KBG} = \frac{8}{10 \cdot 128} = 0.00625 = 0.625\%. \tag{23}$$

The corresponding fraction of time can differ because response lengths and execution efficiency vary across questions; Appendix E.2 reports the measured probe time.

D.4 Analytical Cost Scaling

Scoring N questions with a d-dimensional linear model requires $O(Nd)$ arithmetic operations. A dense factorization of the posterior precision matrix costs $O(d^3)$, and constructing its Gram matrix from a window of WB observations costs $O(WBd^2)$ when recomputed from scratch. These costs depend on the projected context dimension rather than on the language model's hidden-state dimension.

Table 5 lists these estimates for increasing dataset sizes. Even with one million questions, a scoring pass requires about 1.4×10^8 FLOPs and 276 MB of context memory. The estimates cover the linear scoring stage. Embedding extraction and projection training are one-time costs, and the similarity search used for frontier identification runs every K steps; for very large pools, approximate nearest-neighbor search can reduce the cost of this search.

E Additional Experimental Results

E.1 Extended Reasoning Accuracy Performance

Code and logic results for smaller models. Table 6 reports LiveCodeBench and logic puzzle results for the two smaller models (Qwen2.5-3B and Qwen3-4B), complementing the larger-model results in Table 2. The relative ordering of methods is consistent: BanditSched achieves the highest mean accuracy on both in-distribution and OOD splits. The OOD gains are slightly smaller on the 3B/4B models than on the 7B/8B models, likely because weaker base models have less latent capability for frontier probing to surface.

Table 6: Code and logic reasoning results for smaller models (Pass@1).

Method	Qwen2.5-3B					Qwen3-4B				
	LCB	CD-ID	CD-OOD	Zeb-ID	Zeb-OOD	LCB	CD-ID	CD-OOD	Zeb-ID	Zeb-OOD
Full-GRPO	14.8±1.0	58.6±1.5	18.4±2.3	42.1±1.8	12.6±2.2	21.2±0.8	66.4±1.2	27.3±2.0	52.8±1.5	20.4±1.9
DEPO	14.5±1.1	58.0±1.6	17.8±2.4	41.5±1.9	12.1±2.3	20.8±0.9	65.8±1.3	26.7±2.1	52.1±1.6	19.8±2.0
SEC	15.1±0.9	59.2±1.4	19.2±2.1	42.8±1.7	13.3±2.1	21.5±0.7	67.0±1.1	28.1±1.8	53.4±1.4	21.1±1.8
CROPI	15.4±0.8	59.6±1.3	19.8±2.0	43.4±1.6	13.9±2.0	22.0±0.7	67.6±1.0	28.8±1.7	54.2±1.3	21.8±1.7
BANDITSCHED	**16.3**±0.7	**61.2**±1.1	**22.1**±1.8	**45.6**±1.4	**15.8**±1.8	**23.2**±0.6	**69.4**±0.9	**31.5**±1.5	**56.6**±1.1	**24.2**±1.5

Minerva results. Table 7 reports Pass@1 on Minerva Math across all four models. BANDITSCHED achieves the highest mean accuracy on all four models, with the largest gain on Qwen3-8B (+1.6 over Full-GRPO). Minerva problems span undergraduate STEM topics beyond competition math, suggesting that BANDITSCHED's reasoning-pattern features capture transferable structure across mathematical sub-domains.

Table 7: Minerva Math results (Pass@1).

Method	Qwen2.5-3B	Qwen2.5-7B	Qwen3-4B	Qwen3-8B
Full-GRPO	32.4±0.9	41.8±0.7	42.6±0.7	49.2±0.6
DEPO	32.0±1.0	41.4±0.8	42.2±0.8	48.8±0.7
SEC	32.6±0.8	41.6±0.7	42.8±0.7	49.4±0.6
CROPI	32.8±0.7	42.2±0.6	43.0±0.6	49.6±0.5
BANDITSCHED	**33.6**±0.7	**43.1**±0.5	**43.8**±0.5	**50.8**±0.5

Sensitivity on Qwen3-4B. We replicate the sensitivity analysis from Figure 3 on Qwen3-4B. The optimal window size remains $W \approx 50$, the competence-edge thresholds are robust within the same ranges ($p_l \in [0.15, 0.25]$, $p_h \in [0.75, 0.85]$), and the probe interval sweet spot is $K \in [8, 15]$. This consistency across model families indicates that the default hyperparameters transfer without per-model tuning.

Per-difficulty-level breakdown on MATH-500. Table 8 reports Pass@1 stratified by MATH difficulty level (1–5) for Qwen2.5-7B. BANDITSCHED and Full-GRPO perform comparably on Levels 1–3 (easy to medium), where most methods already achieve high accuracy. The gains concentrate on Levels 4–5, where BANDITSCHED outperforms Full-GRPO by 2.4 and 3.1 points, respectively. This indicates that the bandit's primary value lies in routing compute toward the hardest problems at the right time. DEPO shows competitive Level-4 performance but lags on Level 5, likely because its static offline filter excludes hard problems that only become learnable later in training. SEC performs well on Level 5 thanks to its explicit hard-task scheduling, but its coarse difficulty buckets cause slight regressions on Level 3, where intra-bucket variance is high.

Table 8: Per-difficulty breakdown on MATH-500 (Pass@1, Qwen2.5-7B).

Method	Lv.1	Lv.2	Lv.3	Lv.4	Lv.5	All
Full-GRPO	95.2	91.4	82.6	68.3	48.7	74.6
DEPO	94.8	91.0	82.2	68.0	47.4	73.9
SEC	94.6	90.8	81.4	67.8	49.8	73.1
CROPI	95.0	91.2	82.4	68.6	49.2	74.2
BANDITSCHED	**95.4**	**91.8**	**83.2**	**70.7**	**51.8**	**75.8**

Reward component analysis. Table 9 disentangles the three components of the composite reward (Eq. (5)) by training with each component as the sole reward signal. Using only the normalized advantage (α-only) produces results close to DEPO, which also relies on advantage-like signals. The edge-of-competence indicator (β-only) performs surprisingly well as a standalone signal: it is a simple binary indicator yet captures the core intuition. The diversity bonus alone (γ-only) is the weakest individual signal, as expected, since it contains no information about the model's current competence. The full composite reward outperforms all single-component variants, confirming that the three signals are complementary.

Table 9: Reward component analysis on Qwen2.5-3B, MATH-500 (Pass@1) and AIME (Avg@32). Each row uses only one reward component.

Reward Signal	MATH-500	AIME
α-only (advantage)	63.4±0.6	26.5±1.3
β-only (edge indicator)	63.8±0.5	27.0±1.2
γ-only (diversity)	60.6±0.9	23.8±1.6
$\alpha + \beta$ (no diversity)	64.0±0.5	27.1±1.2
$\alpha + \beta + \gamma$ (full)	**64.8**±0.5	**28.3**±1.1

Table 10: Online rollouts to 95% of Full-GRPO's final accuracy, normalized by Full-GRPO's rollouts to the same target. Offline profiling is accounted for separately below.

	Qwen2.5-7B			Qwen3-8B		
Method	LCB	Countdown	Zebra	LCB	Countdown	Zebra
Full-GRPO	1.00	1.00	1.00	1.00	1.00	1.00
DEPO	0.49	0.47	0.50	0.46	0.44	0.47
SEC	0.56	0.53	0.55	0.52	0.50	0.53
CROPI	0.43	0.41	0.44	0.40	0.38	0.41
BANDITSCHED	**0.36**	**0.34**	**0.37**	**0.32**	**0.30**	**0.33**

E.2 Efficiency Metrics and End-to-End Accounting

Time-to-target rollout ratio. For each model and benchmark, let $a^\star$ be 95% of Full-GRPO's final accuracy. We report

$$E_{\text{target}}(m) = \frac{R_m(a^\star)}{R_{\text{Full-GRPO}}(a^\star)}, \tag{24}$$

where $R_m(a^\star)$ is the online rollout count required to reach the target. Lower is better. The reported math ratios of 0.30–0.35 correspond to approximately 2.9–$3.3\times$ fewer rollouts to this target.

Code and logic rollout efficiency. Table 10 applies the same target definition to the larger models on code and logic tasks. The ratios for BANDITSCHED range from 0.30 to 0.37 across these settings, again the lowest among all methods.

Full-run generation time and total training time. These measurements use the complete 400-step run rather than stopping at $a^\star$. Table 11 separates generation time, including probes, from total wall-clock time. Across the four math configurations, generation-time speedups are approximately 2.07–$2.12\times$, and total-time speedups are 1.79–$1.82\times$. The total-time speedup is smaller because policy-update time is similar across methods. Table 12 further breaks down the Qwen2.5-7B math run for all methods into generation, policy-update, and profiling or influence-estimation time. BANDITSCHED has the lowest total time, 11.9 hours, compared with 14.9 hours for CROPI and 16.5 hours for DEPO including offline filtering.

Table 11: Full-run time in hours on $8\times$H100-80G for 400 steps. In the BANDITSCHED generation column, the term after $+$ is probe time. Generation and total speedups are Full-GRPO time divided by BANDITSCHED time.

	Full-GRPO		BANDITSCHED		Speedup	
Model	Generation	Total	Generation	Total	Generation	Total
Qwen2.5-3B	12.6	14.7	$5.9 + 0.2$	8.2	$2.07\times$	$1.79\times$
Qwen2.5-7B	18.4	21.6	$8.4 + 0.3$	11.9	$2.11\times$	$1.82\times$
Qwen3-4B	14.8	17.3	$6.9 + 0.2$	9.6	$2.08\times$	$1.80\times$
Qwen3-8B	23.1	27.2	$10.5 + 0.4$	15.0	$2.12\times$	$1.81\times$

Offline profiling. BANDITSCHED does not use an offline rollout sweep to initialize pass rates. Its preprocessing consists of question-feature extraction and projection fitting, which involve forward

Table 12: Full-run time accounting for Qwen2.5-7B on the math task. The DEPO row includes the default offline filtering cost with $G_{\text{filter}} = 8$. CROPI's influence-estimation time is listed explicitly and counted once. All times are hours.

Method	Generation	Policy update	Profiling/influence	Total
Full-GRPO	18.4	3.2	0.0	21.6
DEPO	10.2	3.2	3.1	16.5
SEC	12.8	3.2	0.0	16.0
CROPI	9.6	3.2	2.1	14.9
BANDITSCHED	$8.4 + 0.3$	3.2	0.0	11.9

passes over the questions but no response generation. For DEPO, filtering $N = 17{,}000$ questions with G_{filter} responses each costs NG_{filter} generated responses before online training. Table 13 reports this cost for different values of G_{filter}, measured against the nominal budget of $TBG = 819{,}200$ training responses.

Table 13: DEPO offline profiling for Qwen2.5-7B. Shares use the nominal 400-step response budget. Totals add profiling time to the 13.4-hour online run. The default is $G_{\text{filter}} = 8$.

G_{filter}	Responses	Nominal share	Profiling hours	Total hours
4	68,000	8.3%	1.5	14.9
8	136,000	16.6%	3.1	16.5
16	272,000	33.2%	6.1	19.5

For a baseline with offline rollout cost R_m^{offline}, the corresponding inclusive target ratio is

$$E_{\text{target}}^{\text{inclusive}}(m) = E_{\text{target}}(m) + \frac{R_m^{\text{offline}}}{R_{\text{Full-GRPO}}(a^\star)}. \tag{25}$$

Evaluating this ratio requires Full-GRPO's rollout count at the target, rather than the full-run nominal budget. For CROPI, the influence-estimation time is already included in its total time in Table 12.

Additional domain-level timing. On code and logic, a supplementary timing comparison on 8×H100 gives 19.8 versus 11.1 hours and 20.5 versus 11.3 hours for Full-GRPO and BANDITSCHED, respectively, corresponding to $1.78\times$ and $1.81\times$ total-time speedups.

E.3 CONTEXTUAL SCHEDULING VERSUS STRONG PASS-RATE FILTERING

To separate contextual scheduling from the effect of restricting training to prompts with intermediate success rates, we compare against two additional baselines. **Band-Filter** samples uniformly among questions whose EMA pass rates lie in $[0.2, 0.8]$. **Band-Filter + greedy value** uses the same band and ranks eligible questions by $2\sqrt{\hat{p}_{i,t}(1-\hat{p}_{i,t})}$. Neither baseline uses a contextual posterior. Table 14 compares them with Full-GRPO, BANDITSCHED, and three ablated variants of BANDITSCHED.

Table 14: Filtering and scheduling comparison on Qwen2.5-3B. Accuracy is mean ± standard deviation over three seeds. Eff. is the online rollout ratio to the common target defined in Appendix E.2.

Method	MATH-500	AIME	Eff.↓
Full-GRPO + DAPO	63.8 ± 0.6	26.8 ± 1.4	1.00
Band-Filter	63.5 ± 0.6	26.7 ± 1.3	0.44
Band-Filter + greedy value	63.9 ± 0.6	27.1 ± 1.2	0.41
Non-contextual TS	62.4 ± 0.7	25.9 ± 1.4	0.60
BANDITSCHED w/o question embedding	63.1 ± 0.6	26.6 ± 1.3	0.46
BANDITSCHED w/o frontier probing	63.6 ± 0.6	26.8 ± 1.2	0.39
BANDITSCHED	$\mathbf{64.8 \pm 0.5}$	$\mathbf{28.3 \pm 1.1}$	**0.35**

Band filtering accounts for a substantial fraction of the improvement over Full-GRPO: it reduces the target rollout ratio to 0.44 while keeping accuracy close to Full-GRPO. The full scheduler further reduces the target rollout count by approximately 20.5% relative to Band-Filter ($1 - 0.35/0.44$) and 14.6% relative to greedy band filtering ($1 - 0.35/0.41$), while improving mean accuracy over greedy band filtering by 0.9 on MATH-500 and 1.2 on AIME. Without the question embedding, the ratio of BANDITSCHED rises to 0.46, and without frontier probing to 0.39. Non-contextual Thompson Sampling reaches only 0.60, which is less efficient than both filtering baselines. The efficiency gains therefore come from the contextual model rather than from posterior sampling alone.

E.4 SIMPLIFIED CONFIGURATIONS

We examine two simplified variants that remove the diversity reward ($\gamma = 0$) and the difficulty feature. The first tunes only W and K, and the second keeps $W = 50$ and $K = 10$ without additional tuning. All other settings follow the default configuration. As shown in Table 15, the fixed lite configuration already matches Full-GRPO in accuracy, while the full configuration achieves the highest mean, exceeding the fixed lite variant by 0.9 points on MATH-500 and 1.3 points on AIME.

Table 15: Final accuracy of simplified configurations on Qwen2.5-3B, over three seeds.

Configuration	MATH-500	AIME
BANDITSCHED full	64.8 ± 0.5	28.3 ± 1.1
Lite, tune only W, K	64.1 ± 0.6	27.1 ± 1.2
Lite, fixed $W = 50, K = 10$	63.9 ± 0.6	27.0 ± 1.0
Full-GRPO + DAPO	63.8 ± 0.6	26.8 ± 1.4

E.5 INITIALIZATION AND TRAINING-SET COVERAGE

We compare the default prior-driven start with three alternatives: uniform sampling for the first T_0 steps, zeroing the embedding block at the first step only, and replacing one selected question per batch with a uniformly sampled training question. Table 16 shows that all variants stay within 0.4 points of the default on MATH-500 and 0.5 points on AIME, and none of them improves the mean accuracy. The prior-driven start therefore requires no separate warm-up phase in this setting.

Table 16: Initialization and exploration variants on Qwen2.5-3B. Results are mean $\pm$ standard deviation over three seeds.

Variant	MATH-500	AIME
Default prior-driven start	64.8 ± 0.5	28.3 ± 1.1
Uniform warm start, $T_0 = 5$	64.7 ± 0.5	28.1 ± 1.1
Uniform warm start, $T_0 = 20$	64.4 ± 0.6	27.8 ± 1.2
Zero embedding block at $t = 1$	64.7 ± 0.5	28.2 ± 1.1
One uniform question per batch	64.5 ± 0.5	27.9 ± 1.1

The fraction of unique training questions selected by BANDITSCHED at steps 50, 100, and 400 is 8.9%, 14.7%, and 28.4%. Over the full run, approximately 4,820 of 17K questions are selected, compared with 14,200 (83.5%) for Full-GRPO and a fixed subset of 3,400 (20%) for DEPO. Only 41% of questions selected in the first 100 steps are also selected in the last 100 steps. Among those no longer selected in the last interval, 82% have estimated success rates above 0.85. Thus the selected set changes over time without removing questions from the candidate pool.

E.6 CONTEXT CONTRIBUTIONS AND EMBEDDING ALTERNATIVES

Table 17 summarizes three complementary diagnostics for each feature block: its share of the posterior coefficient norm, the decrease in predictive fit when the block is zeroed, and the drop in MATH-500 accuracy when the block is removed from training. The question embedding contributes most under all three measures, followed by the competence block, while the scalar difficulty feature contributes least.

Table 17: Context diagnostics on Qwen2.5-3B. Norm shares concern the three non-intercept feature blocks. ΔR^2 is the decrease in predictive fit when the corresponding block is zeroed; MATH drops are percentage points.

Block	Dim.	Norm share	ΔR^2	MATH drop
Competence	3	31.6%	0.118	1.2
Difficulty	1	6.8%	0.021	0.3
Question embedding	64	61.6%	0.164	1.7
All context removed	–	–	–	2.4

We also compare alternative question representations in Table 18. Precision@10 measures same-topic neighborhood retrieval and serves as an offline diagnostic of representation quality, while the accuracy columns come from RL training runs.

Table 18: Embedding alternatives on Qwen2.5-3B. P@10 denotes same-topic precision among the ten nearest neighbors. MATH-500 and Countdown OOD accuracies are mean $\pm$ standard deviation over three seeds.

Representation	Dim.	P@10 ID	P@10 OOD	MATH-500	CD-OOD
None	0	–	–	63.1 ± 0.6	18.9 ± 2.1
Raw mean-pooled hidden state	2048	0.71	0.60	63.7 ± 0.6	19.8 ± 2.0
PCA	64	0.74	0.63	64.1 ± 0.6	20.5 ± 1.9
Frozen linear proxy head	64	0.82	0.74	$\mathbf{64.8 \pm 0.5}$	$\mathbf{22.1 \pm 1.8}$
Sentence encoder	384–768	0.78	0.69	64.5 ± 0.5	21.4 ± 1.8
Head updated during RLVR	64	0.84	0.65	64.6 ± 0.6	20.9 ± 1.9

The frozen proxy head achieves the highest mean accuracy in this comparison. Updating the head during RLVR increases ID neighborhood precision from 0.82 to 0.84 but decreases OOD precision from 0.74 to 0.65, and its Countdown OOD accuracy drops accordingly. Keeping the projection frozen thus preserves neighborhoods that transfer to held-out difficulty levels.

A response-conditioned embedding variant obtains 64.2 ± 0.6 on MATH-500 and 27.4 ± 1.2 on AIME, below the question-only default (64.8 ± 0.5 and 28.3 ± 1.1). It also requires generating a response before an unobserved question can be scored, whereas the question-only representation is available before any rollout.

E.7 Reward-Model Fit and Fitted-Coefficient Drift

To examine the linear surrogate, we refit candidate reward models on logged context–reward pairs and measure the Spearman correlation between fitted scores and realized composite rewards. Table 19 reports these offline correlations together with the MATH-500 accuracy of RL runs that use each model for online selection.

The nonlinear models yield slightly higher rank correlations but similar MATH-500 accuracy, at a noticeably greater overhead. The linear model therefore offers the best trade-off between predictive quality and cost in this setting, with a residual RMSE of 0.083.

For the sliding-window ridge fits, we record

$$\hat{\delta}_t = \left\| \hat{\boldsymbol{\theta}}^{\mathrm{ridge}}_{t+1} - \hat{\boldsymbol{\theta}}^{\mathrm{ridge}}_{t} \right\|_2 \tag{26}$$

on the scaled contexts. Over the 400-step run, its mean, median, and 90th percentile are 0.087, 0.079, and 0.142. This quantity reflects both changes in the underlying utility relationship and estimation noise from the changing set of windowed observations. Its persistence throughout training is consistent with the non-stationarity that motivates the sliding window.

Table 19: Reward-model alternatives on Qwen2.5-3B. Spearman ρ is computed from offline refits. Overhead follows the reported wall-clock comparison.

Reward model	Spearman ρ	MATH-500	Overhead
Random ranking	≈ 0	–	–
Linear	0.71	64.8 ± 0.5	$< 2\%$
Two-layer MLP posterior	0.74	64.9 ± 0.5	4.8%
Gaussian process, RBF	0.73	64.7 ± 0.6	13.2%

Table 20: Accuracy differences in percentage points and approximate two-sided Welch p-values. The second comparator is specifically CROPI. The Zebra OOD/Qwen2.5-7B row is also computed from the existing table's reported mean and standard deviation.

	vs. Full-GRPO		vs. CROPI	
Benchmark / model	Δ	p	Δ	p
MATH-500 / Qwen2.5-3B	1.0	0.093	0.8	0.122
MATH-500 / Qwen2.5-7B	1.2	0.034	1.6	0.014
MATH-500 / Qwen3-4B	1.2	0.034	1.4	0.013
MATH-500 / Qwen3-8B	1.2	0.017	1.4	0.005
AIME / Qwen2.5-7B	2.5	0.061	2.3	0.056
AIME / Qwen3-8B	2.2	0.057	2.0	0.062
LiveCodeBench / Qwen3-8B	2.2	0.014	1.4	0.038
Minerva / Qwen3-8B	1.6	0.025	1.2	0.042
Countdown OOD / Qwen3-8B	5.2	0.019	3.6	0.036
Countdown OOD / Qwen2.5-7B	5.1	0.027	3.0	0.085
Zebra OOD / Qwen3-8B	4.3	0.028	2.7	0.079
Zebra OOD / Qwen2.5-7B	3.9	0.052	2.3	0.155

E.8 Statistical Analysis of Accuracy Differences

Tests from summary statistics. Table 20 reports two-sided Welch tests computed from the reported means and standard deviations, with three runs per method. The p-values are approximate because the summaries are rounded, and they are unadjusted per-comparison values. On MATH-500, the improvements over both Full-GRPO and CROPI are significant at the 0.05 level for Qwen2.5-7B, Qwen3-4B, and Qwen3-8B, as are the LiveCodeBench, Minerva, and Countdown OOD improvements for Qwen3-8B. Against Full-GRPO, the Countdown and Zebra OOD gains are also significant except for Zebra OOD with Qwen2.5-7B ($p = 0.052$). The Qwen2.5-3B MATH-500 and AIME differences are smaller relative to seed variation, with p-values between 0.05 and 0.13.

Paired comparisons. For the two settings with seed-level values available, Table 21 gives the paired observations together with two-sided paired t-tests and exact Wilcoxon signed-rank tests. BanditSched improves over Full-GRPO for every seed in both settings, and the paired t-tests give $p < 0.01$. With only three pairs, the smallest attainable p-value of the exact signed-rank test is 0.25, which is reached in both settings.

Where the gains concentrate. The mean improvements on MATH levels 1–3 are 0.2, 0.4, and 0.6 points, compared with 2.4 and 3.1 points on levels 4–5, and the held-out difficulty splits of Countdown and Zebra show similarly larger gains. BanditSched has the highest mean accuracy in all 32 model–benchmark settings covered by the main and supplementary accuracy tables. Because these settings share models, training runs, and related benchmarks, we view this as consistent evidence rather than as 32 independent observations. The rollout-efficiency improvements and the larger gains on harder splits are therefore the most robust empirical findings.

E.9 Frontier-Probe and Truncation Diagnostics

Using the conversion criterion from the training-dynamics analysis, a probed question converts if its estimated success rate exceeds p_l within the following 20 training steps. Table 22 summarizes the

Table 21: Paired seed-level results. Triples use the same seed order within each row. Test values are recomputed from the displayed observations and are unadjusted for multiple comparisons.

Setting	Full-GRPO	BANDITSCHED	Paired t p	Exact Wilcoxon p
MATH-500 / Qwen3-8B	81.1 / 81.3 / 81.8	82.3 / 82.7 / 82.8	0.0091	0.25
AIME / Qwen2.5-7B	34.8 / 36.4 / 37.4	37.7 / 38.8 / 39.6	0.0069	0.25

Table 22: Frontier conversion rates for different probe configurations and subsets.

Probe configuration or subset	Conversion rate
Embedding-guided frontier, overall	23%
OOD-adjacent frontier subset	21%
Random probing	8%

conversion rates. Embedding-guided probing converts 23% of probed questions overall and 21% on the OOD-adjacent subset, nearly three times the 8% rate of random probing. Embedding similarity therefore concentrates probes on questions that are about to become learnable.

Late-training generation limits. Inspection of the training logs associates the slight decline in Pass@1 around steps 380–400 with more generations reaching the 8192-token cap. The decrease is approximately 0.7 percentage points, and a similar pattern appears for Full-GRPO and E2H. Reporting truncation rates alongside accuracy thus helps interpret late-training curves.

F CASE STUDY

Case 1: A positive frontier probe. Consider a frontier question involving modular arithmetic and the Chinese Remainder Theorem. The probe produces two correct responses out of 16, revealing nonzero success on a question the model had previously failed. Because $q = 2/16 = 0.125$ lies below $p_l = 0.2$, a single probe does not move the question into the competence band: with $\eta_p = 0.3$ and any previous EMA below 0.2,

$$\hat{p}_{\text{new}} = 0.7\hat{p}_{\text{old}} + 0.3(0.125) < 0.1775 < 0.2. \tag{27}$$

The probe nevertheless updates the EMA with evidence of nonzero success and refreshes the context used at subsequent steps. Since main-batch selection has no hard pass-rate gate, the question can then be selected for training, and later observations determine when it enters the band.

Case 2: The diversity bonus. The Countdown example admits both search-based and factorization-based solution strategies. Because the scheduler uses a fixed question embedding, different responses to the same question share the same diversity term. The bonus therefore acts at the level of question coverage: it steers selection toward under-represented regions of the question space, and a broader distribution of generated strategies can emerge indirectly from training on more varied questions. The aggregate diversity and Pass@k results in Table 4 reflect this combined effect.

G LIMITATION ANALYSIS

Statistical power. Each configuration uses three random seeds. Several accuracy improvements, particularly on AIME and for the smallest model, do not reach significance under per-comparison tests, and additional seeds would sharpen these comparisons. The rollout-efficiency improvements are more consistent across settings than the final-accuracy gains.

Reward modeling and drift. The linear model estimates a composite proxy for learning value, and nonlinear models attain slightly higher offline rank correlations. The window-length analysis provides a design heuristic rather than a regret guarantee, and extending sliding-window regret analysis to batched selection with evolving contexts is an interesting direction for future work.

Embedding quality and supervision. The projection uses difficulty and topic labels from 2K training questions, so its quality depends on the available annotations and the base model's repre-

sentations. The diversity bonus operates on questions and does not directly target diversity among responses to the same question.

Coverage and stale statistics. BANDITSCHED does not guarantee that every training question is sampled, and the statistics of unselected questions can become stale; frontier probing mitigates but does not eliminate this effect. With $G = 16$ binary outcomes, the empirical success rate has a standard deviation of up to $\sqrt{0.25/16} = 0.125$, which adds noise to scores and frontier membership near the thresholds.

Scale and domain scope. Our experiments cover models up to 8B parameters and training sets up to 17K questions, and training is performed separately within math, code, and logic. The scaling estimates for larger pools concern linear scoring and context storage rather than measured end-to-end cost. Mixed-domain scheduling, larger models, and substantially larger training pools remain to be evaluated.

H BROADER IMPACTS

This work studies data scheduling for reinforcement learning with verifiable rewards, with the goal of improving the efficiency and effectiveness of training large language models for reasoning. A potential positive societal impact of this work is that more efficient RLVR training may reduce the computational and financial cost of improving reasoning capabilities, making advanced reasoning models more accessible to academic labs and smaller organizations. By selecting training problems that are more informative for the model's current competence level, BANDITSCHED may also reduce unnecessary rollout computation and associated energy consumption. In addition, improved reasoning ability could benefit applications such as mathematical problem solving, code generation, scientific assistance, education, and decision-support systems, provided that such systems are deployed with appropriate human oversight.

At the same time, improving the efficiency of reasoning reinforcement learning may also have negative societal impacts. More capable reasoning models can be misused for malicious or harmful purposes, including generating persuasive disinformation, automating deceptive online behavior, creating fake profiles or synthetic content, assisting cyber abuse, or supporting surveillance and manipulation workflows. Because BANDITSCHED reduces the amount of rollout compute required to train reasoning models, it could lower the barrier for actors with limited resources to improve model capabilities, including in undesirable settings. The method is domain-agnostic and therefore could be applied not only to beneficial reasoning tasks but also to datasets or reward functions that encode harmful objectives.

There are also fairness, privacy, and security considerations. If the training data or verifiable reward functions contain biases, the proposed scheduling mechanism may amplify them by repeatedly selecting examples that appear to provide high learning value under the biased reward signal. This could lead to models that perform unevenly across demographic groups, languages, domains, or problem types, especially when downstream systems are used in high-stakes settings such as education, hiring, finance, or legal assistance. Privacy risks may arise if the training corpus contains sensitive or personal information, since more efficient data scheduling could increase the influence of certain examples during training. Security risks include the possibility that the scheduler may prioritize examples that improve capabilities relevant to vulnerability discovery, exploit generation, or automated code manipulation if such tasks are included in the training data.

To mitigate these risks, we recommend that BANDITSCHED be used together with careful dataset curation, privacy-preserving data handling, bias and subgroup performance evaluation, red-teaming, and restrictions on unsafe training objectives. Since the proposed method optimizes for learning value rather than social desirability, it should not be treated as a substitute for safety alignment, content filtering, human review, or deployment-time safeguards. Future work should investigate safety-aware data scheduling objectives that explicitly incorporate fairness, privacy, and misuse-risk constraints in addition to efficiency and accuracy.