EFFICIENT RLVR SCHEDULING VIA GRAPH-STRUCTURED ONLINE DIFFICULTY ESTIMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) improves the mathematical reasoning capabilities of large language models but relies on costly rollout exploration. Assigning the same exploration budget to samples with different difficulty levels is inefficient: easy samples may receive redundant rollouts, whereas difficult but learnable samples may receive too little exploration. Existing adaptive schedulers address this mismatch through curriculum-based sample selection or non-uniform rollout allocation based on estimated sample difficulty. However, obtaining reliable online difficulty estimates remains challenging: dedicated probing adds substantial generation overhead, whereas history-based estimators face a cold start with no initial observations and stale feedback, and typically ignore relations among samples. To address these limitations, we propose a plug-and-play graph-based online difficulty estimator for RLVR in mathematical reasoning that shares rollout feedback across related samples and continuously updates their difficulty estimates, mitigating cold start and staleness without dedicated probing. Specifically, we first construct a difficulty-aware sample graph based on semantic and reasoning similarities. Based on this graph, we introduce latent difficulty states and use a Potts prior to encourage neighboring samples to share the same state. We then employ a state-level Beta-Binomial model to aggregate the rollout outcomes associated with each state. Finally, we use an online mean-field variational algorithm to continuously update the latent-state assignments and state-level difficulty as new feedback arrives. Our framework can be integrated into sample-selection and rollout-allocation schedulers, enabling difficulty-adaptive exploration without dedicated probing. Experiments across multiple base models, RL schedulers, and mathematical reasoning benchmarks demonstrate that our framework achieves better performance under matched rollout budgets.[1]

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) improves the mathematical reasoning capabilities of large language models through automatically verifiable feedback (Shao et al., 2024; Yu et al., 2026; Zheng et al., 2025). Group-based algorithms, such as Group Relative Policy Optimization (GRPO), generate multiple responses for each training sample and compare their rewards to update the policy (Shao et al., 2024). These rollouts explore alternative solutions but also incur substantial generation cost (Guo et al., 2025; Yao et al., 2026).

However, A common practice in group-based RLVR methods is to assign the same number of rollouts to every selected example (Shao et al., 2024; Guo et al., 2025; Yao et al., 2026), despite differences in difficulty under the current policy. This might waste exploration on easy problems while leaving difficult but learnable ones insufficiently explored (Nguyen et al., 2026; Yao et al., 2026; Zeng et al., 2026). To address this mismatch, existing difficulty-aware schedulers use sample selection or non-uniform rollout allocation to decide which problems to explore or how much exploration each receives (Gao et al., 2026; Qu et al., 2026; Kong et al., 2026; Nguyen et al., 2026; Yao et al., 2026; Zeng et al., 2026). Crucially, both approaches rely on up-to-date difficulty estimates as the policy evolves, making low-cost online estimation essential for adaptive RLVR scheduling in mathematical reasoning.

[1] https://anonymous.4open.science/r/iclr2027_20422

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.

However, obtaining such up-to-date and reliable estimates remains challenging. Some researchers explicitly estimate success probabilities for train examples under the current policy by using additional rollout probing (Zeng et al., 2026; Yao et al., 2026; Team et al., 2025). Although this approach provides estimation, it incurs substantial computational overhead and introduces statistical uncertainty due to the limited number of stochastic rollouts[2]. Further, some researchers estimate success probabilities for train examples from outcomes collected during previous training steps (Nguyen et al., 2026; Do et al., 2026; Kong et al., 2026; Qu et al., 2026). Although this approach reduces the additional rollout overhead, it struggles with cold start and accuracy because historical results may introduce noise from the variance of stochastic outcomes, and the historical results may become outdated as the policy model evolves during training. Thus, these limitations motivate the following question: **How can we continuously estimate the evolving difficulty of all mathematical training examples at low cost as the policy evolves during RLVR training?**

We argue that the shared structure of mathematical examples offers useful evidence for this estimation task. Examples involving similar mathematical concepts and reasoning procedures can depend on overlapping model capabilities, making their rollout outcomes potentially informative about one another's difficulty. This motivates us to use rollout feedback from observed examples to inform the difficulty estimates of related examples with limited or no direct feedback. Thus, in this paper, we propose a graph-based online difficulty estimation framework for RLVR-based LLM training, which can be integrated into different RL training schedulers to improve their performance. Specifically, we first construct a difficulty-aware sample graph that connects samples with similar semantic and reasoning characteristics. Then, we model sample difficulty as graph-structured latent states and develop an online variational inference algorithm to update these states using rollout feedback collected during training. In this way, related samples can share historical evidence, enabling us to continuously estimate the success probability of all training examples without additional rollout probing. Finally, we integrate our estimator into sample selection or rollout allocation methods to improve RLVR training efficiency. Experiments across multiple base models and mathematical reasoning benchmarks demonstrate the accuracy and efficiency of our estimator, while integrations with three distinct RL schedulers demonstrate its versatility across scheduling paradigms.

Our contributions are as follows: (1) We are the first to formulate dynamic difficulty estimation in RLVR for LLMs as a graph-structured latent-variable inference problem. (2) We propose a low-cost online estimator that propagates rollout feedback among related samples and continuously estimates their success probabilities as the policy evolves. (3) We integrate our estimator into both sample selection and rollout allocation methods. Extensive experiments show that our method is cold start friendly and substantially reduces difficulty estimation overhead while consistently improving the downstream performance of RLVR schedulers.

2 Related Work

2.1 Adaptive Sampling and Rollout Allocation in RLVR

Curriculum learning order or reweight examples according to difficulty, competence, or learnability (Bengio et al., 2009; Mindermann et al., 2022). This principle is especially relevant to group-based RLVR, where prompts with nearly deterministic group rewards provide little group-relative learning signal (Shao et al., 2024; Yu et al., 2026; Zheng et al., 2025). Prompt-selection methods therefore use estimated difficulty, competence alignment, uncertainty, or reward dynamics to prioritize informative examples (Yu et al., 2026; Gao et al., 2026; Qu et al., 2026; Kong et al., 2026; Zheng et al., 2026; Bae et al., 2026). A complementary line allocates non-uniform rollout counts. GVM, VIP, and CurES use gradient-variance or stability objectives (Yao et al., 2026; Nguyen et al., 2026; Zeng et al., 2026), whereas recent methods use sequential uncertainty reduction, Bernoulli-variance proxies, or posterior hit utility (Xiong et al., 2025; Fang et al., 2026). Although their scheduling objectives differ, these methods require estimates of current prompt behavior. Our work is complementary: it provides a reusable estimator rather than another scheduling rule.

[2] An intuitive analogy: tossing a fair coin 4 times still has a 12.5% chance of coming up all heads or all tails.

2.2 Dynamic Difficulty Estimation for Reasoning LLMs

Difficulty depends not only on an example but also on the policy model. Item response theory explicitly models the interaction between latent ability and item difficulty and has been adapted to NLP evaluation (Lalor et al., 2016; Rodriguez et al., 2021). Training-dynamics methods instead characterize examples through quantities such as confidence and variability across epochs, or prioritize examples that remain learnable and useful (Swayamdipta et al., 2020; Mindermann et al., 2022). RLVR further requires these model-dependent estimates to track an evolving policy. Existing RLVR estimators follow three broad strategies. Model-based methods learn a value or scoring model to predict difficulty without rolling out every candidate (Gao et al., 2026). Probing-based methods estimate current success probabilities from additional on-policy generations, obtaining timely but costly and noisy signals (Yao et al., 2026; Zeng et al., 2026; Team et al., 2025). History-based methods reuse training feedback: MoPPS performs streaming Bayesian inference (Qu et al., 2026); CDAS aggregates historical performance discrepancies (Kong et al., 2026); GRESO exploits temporal regularity in rewards (Zheng et al., 2026); and VIP fits a lightweight predictor to recent rollouts (Nguyen et al., 2026). Complementary scheduling methods operate at different granularities: Re-Schedule derives a static structural-learnability score from offline reasoning trees for easy-to-hard curriculum scheduling (Wang et al., 2026), whereas ARRoL predicts final correctness from partial rollouts and prunes trajectories online to balance binary rewards (Xu et al., 2026). Learned predictors must remain calibrated under policy drift, probing and tree construction incur generation overhead, and historical observations may be sparse or stale. Our approach also draws on graph-based semi-supervised learning and probabilistic graphical models, using spectral structure, Potts priors, and mean-field inference to propagate sparse feedback across related samples; see Appendix G for further discussion. In contrast, our estimator couples related samples through an explicit graph, pools sparse evidence through shared latent difficulty states.

Our setting combines these ideas with policy-induced nonstationarity: rollout observations are sparse, while their generating policy changes throughout training. We construct a difficulty-aware graph, associate prompts with time-indexed latent states, aggregate binary outcomes through state-level Beta–Binomial factors, and carry the variational posterior across training steps. This yields sequential probabilistic reasoning over relational and evolving latent states, together with the success-probability estimates required by adaptive RLVR schedulers.

3 Method

3.1 Overview

We propose a graph-based online framework for estimating the evolving difficulty of training examples in RLVR. By leveraging historical rollout outcomes from related examples, our framework predicts the success probability and difficulty of each example under the current policy before each training step. As illustrated in Fig. 1, the framework consists of four sequential modules. First, the **Difficulty-aware Sample Graph** jointly encodes the semantic and difficulty features of training examples and constructs a sparse train set graph (Sec. 3.3). Second, the **Graph-Structured Latent Difficulty Model** maps the graph into latent difficulty states that can be dynamically updated by rollouts feedback, with examples in the same state sharing a common success probability (Sec. 3.4). Third, the **Online Mean-Field Variational Inference** updates the latent-state distribution of each example and the posterior success probability of each latent state (Sec. 3.5). Finally, we use the updated posterior to estimate the success probability and difficulty of candidate samples for the next training step (Sec. 3.6).

3.2 Problem Formulation

Let $\mathcal{D} = \{x_i\}_{i=1}^{N}$ be a training dataset of mathematical problems for RLVR, where x_i denotes the i-th problem. Following the actual training order, we let $\mathcal{O}_t$ represent the set of samples for which we generate rollouts at training step t. Under the current policy model π_t, each sample $i \in \mathcal{O}_t$ generates $n_{i,t}$ rollouts, and we denote the binary correctness outcome of the g-th rollout by $y_{i,t,g} \in \{0, 1\}$. We aggregate the rollout outcomes of the same sample into the number of successful rollouts, $s_{i,t} = \sum_{g=1}^{n_{i,t}} y_{i,t,g}$, and define its empirical success rate as $r_{i,t} = s_{i,t}/n_{i,t}$.

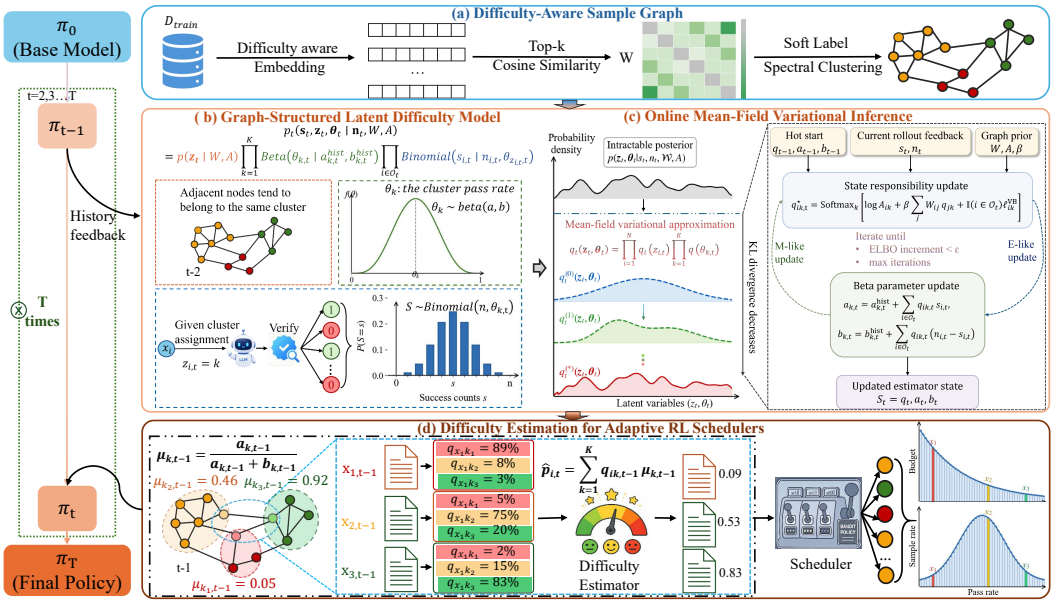


Figure 1: The overview of our framework. We first construct a difficulty-aware sample graph (a). During RL training, our graph-structured latent model aggregates rollout feedback across related samples (b), while online mean-field variational inference updates their evolving difficulty (c). The resulting estimates guide adaptive sample selection and rollout allocation (d).

Our goal is to accurately estimate the probability that the current policy correctly solves each sample before each RL training step: $\hat{p}_{i,t} = \Pr_{Y\sim\pi_t}(Y \text{ is correct} \mid x_i, \mathbf{s}_{<t}, \mathbf{n}_{<t})$. We then define the difficulty of sample i at step t as $d_{i,t} = 1 - p_{i,t}$. Finally, we feed these probability or difficulty estimates into existing RL scheduling frameworks to guide decisions such as sample selection and rollout allocation.

3.3 Difficulty-Aware Sample Graph

In this section, we construct a sparse similarity graph over the training dataset based on semantic and difficulty. This allows samples with few or no prior rollouts to estimate their current expected difficulty by leveraging historical feedback from semantically and difficulty-wise similar neighbors.

To encourage the encoder to capture semantic content, reasoning structure, and difficulty-related characteristics of each sample, we introduce a difficulty-aware instruction I_π: *Instruction: Given a prompt, encode it into a dense vector that captures both its semantic content and difficulty level, for the purpose of similarity comparison. \n Prompt: [SAMPLE].* We let f_{emb} denote a pretrained embedding model. The representation of sample x_i is computed as $\widetilde{\boldsymbol{\phi}}_i = f_{\text{emb}}(I_\pi, x_i)$ and normalized to $\boldsymbol{\phi}_i = \widetilde{\boldsymbol{\phi}}_i / \|\widetilde{\boldsymbol{\phi}}_i\|_2$. For each pair of samples, we measure their similarity with cosine similarity: $c_{ij} = \boldsymbol{\phi}_i^\top \boldsymbol{\phi}_j$.

To capture the underlying manifold structure of high-dimensional sample embeddings while reducing computational cost, we construct a k-nearest-neighbor graph by retaining the k_{nn} nearest neighbors of each node. We then retain only edges where the two nodes are mutual k-nearest neighbors and remove those with negative similarities, yielding a sparse undirected sample graph $W \in \mathbb{R}^{N\times N}$. Specifically, $W_{ij} = c_{ij}$ if x_i and x_j are mutual k-nearest neighbors and $c_{ij} > 0$, and 0 otherwise. A larger W_{ij} reflects greater similarity in semantic content, reasoning structure, and difficulty-related features between x_i and x_j.

3.4 Graph-Structured Latent Difficulty Model

In this section, we aim to perform statistical inference of sample difficulty by leveraging the graph structure and the historical rollout outcomes collected during previous RL training. Specifically, we

first construct a graph-structured prior over sample latent states, then model rollout feedback through shared state-level success probabilities, and finally derive the posterior distribution for subsequent difficulty estimation.

Graph Prior. Although W captures local similarity relationships among samples, it falls short in describing the overall difficulty structure of the training dataset. To further uncover similarities in both semantic features and difficulty levels across samples, we apply spectral clustering to the sparse undirected sample graph $W \in \mathbb{R}^{N\times N}$, and it yields an initial set of static cluster labels $c_i \in \{1,\dots,K\}$. However, spectral clustering relies solely on static sample representations, and its assignments can be affected by representation and clustering errors. More importantly, as we continuously update the policy during RL training, the difficulty of the same sample under the current policy may also change. Therefore, we convert each static cluster label into a smoothed prior distribution over latent states, we define $A_{ik} = \begin{cases} 1-\epsilon, & k = c_i, \\ \frac{\epsilon}{K-1}, & k \neq c_i, \end{cases}$. This conversion retains the structure from spectral clustering, and it also allows subsequent rollout outcomes to refine each sample's latent assignment. Then, inspired by Besag (1974), we introduce a Potts graph prior (Potts, 1952) over the latent assignments to retain the initial clustering structure while encouraging neighboring samples to share the same latent state. Specifically, we define $p(\mathbf{z}_t \mid W, A) = \frac{1}{Z(W,A)} \left(\prod_{i=1}^{N} A_{i,z_{i,t}}\right) \exp\left[\frac{\beta}{2} \sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}\, \mathbb{I}(z_{i,t} = z_{j,t})\right]$, where $\mathbf{z}_t = (z_{1,t},\dots,z_{N,t})$ denotes the vector of latent-state assignments at step t, with $z_{i,t} \in \{1,\dots,K\}$ indicating the assignment for sample x_i. The hyperparameter β controls the strength of graph propagation, and $Z(W,A)$ is the partition function.

State-Level Success Model. To statistically infer sample difficulty and latent-state assignments, we further integrate their relationship with historical rollout feedback into a Bayesian framework. However, estimating an independent success probability for each sample would still require a large number of sample-level rollouts to yield stable estimates. To address this and improve the efficiency of historical feedback utilization, we assume that samples assigned to the same latent state share a common state-level success probability. For each latent state k, following Zeng et al. (2026) and Qu et al. (2026), we model its success probability at step t with a Beta distribution: $\theta_{k,t} \sim \mathrm{Beta}\big(a_{k,t}^{\mathrm{hist}}, b_{k,t}^{\mathrm{hist}}\big)$. This distribution jointly captures the estimate of the state-level success probability while enabling sequential Bayesian updates. We set the prior parameters for step t by inheriting the posterior parameters from step $t-1$, i.e., $a_{k,t}^{\mathrm{hist}} = a_{k,t-1}$ and $b_{k,t}^{\mathrm{hist}} = b_{k,t-1}$. Both parameters are initialized to 1: $a_{k,0} = 1$, $b_{k,0} = 1$. Conditioned on the latent-state assignment $z_{i,t} = k$ and the conditional independence of rollout outcomes, the number of successes $s_{i,t}$ out of $n_{i,t}$ rollouts follows a binomial distribution: $(s_{i,t} \mid n_{i,t}, z_{i,t} = k, \theta_{k,t}) \sim \mathrm{Binomial}(n_{i,t}, \theta_{k,t})$. Let $\boldsymbol{\theta}_t = (\theta_{1,t},\dots,\theta_{K,t})$ collect the success probabilities of all latent states. By combining the graph-structured prior over $\mathbf{z}_t$, the independent Beta priors over each $\theta_{k,t}$, and the sample-level binomial likelihoods, we obtain the joint distribution at step t: $p_t(\mathbf{s}_t, \mathbf{z}_t, \boldsymbol{\theta}_t \mid \mathbf{n}_t, W, A) = p(\mathbf{z}_t \mid W, A) \prod_{k=1}^{K} \mathrm{Beta}\big(\theta_{k,t} \mid a_{k,t}^{\mathrm{hist}}, b_{k,t}^{\mathrm{hist}}\big) \prod_{i\in\mathcal{O}_t} \mathrm{Binomial}\big(s_{i,t} \mid n_{i,t}, \theta_{z_{i,t},t}\big)$.

Posterior Inference. From this joint distribution, we apply Bayes' rule to infer the posterior over the latent-state assignments and state-level success probabilities after observing the rollout outcomes. This posterior allows us to jointly update our beliefs about each sample's latent state and the shared success probabilities, which is essential for subsequent difficulty estimation. It factorizes as $p_t(\mathbf{z}_t, \boldsymbol{\theta}_t \mid \mathbf{s}_t, \mathbf{n}_t, W, A) = \frac{p_t(\mathbf{s}_t,\mathbf{z}_t,\boldsymbol{\theta}_t \mid \mathbf{n}_t, W, A)}{p_t(\mathbf{s}_t \mid \mathbf{n}_t, W, A)} = \frac{p_t(\mathbf{s}_t,\mathbf{z}_t,\boldsymbol{\theta}_t \mid \mathbf{n}_t, W, A)}{\sum_{\mathbf{z}_t} \int p_t(\mathbf{s}_t,\mathbf{z}_t,\boldsymbol{\theta}_t \mid \mathbf{n}_t, W, A)\, \mathrm{d}\boldsymbol{\theta}_t}$.

3.5 Online Mean-Field Variational Inference

In this section, we employ online mean-field variational inference to support sample difficulty estimation for adaptive LLM RL scheduling. Given the joint model in Section 3.4, we construct a tractable variational posterior over the sample assignments $\mathbf{z}_t$ and state-level success probabilities $\boldsymbol{\theta}_t$. Exact inference requires summing over K^N possible assignments because the Potts prior couples neighboring latent states. This exponential complexity makes exact inference impractical. To obtain a tractable approximation, we adopt a mean-field variational family, factorizing the posterior across samples and latent states as $q_t(\mathbf{z}t, \boldsymbol{\theta}t) = \prod i = 1^N q_i(zi, t) \prod_{k=1}^{K} q(\theta_{k,t})$. Here, $q_i(z_{i,t})$ represents the latent-state distribution of sample i. We denote its state-assignment

probability by $q_{ik,t} \equiv q_i(z_{i,t} = k)$. We restrict each state-level factor to the Beta family: $q(\theta_{k,t}) = \mathrm{Beta}(a_{k,t}, b_{k,t})$. Together, these factors define the variational family $\mathcal{Q}$. Inference then amounts to fitting two sets of parameters: the sample-assignment probabilities $q_{ik,t}$ and the state-level Beta parameters $(a_{k,t}, b_{k,t})$.

Within this family, we seek the distribution that minimizes the Kullback–Leibler (KL) divergence to the exact posterior: $q_t^\star = \arg\min_{q_t \in \mathcal{Q}} \mathrm{KL}\left[q_t(\mathbf{z}_t, \boldsymbol{\theta}_t) \,\|, p_t(\mathbf{z}_t, \boldsymbol{\theta}_t \mid \mathbf{s}_t, \mathbf{n}_t, W, A)\right]$. Because the normalized posterior is intractable, we instead maximize the evidence lower bound (ELBO):$\mathcal{L}_t(q_t) = \mathbb{E}_{q_t}[\log p_t(\mathbf{s}_t, \mathbf{z}_t, \boldsymbol{\theta}_t \mid \mathbf{n}_t, W, A)] - \mathbb{E}_{q_t}[\log q_t(\mathbf{z}_t, \boldsymbol{\theta}_t)]$. The log marginal likelihood satisfies $\log p_t(\mathbf{s}_t \mid \mathbf{n}_t, W, A) = \mathcal{L}_t(q_t) + \mathrm{KL}\left[q_t(\mathbf{z}_t, \boldsymbol{\theta}_t) \,\|, p_t(\mathbf{z}_t, \boldsymbol{\theta}_t \mid \mathbf{s}_t, \mathbf{n}_t, W, A)\right]$. Since the left-hand side does not depend on q_t, minimizing the KL divergence is equivalent to maximizing the ELBO: $q_t^\star = \arg\max_{q_t \in \mathcal{Q}} \mathcal{L}_t(q_t)$.

Under the mean-field factorization, the expected Potts interaction is $\mathbb{E}_{q_t}[\mathbb{I}(z_{i,t} = z_{j,t})] = \sum_{k=1}^K q_{ik,t} q_{jk,t}$. Substituting this expectation and the variational factors into the ELBO, and omitting terms independent of q_t, yields

$$\begin{aligned}\mathcal{L}_t(q) = (\beta/2) \textstyle\sum_{i=1}^N \sum_{j=1}^N W_{ij} \sum_{k=1}^K q_{ik,t} q_{jk,t} + \sum_{i=1}^N \sum_{k=1}^K q_{ik,t} \log A_{ik} - \sum_{i=1}^N \sum_{k=1}^K q_{ik,t} \log q_{ik,t} \\ + \textstyle\sum_{i \in \mathcal{O}_t} \sum_{k=1}^K q_{ik,t} \big\{ s_{i,t} [\psi(a_{k,t}) - \psi(a_{k,t} + b_{k,t})] + (n_{i,t} - s_{i,t}) [\psi(b_{k,t}) - \psi(a_{k,t} + b_{k,t})] \big\} \\ - \textstyle\sum_{k=1}^K \mathrm{KL}[\mathrm{Beta}(a_{k,t}, b_{k,t}) \| \, \mathrm{Beta}(a_{k,t}^{\mathrm{hist}}, b_{k,t}^{\mathrm{hist}})].\end{aligned}$$

Here, $\psi(\cdot)$ denotes the digamma function. The ELBO provides a common objective for fitting both parameter sets. We maximize it by alternating two types of coordinate updates. Proposition 1 derives the assignment update for $q_{ik,t}$, which adjusts each sample's state membership using graph neighbors and available rollout feedback. Proposition 2 derives the Beta update for $(a_{k,t}, b_{k,t})$, which pools successes and failures using the assignment probabilities as weights. These updates respectively determine how feedback is shared and how state-level success probabilities are estimated. Proposition 3 establishes ELBO monotonicity and convergence for the resulting coordinate-ascent procedure.Appendix A provides the proofs.

Proposition 1: Coordinate-Optimal Assignment Update. Fix $q(\boldsymbol{\theta}_t)$ and all $q_j(z_{j,t})$ for $j \neq i$. The coordinate-optimal update is $q_{ik,t}^\star = \mathrm{Softmax}_k[\log A_{ik} + \beta \sum_{j=1}^N W_{ij} q_{jk,t} + \mathbb{I}(i \in \mathcal{O}_t) \ell_{ik,t}^{\mathrm{VB}}]$, where $\ell_{ik,t}^{\mathrm{VB}} = s_{i,t} \left[\psi(a_{k,t}) - \psi(a_{k,t} + b_{k,t})\right] + (n_{i,t} - s_{i,t}) \left[\psi(b_{k,t}) - \psi(a_{k,t} + b_{k,t})\right]$.

Proposition 2: Coordinate-Optimal Beta Update. Fix $q(\mathbf{z}_t)$. The coordinate-optimal factor is $q^\star(\theta_{k,t}) = \mathrm{Beta}(a_{k,t}, b_{k,t})$, where $a_{k,t} = a_{k,t}^{\mathrm{hist}} + \sum_{i \in \mathcal{O}_t} q_{ik,t} s_{i,t}$, and $b_{k,t} = b_{k,t}^{\mathrm{hist}} + \sum_{i \in \mathcal{O}_t} q_{ik,t} (n_{i,t} - s_{i,t})$.

Proposition 3: Monotonicity and Convergence. At training step t, we keep $a_{k,t}^{\mathrm{hist}}, b_{k,t}^{\mathrm{hist}}$ fixed. Each coordinate update in Propositions 1 and 2 maximizes the ELBO with respect to the corresponding variational factor while holding the others fixed. Therefore, neither update decreases $\mathcal{L}_t(q)$ and the ELBO sequence converges.

The online algorithm executes these same two updates after each training step.After training step t produces rollout feedback for $\mathcal{O}_t$, we initialize inference with the previous estimator state: $q_{ik,t}^{(0)} = q_{ik,t-1}$, $a_{k,t}^{(0)} = a_{k,t-1}$, $b_{k,t}^{(0)} = b_{k,t-1}$.· We keep $a_{k,t}^{\mathrm{hist}}, b_{k,t}^{\mathrm{hist}}$ fixed during inference at step t. Each iteration first updates the sample assignments using Proposition 1, then updates the state-level Beta parameters using Proposition 2. The revised state estimates inform the assignment updates in the next iteration, allowing the two parameter sets to be refined together. We repeat this procedure until the ELBO increment falls below a preset threshold or the maximum iteration count is reached. The resulting state $\mathcal{S}_t = q_t, a_t, b_t$ combines sample memberships with state-level success estimates. Section 3.6 uses these quantities to predict sample difficulty for subsequent scheduling. We also use $\mathcal{S}_t$ to initialize inference at the next training step.

3.6 Difficulty Estimation for Adaptive RL Schedulers

In this section, we aggregate the state-level posterior from the previous training step into sample-level estimates of success probability, difficulty. These estimates then guide adaptive sample selection and rollout allocation. At the beginning of RL training step t, the estimator loads the state $\mathcal{S}_{t-1} = \{q_{t-1}, a_{t-1}, b_{t-1}\}$ from the previous step. For each latent state k, we compute the posterior

mean of its success probability as $\mu_{k,t-1} = \frac{a_{k,t-1}}{a_{k,t-1}+b_{k,t-1}}$, We first form a model-based prediction for sample x_i: $\widehat{p}_{i,t}^{\text{model}} = \sum_{k=1}^{K} q_{ik,t-1}\mu_{k,t-1}$. Then we adjust the prediction based on whether the sample has been rolled out before. If x_i has no prior rollouts, we directly use the model prediction: $\widehat{p}_{i,t} = \widehat{p}_{i,t}^{\text{model}}$. If it has been sampled in earlier steps, we average the model prediction with its historical success rate $\bar{r}_{i,<t}$: $\widehat{p}_{i,t} = \gamma\widehat{p}_{i,t}^{\text{model}} + (1-\gamma)\bar{r}_{i,<t})$ ($\gamma = 0.5$, see discussion in Appendix D). Also, the estimated difficulty is then $\widehat{d}_{i,t} = 1 - \widehat{p}_{i,t}$.

Overall, before training step t, the estimator predicts the difficulty of every sample. These outputs provide a plug-and-play interface for sample-selection and rollout-allocation schedulers that use sample-level success probabilities or difficulty estimates.

4 Experiment

4.1 Implementation Details

Setting. We conduct all training and inference on 4×A100 80G GPUs. Following Yao et al. (2026) and Zeng et al. (2026), we adopt Qwen 2.5 1.5B Math (Yang et al., 2024) and Llama 3 1B-Instruct (Grattafiori et al., 2024) as base models, we discuss it in Appendix. H.2. For the main experiments, we keep the environment strictly consistent with the original baselines. We integrate our framework as a plug-and-play component into the original code to demonstrate that it improves the final RL performance. For fairness, we use the same computational budget for all experiments, equivalent to one round of the standard GVM algorithm (∼1.5M rollouts, including pre-sampling). We use Qwen3-0.6B (Zhang et al., 2025) as the sentence embedding model, and set $K = 320$, $k_{\text{nn}} = 80$, $\gamma = 0.5$ and $\beta = 2$. Guidelines for hyperparameter or embedding model selection across different setting are provided in Appendix D.

Dataset. Since all baselines we integrate are based on mathematical problems, we follow the experimental settings of previous works (Yao et al., 2026; Gao et al., 2026; Zheng et al., 2026) and conduct reinforcement learning on NuminaMath (Li et al., 2024; Cui et al., 2025). We evaluate the resulting models on MATH500 (Hendrycks et al., 2021; Lightman et al., 2024), AIME 2024 (Balunovic et al., 2026), AIME 2025 (Dekoninck et al., 2026), and OlympiadBench (He et al., 2024). We provide detailed descriptions of these datasets in Appendix H.1.

Baseline. We evaluate our plug-and-play component on three representative baselines from different scheduling paradigms: GVM, a rollout-allocation method (Yao et al., 2026); PCL, a curriculum-learning method that trains a dedicated difficulty predictor (Gao et al., 2026); and GRESO, a curriculum-learning method that uses historical success rates as difficulty signals (Zheng et al., 2026). For each baseline, we replace only its original difficulty estimator with ours.

Metric. We use rule-based matching to judge whether the result matches the ground truth. Because a single evaluation on small and challenging datasets such as AIME can be sensitive to sampling randomness, we follow prior work (Yao et al., 2026; Zheng et al., 2026) and sample eight independent rollouts per prompt, reporting Average@8 accuracy.

4.2 Overall Performance

We integrate our estimator into three representative RL schedulers: GVM for rollout allocation, PCL for curriculum selection, and GRESO for selective rollout. We replace their original difficulty estimators while retaining their scheduling strategies, and compare all methods under the same total generation budget. "Windowed" denotes a variant that estimates difficulty using rollout feedback from only the recent steps; its modified inference algorithm is described in Appendix F.

As shown in Table 1, our estimator improves the overall performance of all three schedulers, with gains in most model–dataset settings. For GVM, GVM+Ours retains pre-sampling only for gradient-related estimation and replaces the original difficulty estimate with ours. It improves performance in seven of the eight settings and matches the baseline in the remaining one. We provide a more in-depth analysis in Section 4.3. When integrated into PCL and GRESO, our estimator also improves the corresponding baselines in most settings. The windowed variant generally underperforms the full-history version; we analyze possible reasons in Section 4.3. A plausible explanation for the

Table 1: The Average@8 accuracy of different RL scheduling methods on four mathematical reasoning benchmarks using two base models. The first column lists the original schedulers and their variants integrated with our difficulty estimator. Underlined values denote the highest accuracy within each group for each model and benchmark. Significance analysis is provided in Appendix C.

Method	Qwen-2.5-Math-1.5B (Average@8)				Llama-3.2-1B-Instruct (Average@8)			
	MATH500	AIME24	AIME25	Olym	MATH500	AIME24	AIME25	Olym
Base	39.3	3.30	3.33	25.2	17.5	0.00	0.00	3.02
GVM	71.9	11.3	11.3	35.8	23.9	0.42	0.00	4.59
GVM+Ours	74.7	16.7	13.3	38.3	25.7	3.30	0.00	4.89
GVM+Ours (Windowed)	74.3	14.6	13.3	37.9	25.4	2.92	0.42	4.81
PCL	59.7	7.92	4.58	26.1	26.7	0.42	0.00	5.04
PCL+Ours	61.6	11.3	6.67	27.1	27.4	0.00	0.42	5.81
PCL+Ours (Windowed)	61.4	10.8	7.50	27.6	26.9	0.83	0.00	5.72
GRESO	66.8	10.0	7.50	30.3	30.0	2.50	0.00	7.59
GRESO+Ours	68.2	10.0	10.0	31.9	32.6	6.67	0.42	8.26
GRESO+Ours (Windowed)	68.4	9.58	9.17	31.7	32.3	6.25	0.00	8.35

Table 2: Difficulty-estimation accuracy and overhead at different stages of GRPO training (batch size =256). MAE is computed at the sample level and Pearson correlation r at the batch level. Bold denotes the best overall result, underlining denotes the best result among low-cost methods (VIP, PCL, MoPPS, Ours, and Ours (Windowed)), and Time (including embedding, clustering and inference) denotes additional estimation overhead for over 3 epochs on 150K training samples.

Base Model	Methods	Early Steps		Middle Steps		Last Steps		Full Steps		Time
		MAE (↓)	r (↑)	MAE (↓)	r (↑)	MAE (↓)	r (↑)	MAE (↓)	r (↑)	A100*h
Qwen-2.5 Math-1.5B	SPO	0.136	0.544	0.117	0.886	0.090	0.935	0.115	0.826	∼29.6
	Sample before Step	**0.063**	**0.959**	**0.054**	**0.971**	**0.049**	**0.974**	**0.055**	**0.969**	∼45.3
	VIP	0.373	–	0.094	0.794	0.084	0.885	0.184	0.423	∼0.05
	PCL	0.404	0.138	0.336	0.407	0.328	0.474	0.356	0.281	∼7.36
	MoPPS	0.373	–	0.149	0.910	0.118	0.929	0.214	0.754	–
	Ours	0.290	0.482	0.133	0.912	0.123	0.943	0.183	0.836	∼0.12
	Ours (Windowed)	0.291	0.479	0.115	0.909	0.109	0.940	0.172	0.712	∼0.20
Llama-3.2 1B-Instruct	SPO	0.138	0.374	0.101	0.840	0.079	0.897	0.106	0.710	∼33.5
	Sample before Step	**0.044**	**0.986**	**0.046**	**0.967**	**0.046**	**0.967**	**0.045**	**0.986**	∼47.9
	VIP	0.413	–	0.107	0.895	0.084	0.880	0.201	0.717	∼0.05
	PCL	0.204	0.467	0.181	0.404	0.187	0.525	0.191	-0.231	∼4.38
	MoPPS	0.413	–	0.133	0.864	0.109	0.869	0.218	0.606	–
	Ours	0.197	0.469	0.100	0.868	0.075	0.913	0.124	0.776	∼0.12
	Ours (Windowed)	0.192	0.466	0.101	0.847	0.076	0.905	0.123	0.729	∼0.20

gains over the original schedulers is that the more stable difficulty estimates help the schedulers identify samples that are likely to yield both correct and incorrect responses within the same rollout group. Such samples produce nondegenerate group-relative advantages and potentially more informative optimization signals. Although full-history aggregation may introduce temporal lag, graph propagation and shared state-level statistics make its effect largely systematic across related samples, affecting absolute calibration more than relative difficulty ranking. Since adaptive schedulers primarily rely on ranking for sample selection and rollout allocation, this bias has limited impact on scheduling, as supported by the high correlations and downstream gains above.

Overall, these results suggest that our estimator can serve as a reusable component for different RL scheduling strategies and improve downstream performance under a matched generation budget.

4.3 Can Our Method Continuously Track Sample Difficulty During RL Training?

We evaluate difficulty tracking along the same GRPO training trajectory. At each evaluation point, empirical success rates from 8 independent rollouts per sample under the corresponding policy serve as references. Table 2 reports sample-level MAE for calibration and batch-level Pearson correlation r (batch size 256) for ranking consistency, since sample-level empirical rates are discrete and sparse. Lower MAE and higher r indicate better estimation.

We compare two probing-based baselines: SPO (Xu & Ding, 2025), which probes before training and updates estimates using subsequent training feedback, and Sample Before Step (SBS), which

probes before every training step. Low-cost baselines include PCL (LLM-based prediction), VIP (history-based statistical estimation), and MoPPS (history-based dynamic modeling) (Qu et al., 2026). SBS achieves the best estimation accuracy at every stage but incurs 45–48 A100-hours of additional computation.

Among low-cost methods, our cumulative estimator achieves the highest early-stage correlations on both models, while the cumulative and windowed variants achieve the lowest early-stage MAE on Qwen and Llama, respectively. These results support cross-sample feedback transfer as a low-cost means of alleviating observation sparsity, enabling cold-start-friendly difficulty estimation without expensive pre-training probing. Across the full trajectory, our cumulative estimator achieves the highest low-cost correlations $0.836/0.776$ with minimal overhead, whereas the windowed variant achieves the lowest MAE $0.172/0.123$. Based on these results, we argue that although focusing on recent history improves pointwise estimation accuracy, the reduced amount of historical evidence may weaken order preservation, potentially explaining the windowed variant's generally lower downstream performance than the cumulative estimator in Table 1.

4.4 Ablation Study

Table 3: Full-trajectory ablation results for difficulty estimation on two base models.

Methods	Qwen-2.5 Math-1.5B		Llama-3.2 1B-Instruct	
	MAE (↓)	r (↑)	MAE (↓)	r (↑)
Ours	**0.183**	**0.836**	**0.124**	**0.776**
w/o Spectral Init.	0.218	0.710	0.197	0.686
w/o Prior Smoothing	0.197	0.783	0.144	0.736
w/o Graph Sparsification	0.187	0.835	0.134	0.742
w/o β	0.187	0.825	0.133	0.593

We ablate spectral initialization, prior smoothing, graph sparsification, and Potts coupling. Table 3 reports full-trajectory results. Replacing spectral initialization with random assignments substantially degrades estimation, indicating the importance of an informative initial structure. Removing prior smoothing also hurts performance, as hard assignments prevent rollout feedback from revising latent-state membership. Using a dense graph slightly worsens estimation, suggesting that sparsification reduces interference from weakly related samples. Finally, setting $\beta = 0$ degrades estimation quality, suggesting that rollout feedback from neighboring samples in the graph helps infer sample difficulty.

4.5 Does the Clustering Structure Capture Difficulty Throughout Training?

Following the setup in Section 4.3, we compare the within-cluster variance of historical difficulty against a stratified permutation baseline before training and at the end of each of the three epochs. The observed variance is lower in all eight model–checkpoint combinations, with Holm-adjusted $p = 0.0016$ when jointly correcting the main tests and sensitivity analyses. This indicates that the clusters consistently group samples with similar historical difficulty, supporting feedback sharing through latent states. Complementary analysis on MATH reveals associations with subject categories (Cramér's $V = 0.541$) and significant within-subject differences in annotated difficulty (all BH-adjusted $p \leq 9.151 \times 10^{-4}$), see more details in Appendix. E.

5 Conclusion

In this work, we address inefficient exploration-budget allocation in RLVR for mathematical reasoning through graph-based online difficulty estimation. Our framework connects samples with similar semantic content and reasoning structures and uses rollout feedback to track their evolving difficulty without dedicated probing. The resulting estimates support both sample-selection and rollout-allocation schedulers. Experiments across multiple base models, schedulers, and benchmarks show that our framework improves downstream mathematical reasoning performance in most settings under matched rollout budgets while adding little online computational overhead. These results demonstrate the potential of online difficulty estimation to support more efficient RLVR training.

AI USE STATEMENT

We used generative AI tools to check grammar and improve the clarity of our writing, and AI-assisted Deep Research tools to help identify potentially relevant prior work that we might have overlooked in our literature review. We also used generative AI tools for LaTeX formatting and template conversion. We carried out the mathematical derivations without AI assistance. We subsequently used multimodal AI tools to organize our handwritten derivations and transcribe the mathematical expressions into LaTeX to facilitate manuscript preparation. We used these tools only to present and transcribe existing derivations, not to develop mathematical arguments or proofs. We reviewed the AI-assisted text, checked the transcribed equations against our handwritten derivations, and verified the references included in the manuscript. We take full responsibility for the final content of this work, including its text, claims, results, references, and AI-assisted artifacts.

ETHICS STATEMENT

This work studies computationally efficient RLVR scheduling, with a primary focus on mathematical reasoning. The experiments use existing mathematical and programming benchmarks and pretrained language models. Our difficulty estimates describe the success probabilities of a particular model on individual problems; they are not assessments of human ability. Although improving training efficiency may reduce the resources required to achieve a given level of performance, it does not by itself address the broader risks of language models, including biased or unreliable outputs and potential misuse. Any reuse of the datasets, models, and associated software should respect their licenses and intended conditions of use.

REPRODUCIBILITY STATEMENT

Section 3 specifies the graph construction, probabilistic model, online inference procedure, and sample-level prediction rule. Appendix A provides the detailed derivations and proofs supporting the theoretical results, while Appendix F describes the windowed extension. The experimental configurations, baseline integrations, and evaluation protocols are described in Section 4, with dataset descriptions in Appendix H.1 and additional analyses of hyperparameters and embedding choices in Appendix D. Appendix C details the statistical analysis of downstream improvements.We provide our implementation in the anonymous supplementary code submission. For the updates in Propositions 1 and 2, our implementation uses matrix-based numerical approximations to improve computational efficiency. Despite these approximations, we empirically observe a non-decreasing ELBO across iterations in the evaluated experiments, together with a nearly $25.7\times \sim 50\times$ speedup over the direct implementation. Figure 2 provides a visual example. The implementation has been tested and runs successfully with verl version 0.7.0; compatibility and stability with other versions have not been evaluated.

REFERENCES

Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, JeongYeon Nam, and Donghyun Kwak. Online difficulty filtering for reasoning oriented reinforcement learning. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 700–719, 2026.

Mislav Balunovic, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *Advances in Neural Information Processing Systems*, 38, 2026.

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7(85):2399–2434, 2006. URL http://jmlr.org/papers/v7/belkin06a.html.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum Learning. In *International Conference on Machine Learning*, pp. 41–48, 2009. doi: 10.1145/1553374.1553380. URL https://mlanthology.org/icml/2009/bengio2009icml-curriculum/.

Julian Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):192–225, 1974.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017. doi: 10.1080/01621459.2017.1285773. URL https://doi.org/10.1080/01621459.2017.1285773.

Tamara Broderick, Nicholas Boyd, Andre Wibisono, Ashia Wilson, and Michael Jordan. Streaming variational bayes. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/51ef186e18dc00c2d31982567235c559-Paper.pdf.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2025. URL https://arxiv.org/abs/2402.03216.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: Matharena as an evaluation platform for mathematics with llms. *arXiv preprint arXiv:2605.00674*, 2026.

Wilfrid J. Dixon and Alexander M. Mood. The statistical sign test. *Journal of the American Statistical Association*, 41(236):557–566, 1946. doi: 10.1080/01621459.1946.10501898.

Dai Do, Manh Nguyen, Svetha Venkatesh, and Hung Le. SPaCe: Unlocking sample-efficient large language models training with self-pace curriculum learning. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 3480–3507, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.171. URL https://aclanthology.org/2026.findings-acl.171/.

Yangyi Fang, Jiaye Lin, Xiaoliang Fu, Cong Qin, Haolin Shi, Chaowen Hu, Lu Pan, Ke Zeng, and Xunliang Cai. How to allocate, how to learn? dynamic rollout allocation and advantage modulation for policy optimization. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 14727–14744, 2026.

Zhaolin Gao, Joongwon Kim, Wen Sun, Thorsten Joachims, Sid Wang, Richard Yuanzhe Pang, and Liang Tan. Prompt curriculum learning for efficient LLM post-training. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=zqOCacBD3P.

Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6): 721–741, 1984.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

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.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024.

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

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of machine learning research*, 2013.

Deyang Kong, Qi Guo, Xiangyu Xi, Wei Wang, Jingang Wang, Xunliang Cai, Shikun Zhang, and Wei Ye. Rethinking the sampling criteria in reinforcement learning for llm reasoning: A competence-difficulty alignment perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 31438–31446, 2026.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. Matryoshka representation learning. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/c32319f4868da7613d78af9993100e42-Abstract-Conference.html.

John P. Lalor, Hao Wu, and Hong Yu. Building an evaluation scale using item response theory. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 648–657, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1062. URL https://aclanthology.org/D16-1062/.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9, 2024.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Clare A McGrory, D Michael Titterington, Robert Reeves, and Anthony N Pettitt. Variational bayes for estimating the parameters of a hidden potts model. *Statistics and Computing*, 19(3):329–340, 2009.

Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, and Yarin Gal. Prioritized training on points that are learnable, worth learning, and not yet learnt. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 15630–15649. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/mindermann22a.html.

Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14, 2001.

Hieu Trung Nguyen, Bao Nguyen, Wenao Ma, Yuzhi Zhao, Ruifeng She, and Viet Anh Nguyen. Adaptive rollout allocation for online reinforcement learning with verifiable rewards. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Z5sWYACAop.

Renfrey Burnard Potts. Some generalized order-disorder transformations. In *Mathematical proceedings of the cambridge philosophical society*, volume 48, pp. 106–109. Cambridge University Press, 1952.

Yun Qu, Qi Wang, Yixiu Mao, Vincent Tao Hu, Björn Ommer, and Xiangyang Ji. Can prompt difficulty be online predicted for accelerating rl finetuning of reasoning models? In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 1240–1250, 2026.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 3982–3992. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Pedro Rodriguez, Joe Barrow, Alexander Hoyle, John P. Lalor, Robin Jia, and Jordan Boyd-Graber. Evaluation examples are not equally informative: How should that change NLP leaderboards? In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4486–4503, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.346. URL https://aclanthology.org/2021.acl-long.346/.

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

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9275–9293, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.746. URL https://aclanthology.org/2020.emnlp-main.746/.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

Hong Wang, Zhezheng Hao, Jian Luo, Chenxing Wei, Yao Shu, Lei Liu, Cheaterlin, Hande Dong, and Jiawei Chen. Scheduling your LLM reinforcement learning with reasoning trees. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=V4zln7XiJj.

Mingqi Wu, Zhihao Zhang, Qiaole Dong, Zhiheng Xi, Jun Zhao, Senjie Jin, Xiaoran Fan, Yuhao Zhou, Huijie Lv, Ming Zhang, et al. Reasoning or memorization? unreliable results of reinforcement learning due to data contamination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 33944–33952, 2026.

Wei Xiong, Chenlu Ye, Baohao Liao, Hanze Dong, Xinxing Xu, Christof Monz, Jiang Bian, Nan Jiang, and Tong Zhang. Reinforce-ada: An adaptive sampling framework for reinforce-style llm training. *arXiv e-prints*, pp. arXiv–2510, 2025.

Haobo Xu, Sirui Chen, Ruizhong Qiu, Yuchen Yan, Chen Luo, Monica Cheng, Jingrui He, and Hanghang Tong. Prune as you generate: Online rollout pruning for faster and better rlvr, 2026. URL https://arxiv.org/abs/2603.24840.

Zhongwen Xu and Zihan Ding. Single-stream policy optimization, 2025. URL https://arxiv.org/abs/2509.13232.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement, 2024. URL https://arxiv.org/abs/2409.12122.

Jiarui Yao, Yifan Hao, Hanning Zhang, Hanze Dong, Wei Xiong, Nan Jiang, and Tong Zhang. Optimizing chain-of-thought reasoners via gradient variance minimization in rejection sampling and rl. *Advances in Neural Information Processing Systems*, 38:163245–163284, 2026.

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. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Yongcheng Zeng, Zexu Sun, Bokai Ji, Erxue Min, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Haifeng Zhang, Xu Chen, and Jun Wang. CurES: From gradient analysis to efficient curriculum learning for reasoning LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=QXrZ0Y3yGJ.

Jun Zhang. The mean field theory in em procedures for markov random fields. *IEEE Transactions on signal processing*, 40(10):2570–2583, 1992.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models, 2025. URL https://arxiv.org/abs/2506.05176.

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

Haizhong Zheng, Yang Zhou, Brian Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for llm reasoning via selective rollouts. *Advances in Neural Information Processing Systems*, 38:124321–124346, 2026.

Dengyong Zhou, Olivier Bousquet, Thomas Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In S. Thrun, L. Saul, and B. Schölkopf (eds.), *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2003. URL https://proceedings.neurips.cc/paper_files/paper/2003/file/87682805257e619d49b8e0dfdc14affa-Paper.pdf.

Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pp. 912–919, 2003.

A ELBO DERIVATION AND PROOFS

This section derives the expanded evidence lower bound (ELBO) in Sec. 3.5 and proves Propositions 1–3. For compactness, define $\delta_{ik,t} = \mathbb{I}(z_{i,t} = k)$ and let C_t collect all terms that are independent of the variational distribution. The mutual-neighbor graph is undirected, so $W_{ij} = W_{ji}$, and it has no self-loops, so $W_{ii} = 0$.

A.1 DERIVATION OF THE EXPANDED ELBO

From the joint model in Sec. 3.4, its log density can be written, up to the constant C_t, as

$$\begin{aligned}\log p_t(\mathbf{s}_t, \mathbf{z}_t, \boldsymbol{\theta}_t \mid \mathbf{n}_t, W, A) =& C_t + \sum_{i=1}^{N}\sum_{k=1}^{K} \delta_{ik,t} \log A_{ik} + \frac{\beta}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij} \sum_{k=1}^{K} \delta_{ik,t}\delta_{jk,t} \\ &+ \sum_{k=1}^{K} \Big[(a_{k,t}^{\text{hist}} - 1)\log\theta_{k,t} + (b_{k,t}^{\text{hist}} - 1)\log(1-\theta_{k,t})\Big] \\ &+ \sum_{i\in\mathcal{O}_t}\sum_{k=1}^{K} \delta_{ik,t} s_{i,t} \log\theta_{k,t} \\ &+ \sum_{i\in\mathcal{O}_t}\sum_{k=1}^{K} \delta_{ik,t}(n_{i,t} - s_{i,t}) \times \log(1-\theta_{k,t}). \end{aligned} \tag{1}$$

Here, C_t includes $-\log Z(W, A)$, the Beta normalizers, and the binomial coefficients, all of which are fixed with respect to q_t.

Under the mean-field distribution,

$$\mathbb{E}_{q_t}[\delta_{ik,t}] = q_{ik,t}, \tag{2}$$

$$\mathbb{E}_{q_t}[\delta_{ik,t}\delta_{jk,t}] = q_{ik,t}q_{jk,t} \quad (i \neq j), \tag{3}$$

$$\mathbb{E}_{q_t}[\log\theta_{k,t}] = \psi(a_{k,t}) - \psi(a_{k,t} + b_{k,t}), \tag{4}$$

$$\mathbb{E}_{q_t}[\log(1-\theta_{k,t})] = \psi(b_{k,t}) - \psi(a_{k,t} + b_{k,t}). \tag{5}$$

The second identity is sufficient for the graph term because $W_{ii} = 0$. The negative entropy contribution factorizes as

$$-\mathbb{E}_{q_t}[\log q_t] = -\sum_{i=1}^{N}\sum_{k=1}^{K} q_{ik,t}\log q_{ik,t} - \sum_{k=1}^{K}\mathbb{E}_{q_t}[\log q(\theta_{k,t})].$$

Combining the expected log Beta prior with the corresponding Beta-factor entropy gives

$$\mathbb{E}_{q_t}\big[\log \text{Beta}(\theta_{k,t} \mid a_{k,t}^{\text{hist}}, b_{k,t}^{\text{hist}}) - \log q(\theta_{k,t})\big] = -\,\text{KL}\big[\text{Beta}(a_{k,t}, b_{k,t}) \,\big\|\, \text{Beta}(a_{k,t}^{\text{hist}}, b_{k,t}^{\text{hist}})\big]\,.$$

Substituting these identities into $\mathcal{L}_t(q_t) = \mathbb{E}_{q_t}[\log p_t] - \mathbb{E}_{q_t}[\log q_t]$ yields the expression below. To keep the display compact, let

$$D_{k,t}^{\text{Beta}} = \text{KL}\big[\text{Beta}(a_{k,t}, b_{k,t}) \,\big\|\, \text{Beta}(a_{k,t}^{\text{hist}}, b_{k,t}^{\text{hist}})\big]\,. \tag{6}$$

Then

$$\begin{aligned}\mathcal{L}_t(q_t) = &\frac{\beta}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}\sum_{k=1}^{K} q_{ik,t}q_{jk,t} + \sum_{i=1}^{N}\sum_{k=1}^{K} q_{ik,t}\log A_{ik} - \sum_{i=1}^{N}\sum_{k=1}^{K} q_{ik,t}\log q_{ik,t} \\ &+ \sum_{i\in\mathcal{O}_t}\sum_{k=1}^{K} q_{ik,t}\ell_{ik,t}^{\text{VB}} - \sum_{k=1}^{K} D_{k,t}^{\text{Beta}} + C_t,\end{aligned}$$

where

$$\ell_{ik,t}^{\text{VB}} = s_{i,t}\left[\psi(a_{k,t}) - \psi(a_{k,t} + b_{k,t})\right] + (n_{i,t} - s_{i,t})\left[\psi(b_{k,t}) - \psi(a_{k,t} + b_{k,t})\right].$$

Dropping C_t, which is independent of q_t, gives exactly the expanded ELBO reported in Sec. 3.5.

A.2 Proof of Proposition 1

Fix $q(\boldsymbol{\theta}_t)$ and every assignment factor except $q_i(z_{i,t})$. Because W is symmetric, the two appearances of node i in the double graph sum combine and cancel the factor $1/2$. The terms of Eq. (7) that depend on $q_{ik,t}$ are therefore

$$\mathcal{L}_t(q_i) = \sum_{k=1}^{K} q_{ik,t} h_{ik,t} - \sum_{k=1}^{K} q_{ik,t} \log q_{ik,t} + C, \tag{7}$$

$$h_{ik,t} = \log A_{ik} + \beta \sum_{j=1}^{N} W_{ij} q_{jk,t} + \mathbb{I}(i \in \mathcal{O}_t) \ell^{\mathrm{VB}}_{ik,t}. \tag{8}$$

Introduce a Lagrange multiplier λ_i for $\sum_{k=1}^{K} q_{ik,t} = 1$ and define

$$\widetilde{\mathcal{L}}_t(q_i) = \mathcal{L}_t(q_i) + \lambda_i \left(\sum_{r=1}^{K} q_{ir,t} - 1 \right). \tag{9}$$

Differentiating gives

$$\frac{\partial \widetilde{\mathcal{L}}_t(q_i)}{\partial q_{ik,t}} = h_{ik,t} - \log q_{ik,t} - 1 + \lambda_i. \tag{10}$$

Setting this derivative to zero and normalizing over k yields

$$q^{\star}_{ik,t} = \frac{\exp(h_{ik,t})}{\sum_{r=1}^{K} \exp(h_{ir,t})} = \mathrm{Softmax}_k(h_{ik,t}), \tag{11}$$

which is the update in Proposition 1.

A.3 Proof of Proposition 2

Fix $q(\mathbf{z}_t)$ and define $q_{-k,t} = q(\mathbf{z}_t) \prod_{r \neq k} q(\theta_{r,t})$. The standard mean-field coordinate identity gives

$$\log q^{\star}(\theta_{k,t}) = \mathbb{E}_{q_{-k,t}} \big[\log p_t(\mathbf{s}_t, \mathbf{z}_t, \boldsymbol{\theta}_t \mid \mathbf{n}_t, W, A) \big] + C.$$

Retaining only terms involving $\theta_{k,t}$ in Eq. (1) gives

$$\alpha_{k,t} = a^{\mathrm{hist}}_{k,t} + \sum_{i \in \mathcal{O}_t} q_{ik,t} s_{i,t}, \tag{12}$$

$$\gamma_{k,t} = b^{\mathrm{hist}}_{k,t} + \sum_{i \in \mathcal{O}_t} q_{ik,t} (n_{i,t} - s_{i,t}), \tag{13}$$

$$\begin{aligned} \log q^{\star}(\theta_{k,t}) = {} & (\alpha_{k,t} - 1) \log \theta_{k,t} \\ & + (\gamma_{k,t} - 1) \log(1 - \theta_{k,t}) + C. \end{aligned} \tag{14}$$

This is the log density of a Beta distribution. Hence

$$q^{\star}(\theta_{k,t}) = \mathrm{Beta}(a_{k,t}, b_{k,t}), \tag{15}$$

$$a_{k,t} = a^{\mathrm{hist}}_{k,t} + \sum_{i \in \mathcal{O}_t} q_{ik,t} s_{i,t}, \tag{16}$$

$$b_{k,t} = b^{\mathrm{hist}}_{k,t} + \sum_{i \in \mathcal{O}_t} q_{ik,t} (n_{i,t} - s_{i,t}), \tag{17}$$

which proves Proposition 2.

Interpretation of the posterior mean update.

From the perspective of the mean update, this mechanism can be viewed as an adaptive variant of the historical-mean discounting mechanisms employed by Qu et al. (2026) and Zheng et al. (2026). Its effective update weight is

$$1 - \lambda_{k,t} = \frac{N_{k,t}}{C_{k,t-1} + N_{k,t}},$$

where $\lambda_{k,t}$ is the retention weight assigned to the historical mean of state k at training step t, $C_{k,t-1} = a_{k,t-1} + b_{k,t-1}$ denotes the historical effective count, including the prior contribution, and $N_{k,t} = \sum_{i \in O_t} q_{ik,t} n_{i,t}$ denotes the responsibility-weighted number of rollouts assigned to that state at the current step. Here, $q_{ik,t}$ is the probability that sample i belongs to state k, and $n_{i,t}$ is its number of rollouts at the current step. This adaptive mechanism assigns greater weight to new feedback when historical observations are scarce, allowing the estimate to adjust rapidly. As historical observations accumulate, the state-level statistics gain greater observational support; for comparable amounts of new evidence, the update weight decreases accordingly, making the estimate more stable against sampling fluctuations at individual steps.

A.4 Sequential and Synchronous Implementations

Proposition 1 gives the exact coordinate-optimal update for a single assignment factor $q_i(z_{i,t})$ while all other factors are held fixed. We consider two implementations of this update.

Sequential reference implementation. Our reference implementation performs a strict sequential Gauss–Seidel sweep over the samples. Under the update order $i = 1, \ldots, N$, the assignment update at iteration $m + 1$ is

$$q_{ik,t}^{(m+1)} = \mathrm{Softmax}_k \left[\log A_{ik} + \beta \sum_{j<i} W_{ij} q_{jk,t}^{(m+1)} + \beta \sum_{j>i} W_{ij} q_{jk,t}^{(m)} + \mathbb{I}(i \in \mathcal{O}_t) \ell_{ik,t}^{\mathrm{VB},(m)} \right].$$

Thus, each update uses the most recently available values of the previously updated factors. After one complete assignment sweep, the Beta factors are updated according to Proposition 2. Because every assignment factor is optimized while the remaining factors are fixed, this implementation is an exact coordinate-ascent variational inference procedure. The monotonicity and convergence results in Proposition 3 apply to this sequential implementation.

Synchronous accelerated implementation. The strict sequential update is difficult to parallelize over samples. We therefore additionally implement a synchronous Jacobi-style update:

$$\widetilde{q}_{ik,t}^{(m+1)} = \mathrm{Softmax}_k \left[\log A_{ik} + \beta \sum_{j=1}^{N} W_{ij} q_{jk,t}^{(m)} + \mathbb{I}(i \in \mathcal{O}_t) \ell_{ik,t}^{\mathrm{VB},(m)} \right]. \tag{18}$$

All assignment factors in Eq. (18) are computed in parallel from the same previous iterate, after which the Beta factors are updated using Proposition 2.

Unlike the sequential implementation, a synchronous assignment sweep is not an exact block-coordinate maximization of the ELBO. Consequently, Proposition 3 does not directly guarantee that every synchronous sweep is ELBO non-decreasing. We treat the synchronous implementation as a parallel approximation to the theoretically grounded sequential procedure.

To evaluate this approximation, we compare the converged sample-level success-probability estimates produced by the two implementations. We define their discrepancy at training step t as

$$\Delta_t = \max_{1 \le i \le N} \left| \widehat{p}_{i,t}^{\mathrm{sync}} - \widehat{p}_{i,t}^{\mathrm{seq}} \right|. \tag{19}$$

Across the evaluated training checkpoints, the discrepancy remains within 10^{-2}. At the same time, the synchronous implementation allows the graph-message aggregation and assignment updates to be executed efficiently in parallel, resulting in substantially better wall-clock performance (nearly $25.7\times \sim 50\times$ faster).

Our submitted code includes both the sequential reference implementation, which satisfies the ELBO monotonicity and convergence guarantees, and the synchronous accelerated implementation. Given the small discrepancies observed in our experiments, we recommend the accelerated implementation for real-time estimator updates in practical deployments. Alternatively, the sequential implementation can run asynchronously alongside GPU training on a separate CPU worker, for example, an Intel Xeon Gold 6148 CPU paired with A100 GPUs. In this configuration, training uses the latest completed estimator state while the CPU processes a fixed feedback snapshot using sequential coordinate updates. In our 4*A100 80G environment, the asynchronous implementation typically introduces a lag of one training step in estimator updates.

A.5 Proof of Proposition 3

We prove the result for the strict sequential implementation described in Sec. A.4. Fix a training step t. Throughout the inference procedure at this step, the historical parameters $(a_{k,t}^{\text{hist}}, b_{k,t}^{\text{hist}})$, the graph (W, A), and all rollout observations are fixed.

We assume that

$$A_{ik} > 0 \quad \text{for every } i, k, \tag{20}$$

$$a_{k,t}^{\text{hist}} > 0, \qquad b_{k,t}^{\text{hist}} > 0 \quad \text{for every } k, \tag{21}$$

$$|W_{ij}| < \infty, \qquad n_{i,t} < \infty. \tag{22}$$

The first condition is satisfied by the smoothed initialization when $0 < \epsilon < 1$. We further assume that the categorical factors are updated cyclically and that every factor is updated once in each complete sequential sweep.

Monotonicity. Fix all variational factors except one categorical factor $q_i(z_{i,t})$. Proposition 1 gives the unique maximizer of the ELBO with respect to this factor. Therefore, updating $q_i(z_{i,t})$ cannot decrease the ELBO.

Similarly, after fixing $q(\mathbf{z}_t)$ and all Beta factors except $q(\theta_{k,t})$, Proposition 2 gives the coordinate-optimal Beta factor. Updating this factor therefore cannot decrease the ELBO. Consequently, if $q_t^{(m)}$ denotes the variational distribution after the m-th coordinate update, then

$$\mathcal{L}_t(q_t^{(m+1)}) \geq \mathcal{L}_t(q_t^{(m)}). \tag{23}$$

Boundedness of the parameter sequence. Each categorical parameter belongs to the probability simplex:

$$(q_{i1,t}, \ldots, q_{iK,t}) \in \Delta_K. \tag{24}$$

The product of the N categorical simplexes is compact.

Define the total number of observed successes and failures at training step t as

$$S_t = \sum_{i \in \mathcal{O}_t} s_{i,t}, \tag{25}$$

$$F_t = \sum_{i \in \mathcal{O}_t} (n_{i,t} - s_{i,t}). \tag{26}$$

By Proposition 2 and $0 \leq q_{ik,t} \leq 1$, the Beta parameters satisfy

$$a_{k,t}^{\text{hist}} \leq a_{k,t} \leq a_{k,t}^{\text{hist}} + S_t, \tag{27}$$

$$b_{k,t}^{\text{hist}} \leq b_{k,t} \leq b_{k,t}^{\text{hist}} + F_t. \tag{28}$$

Therefore, all Beta parameters remain in compact positive intervals. The complete variational-parameter sequence consequently lies in the compact set

$$\mathcal{X}_t = \left(\prod_{i=1}^{N} \Delta_K \right) \times \prod_{k=1}^{K} \left[a_{k,t}^{\text{hist}}, a_{k,t}^{\text{hist}} + S_t \right] \times \prod_{k=1}^{K} \left[b_{k,t}^{\text{hist}}, b_{k,t}^{\text{hist}} + F_t \right].$$

In particular, the parameter sequence has at least one accumulation point.

Convergence of the ELBO values. Let

$$p_t^{\text{post}} = p_t(\mathbf{z}_t, \boldsymbol{\theta}_t \mid \mathbf{s}_t, \mathbf{n}_t, W, A) \tag{29}$$

denote the exact posterior. The ELBO satisfies

$$\mathcal{L}_t(q_t) = \log p_t(\mathbf{s}_t \mid \mathbf{n}_t, W, A) - \mathrm{KL}\left[q_t \,\middle\|\, p_t^{\text{post}} \right].$$

Because the model has a finite number of latent assignments, proper Beta priors, and finite rollout counts, the log evidence is finite. Since the KL divergence is nonnegative,

$$\mathcal{L}_t(q_t) \leq \log p_t(\mathbf{s}_t \mid \mathbf{n}_t, W, A). \tag{30}$$

Combining this upper bound with Eq. (23) shows that the ELBO-value sequence is monotone and bounded above. Hence there exists a finite $\mathcal{L}_t^\star$ such that

$$\lim_{m \to \infty} \mathcal{L}_t(q_t^{(m)}) = \mathcal{L}_t^\star. \tag{31}$$

Stationarity of accumulation points. Consider the iterates at the boundaries of complete sequential sweeps, and let T_t denote the mapping corresponding to one full sweep of all categorical and Beta-factor updates. Under Eqs. (20)–(22), the softmax assignment update, the Beta update, and the associated digamma expectations are continuous on $\mathcal{X}_t$. Therefore, T_t is continuous.

Let $\bar{q}_t$ be an accumulation point of the sweep-level variational sequence. Suppose, for contradiction, that $T_t(\bar{q}_t) \neq \bar{q}_t$. Because every coordinate update is the unique maximizer of its coordinate subproblem, at least one update in the sweep would produce a strict ELBO increase. Hence

$$\mathcal{L}_t(T_t(\bar{q}_t)) > \mathcal{L}_t(\bar{q}_t). \tag{32}$$

By continuity, the same positive improvement would hold in a neighborhood of $\bar{q}_t$. This contradicts the convergence of the ELBO values in Eq. (31), according to which the ELBO improvement over a complete sweep must approach zero. Therefore,

$$T_t(\bar{q}_t) = \bar{q}_t. \tag{33}$$

Thus, every accumulation point is a fixed point of all exact coordinate updates and is consequently coordinate-wise optimal. Since the categorical updates are strictly positive under $A_{ik} > 0$, and the Beta parameters remain strictly positive, the ELBO is differentiable at such a fixed point subject to the simplex constraints. The corresponding first-order Karush–Kuhn–Tucker conditions are therefore satisfied. Every accumulation point is hence a stationary point of the ELBO.

The argument establishes convergence of the ELBO values and stationarity of every accumulation point.

A.6 EM Iteration Visualization

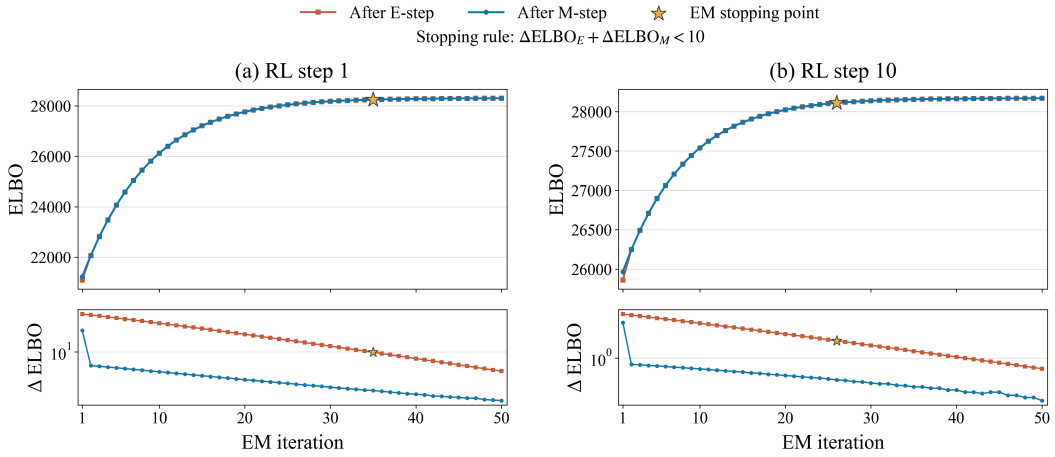


Figure 2: Examples of EM iterations

We show the evolution of the ELBO under EM iterations in Figure 2. In general, the M-step performs a notably large update only at the very first iteration. The per-round improvement in the ELBO from the E-step and M-step decays roughly exponentially, and convergence to a local stationary point or local maximum is typically reached within 50 iterations.

B Significance Analysis of Main Results

B.1 Question and Comparison Units

The main experiments report one Average@8 point estimate for each scheduler–base-model–benchmark combination. We use a two-sided exact sign test Dixon & Mood (1946) to examine whether the observed improvements are directionally consistent across these heterogeneous experimental settings. This analysis directly evaluates whether integrating our method produces positive changes more frequently than would be expected under an equal-probability null hypothesis.

The analysis includes the three full integrations: GVM + Ours, PCL + Ours, and GRESO + Ours.

For integration method m, base model b, and benchmark d, define

$$\Delta_{m,b,d} = \mathrm{Acc}_{m+\mathrm{Ours},b,d} - \mathrm{Acc}_{m,b,d}\,. \tag{34}$$

We record a gain if $\Delta_{m,b,d} > 0$, a loss if $\Delta_{m,b,d} < 0$, and a tie if $\Delta_{m,b,d} = 0$. Let W, L, and T denote the resulting numbers of gains, losses, and ties, respectively. Because ties provide no directional information, they are excluded from the sign-test sample size, yielding $n = W + L$.

C Significance Analysis of Main Results

We assess the overall directional consistency of the improvements in Table 1 using a two-sided exact sign test Dixon & Mood (1946). For each estimator variant, we compare its integrations with the corresponding original schedulers across three schedulers, two base models, and four benchmarks, yielding 24 paired comparisons.

For scheduler m, estimator variant v, base model b, and benchmark d, the accuracy change is

$$\Delta_{m,v,b,d} = \mathrm{Acc}_{m+v,b,d} - \mathrm{Acc}_{m,b,d}\,. \tag{35}$$

Positive, negative, and zero changes are counted as gains, losses, and ties, respectively. Both the cumulative and windowed variants achieve 21 gains, one loss, and two ties. Thus, each variant improves 87.5% of all evaluated settings and 95.5% of the non-tied comparisons.

Excluding ties gives $n = 22$ comparisons for each variant. Under the sign-test null model of independent, equally probable positive and negative changes, the two-sided exact p-value is

$$p_{\mathrm{sign}} = 2\sum_{r=0}^{1}\binom{22}{r}2^{-22} = 1.0967 \times 10^{-5}. \tag{36}$$

Both variants therefore yield the same pooled sign-test p-value, with gains in 21 of the 22 non-tied comparisons. This consistently positive pattern, together with the small nominal p-value, supports the overall effectiveness of both variants when integrated into existing RL schedulers.

D Hyper Parameter Analysis

D.1 Guideline of K, k_{knn} and β

We discuss three hyperparameters controlling our difficulty estimator. K is the number of latent difficulty states, determining modeling granularity; k_{nn} is the neighborhood size for graph construction, controlling feedback propagation range; β is the Potts prior strength, balancing graph smoothing against adaptation to policy changes.

Firstly, we analyze the influence of hyperparameters β and K on difficulty estimation, as shown in Table 4, where MAE is sample-level and r is the batch-level Pearson correlation. When β increases from 2 to 5 or above, all metrics become nearly identical and slightly worse, indicating that an over-strong Potts prior makes the model rely too heavily on neighbor information and neglect the global difficulty structure. In contrast, a smaller K slightly improves the overall correlation r, suggesting that fewer latent states better capture the global difficulty trend, but the sample-level MAE increases, degrading per-sample accuracy. Considering this trade-off, we choose $\beta = 2$ and $K = 320$ in the main experiments.

The neighborhood size k_{nn} determines the initial connectivity of the difficulty-aware sample graph, and thereby controls how far rollout feedback can propagate across samples. Experiments show that as long as k_{nn} is not set extremely small (e.g., below 10), the estimation accuracy is barely affected: after reciprocal-neighbor and positive-similarity filtering, the graph itself is already sufficiently sparse, so a small number of neighbors suffices for effective feedback propagation. Meanwhile, the ablation study shows that removing graph sparsification leads to only a modest performance drop, indicating that excessively increasing the neighborhood size brings no additional benefit. From a computational perspective, increasing k_{nn} raises the cost of message propagation on the graph—in

Table 4: Sensitivity analysis of β and K on Qwen-2.5-Math-1.5B. MAE is sample-level, and r is batch-level Pearson correlation.

β	K	Early Steps		Full Steps	
		r (↑)	MAE (↓)	r (↑)	MAE (↓)
2	320	0.482	0.290	0.836	0.183
2	160	0.480	0.306	0.838	0.191
2	80	0.494	0.313	0.843	0.194
2	40	0.507	0.327	0.848	0.196
1	320	0.488	0.302	0.849	0.189
5	320	0.494	0.297	0.847	0.186
10	320	0.490	0.297	0.847	0.186
20	320	0.493	0.298	0.847	0.187

each variational inference iteration, updating the latent-state distribution of each sample requires aggregating information from its neighbors, giving a theoretical complexity of $O(Nk_{\mathrm{nn}})$. However, due to random memory access on sparse graphs and edge saturation caused by positive-similarity filtering, the actual wall-clock time does not grow linearly with k_{nn}[3].

D.2 Discussion of Embedding Models and Difficulty-Aware Instruction

Models Selection

We compare several recent text-embedding models that support task-aware or instruction-conditioned embedding, including Qwen3-Embedding-0.6B, Qwen3-Embedding-4B Zhang et al. (2025), and BGE-M3 Chen et al. (2025). When their standard dense representations are used to construct the sample graph, the resulting performance is broadly similar (see Table 5), suggesting that our framework is not sensitive to the exact choice among recent embedding models. Among the direct, uncompressed configurations, Qwen3-Embedding-0.6B achieves the lowest MAE on both base models: 0.183 versus 0.188 for both Qwen3-Embedding-4B and BGE-M3 on Qwen-2.5-Math-1.5B, and 0.124 versus 0.130 and 0.139, respectively, on Llama-3.2-1B-Instruct. Its batch-level correlation is also higher than that of Qwen3-Embedding-4B on both base models, although BGE-M3 achieves a slightly higher correlation on Llama (0.783 versus 0.776). Given that Qwen3-Embedding-0.6B is substantially more efficient, we adopt it as the default encoder as a trade-off between accuracy and computational cost.

Model scale and raw embedding dimensionality do not exhibit a monotonic relationship with difficulty-estimation performance. In particular, compressing the Qwen3-Embedding-4B output to 1,024 dimensions using its Matryoshka-compatible representation Kusupati et al. (2022) improves its performance relative to using the full-dimensional representation (0.179 vs. 0.188 MAE on Qwen; 0.121 vs. 0.130 on Llama, as shown in Table 5). Notably, the MRL-compressed Qwen3-Embedding-4B at 1,024 dimensions achieves the best overall results among the evaluated embedding configurations on both base models (0.179/0.844 for Qwen and 0.121/0.787 for Llama), suggesting that a larger encoder combined with dimensionality reduction can further improve graph quality if the additional computational cost is acceptable. Compared with the default 0.6B encoder, however, the absolute MAE reductions are modest: 0.004 on Qwen and 0.003 on Llama. For the main experiments we prioritize low overhead and therefore use Qwen3-Embedding-0.6B as the default.

A plausible explanation for the benefit of MRL compression is the curse of dimensionality: in an excessively high-dimensional space, pairwise distances can become less discriminative, which may make nearest-neighbor selection noisier and weaken the local graph structure. Matryoshka Representation Learning (MRL) preserves information at nested dimensionalities, allowing the representation to be shortened without applying an arbitrary post-hoc projection. Our observation is

[3]Theoretically, the complexity grows linearly with k_{nn}, but in practice, graph sparsity leads to random memory access, and positive-similarity filtering gradually saturates the actual edge count, so the time does not increase linearly even if k_{nn} is increased. In our setting, the wall-clock time when the edge count saturates is about $1.61\times$ that of the proposed method in the main text.

consistent with this explanation, although it does not by itself establish dimensionality as the sole causal factor.

In contrast, replacing the recent instruction-aware encoders with earlier Sentence-BERT-family models Reimers & Gurevych (2019), or with the encoder configuration adopted by VIP Nguyen et al. (2026), causes a substantial performance drop. These older representations were primarily optimized for general sentence-level similarity and appear less suitable for capturing the task-specific semantic, reasoning, and difficulty cues required by our graph. Based on these results, we recommend pairing our method with a recent task-aware embedding model. Qwen3-Embedding-0.6B provides a strong default efficiency–performance trade-off, while a larger MRL-compatible encoder truncated to a moderate dimension is a viable alternative.

Discussion of Difficulty-Aware Instruction

Table 5: Full-trajectory difficulty-estimation performance of different embedding configurations. The row "-Difficulty-Aware Instruction" removes the instruction from the default encoder. MAE: sample-level mean absolute error (lower is better); r: batch-level Pearson correlation (higher is better).

Base Model	Methods	Full Steps	
		MAE (↓)	r (↑)
Qwen-2.5 Math-1.5B	Qwen3-Embedding-0.6B	0.183	0.836
	-Difficulty-Aware Instruction	0.309	0.583
	Qwen3-Embedding-4B	0.188	0.827
	Qwen3-Embedding-4B MRL(1024 dimensions)	0.179	0.844
	BGE-M3	0.188	0.823
	MiniLM	0.326	0.407
Llama-3.2 1B-Instruct	Qwen3-Embedding-0.6B	0.124	0.776
	-Difficulty-Aware Instruction	0.302	0.486
	Qwen3-Embedding-4B	0.130	0.749
	Qwen3-Embedding-4B MRL(1024 dimensions)	0.121	0.787
	BGE-M3	0.139	0.783
	MiniLM	0.304	0.407

Table 5 shows that removing the difficulty-aware instruction from Qwen3-Embedding-0.6B causes a substantial degradation in estimation quality. On Qwen-2.5-Math-1.5B, the MAE increases from 0.183 to 0.309 and the correlation drops from 0.836 to 0.583; on Llama-3.2-1B-Instruct, the MAE increases from 0.124 to 0.302 and the correlation drops from 0.776 to 0.486. This confirms that the instruction, which encourages the encoder to capture difficulty-related and reasoning-oriented features beyond pure surface semantics, contributes critically to the graph quality. Without it, the embeddings tend to connect samples that are semantically similar but not necessarily informative for difficulty transfer.

The comparison between Qwen3-Embedding-4B and its MRL-truncated 1,024-dimension variant also suggests that reducing dimensionality, when done through a representation-learning-compatible method, can improve the discriminability of nearest-neighbor structure. BGE-M3 achieves comparable performance to the Qwen3 variants, while MiniLM performs poorly, which is consistent with the above observation that older generic encoders lack the task-awareness needed for constructing a difficulty-aware graph.

D.3 Discussion of the Interpolation Weight γ

For samples with prior rollouts, we combine the graph-based prediction and the sample-level empirical success rate as$\widehat{p}_{i,t} = \gamma \widehat{p}_{i,t}^{\text{model}} + (1-\gamma)\bar{r}_{i,<t}$ The parameter γ controls the balance between transferable graph-based evidence and direct sample-level history. A small γ makes the estimate dominated by ($\bar{r}_{i,<t}$, which weakens graph propagation and can hurt difficulty ranking. A large γ relies more on the latent-state model and may ignore reliable direct observations, leading to larger estimation error.

We set $\gamma = 0.5$ as a general default. In our experiments, performance is relatively stable for $\gamma \in [0.5, 0.8]$. When $\gamma < 0.5$, ranking quality tends to decrease; when $\gamma > 0.8$, sample-level estimation error usually increases. This suggests that a moderate interpolation weight is preferable.

A more principled choice would adapt γ according to uncertainty. The uncertainty of latent state k can be estimated from the variance of its Beta posterior:

$$v_{k,t-1} = \frac{a_{k,t-1} b_{k,t-1}}{(a_{k,t-1} + b_{k,t-1})^2 (a_{k,t-1} + b_{k,t-1} + 1)}.$$

Combined with assignment uncertainty, this gives

$$\widehat{u}_{i,t} = \sum_{k=1}^{K} q_{ik,t-1} \left[v_{k,t-1} + \left(\mu_{k,t-1} - \widehat{p}_{i,t}^{\,\mathrm{model}} \right)^2 \right].$$

In principle, one could increase γ when the model-based prediction is confident, and decrease it when the sample has sufficient direct observations. However, such a rule also depends on rollout counts, sampling distributions, and the downstream scheduler. These factors vary across integrated frameworks and are hard to fix with limited experiments. We therefore use a fixed $\gamma = 0.5$.

E Difficulty and Semantic Structure of the Clusters

We examine the clustering structure from two complementary perspectives. First, we test whether samples within the same cluster have more similar historical difficulty throughout GRPO training. We then use subject and difficulty annotations from MATH to characterize the information captured by the initial spectral clusters.

E.1 Within-Cluster Difficulty Concentration Throughout Training

Setup and statistic. Following Section 4.3, we construct snapshots before training and at the end of each of the three epochs for both base models. Let $C_k(t)$ denote the set of samples with available historical observations in cluster k at checkpoint t, with $n_k(t) = |C_k(t)|$. Let $\widehat{d}_i(t)$ denote the historical difficulty estimate of sample i. The cluster mean and within-cluster variance are

$$\overline{d}_k(t) = \frac{1}{n_k(t)} \sum_{i \in C_k(t)} \widehat{d}_i(t), \tag{37}$$

$$V_k(t) = \frac{1}{n_k(t)} \sum_{i \in C_k(t)} \left(\widehat{d}_i(t) - \overline{d}_k(t) \right)^2. \tag{38}$$

We aggregate over nonempty clusters using

$$W(t) = \frac{1}{N_t} \sum_k n_k(t) V_k(t), \qquad N_t = \sum_k n_k(t). \tag{39}$$

This statistic weights samples equally and measures their average squared deviation from the corresponding cluster mean. A smaller value indicates more concentrated historical difficulty within clusters.

Stratified permutation baseline. We permute complete per-sample history vectors among samples from the same source slice and with the same epoch-presence pattern, while keeping cluster assignments fixed. This preserves cluster sizes and retains the observation counts and cross-epoch dependencies encoded in each history vector. The RL training data do not contain subject labels, so the permutations are stratified by source slice and epoch-presence pattern rather than subject category.

We perform $B = 9{,}999$ permutations. Let $W^{(b)}(t)$ denote the statistic obtained from permutation b. The mean permutation baseline is

$$\overline{W}_{\mathrm{perm}}(t) = \frac{1}{B} \sum_{b=1}^{B} W^{(b)}(t). \tag{40}$$

Table 6: Within-cluster variance of historical difficulty at different training checkpoints. The random baseline is the mean over 9,999 stratified permutations. Values are rounded to four decimal places.

Model	Checkpoint	Observed variance	Random baseline
Qwen	Initial	0.0798	0.1095
	End of epoch 1	0.0836	0.1404
	End of epoch 2	0.0762	0.1463
	End of epoch 3	0.0797	0.1517
Llama	Initial	0.0236	0.0274
	End of epoch 1	0.0468	0.0812
	End of epoch 2	0.0469	0.0831
	End of epoch 3	0.0496	0.0870

To test whether the observed within-cluster variance is lower than expected under the permutation baseline, we compute the one-sided Monte Carlo p-value

$$p(t) = \frac{1 + \sum_{b=1}^{B} \mathbb{I}\big[W^{(b)}(t) \le W(t)\big]}{B + 1}. \tag{41}$$

We define the relative variance reduction as

$$R(t) = 1 - \frac{W(t)}{\overline{W}_{\mathrm{perm}}(t)}, \tag{42}$$

where $R(t) > 0$ indicates greater within-cluster concentration than the random baseline.

Results. Table 6 shows that the observed within-cluster variance is lower than the random baseline at every checkpoint for both models. All eight primary tests have raw permutation p-values of 0.0001, yielding Holm-adjusted p-values of 0.0008 when corrected across the eight tests.

We additionally repeat the analysis after removing training-step means to account for changes in the overall difficulty level across training steps. Joint Holm correction over the eight primary tests and eight sensitivity tests yields adjusted p-values of 0.0016 for all sixteen tests. The concentration therefore persists after accounting for training-step mean differences.

These results show that the clusters retain information about historical sample difficulty throughout policy training. The smaller difficulty variation within clusters supports pooling feedback at the latent-state level, allowing observations from related samples to inform estimates for samples with sparse feedback.

E.2 Subject and Annotated Difficulty Structure on MATH

The RL training data lack subject-category annotations. We therefore use MATH, which provides both subject labels and human-annotated difficulty levels, to characterize the semantic and difficulty structure of the initial spectral clusters. This complements the trajectory-based analysis above with evidence from external annotations.

We analyze 7,500 problems from seven subject categories and construct 16 spectral clusters using a symmetrized nearest-neighbor connectivity graph with 100 neighbors and a fixed random seed of 42. Subject and difficulty annotations are used only for evaluation, not for cluster construction. Two geometry problems with missing difficulty annotations are excluded only from the difficulty analysis, leaving 7,498 valid observations.

Association with subject categories. We construct a contingency table between cluster assignments and subject categories and apply Pearson's chi-squared test of independence. To quantify association strength, we report Cramér's V:

$$V = \sqrt{\frac{\chi^2}{N \min(K-1, C-1)}}, \tag{43}$$

where N is the number of problems, K is the number of clusters, and C is the number of subject categories. The test yields $\chi^2 = 13{,}176.35$ with 90 degrees of freedom ($p < 10^{-300}$) and $V = 0.541$. All expected cell counts are at least five. These results indicate a substantial association between the spectral partition and subject identity.

Table 7: Within-subject Kruskal–Wallis tests of annotated difficulty differences across spectral clusters on MATH. N counts problems with valid difficulty annotations, and H is the tie-corrected test statistic. BH correction is applied jointly to the seven tests using unrounded p-values.

Subject	N	H	Raw p	BH-adjusted p
Algebra	1,744	194.37	2.945×10^{-33}	2.062×10^{-32}
Prealgebra	1,205	101.89	2.053×10^{-15}	7.186×10^{-15}
Intermediate algebra	1,295	67.72	2.105×10^{-9}	4.911×10^{-9}
Precalculus	746	40.62	5.667×10^{-5}	9.917×10^{-5}
Counting/probability	771	43.09	8.279×10^{-5}	1.088×10^{-4}
Geometry	868	39.32	9.325×10^{-5}	1.088×10^{-4}
Number theory	869	34.78	9.151×10^{-4}	9.151×10^{-4}

Annotated difficulty differences within subjects. To determine whether clusters capture difficulty information beyond subject identity, we analyze each subject separately. Within each subject, we apply a tie-corrected Kruskal–Wallis test to compare the ordinal difficulty annotations across clusters. This rank-based test does not require equal spacing between difficulty levels. We correct the seven resulting p-values using the Benjamini–Hochberg (BH) procedure.

Table 7 shows significant difficulty differences across clusters in every subject, with all BH-adjusted $p \leq 9.151 \times 10^{-4}$. Thus, the clusters capture annotated difficulty structure beyond differences in subject composition. Together with the persistent within-cluster concentration observed on GRPO trajectories, these findings support using the clustering structure to organize and share difficulty-related feedback.

F Windowed Memory under Non-stationary Policies

F.1 Windowed Forgetting Mechanism

The online inference procedure described in the main text accumulates rollout feedback across training steps by inheriting the state-level Beta posterior, i.e., $a_{k,t}^{\text{hist}} = a_{k,t-1}$ and $b_{k,t}^{\text{hist}} = b_{k,t-1}$. However, the rollout distribution is non-stationary because the policy parameters continuously change during RL optimization. Consequently, rollout observations collected many training steps earlier may no longer accurately characterize the success probability of the current policy. To reduce the influence of such outdated observations while remaining responsive to the current policy, we introduce a windowed forgetting mechanism. The windowed variant preserves the difficulty-aware sample graph W, the smoothed assignment prior A, the Potts graph prior, and the mean-field variational family introduced in the main method. It only changes how historical rollout evidence is organized: state-level success probabilities are estimated from a recent sliding window, whereas the complete rollout history of each previously observed sample is retained for sample-level prediction.

Windowed State-Level Statistics.

Before predicting training step t, we define the preceding window of length $L \geq 1$ as

$$\mathcal{T}_t^{(L)} = \{\tau : \max(1, t - L) \leq \tau < t\}, \tag{44}$$

where L denotes the window length and is distinguished from the graph adjacency matrix W. For latent state k, the responsibility-weighted numbers of successful and unsuccessful rollouts within this window are

$$S_{k,t}^{(L)} = \sum_{\tau \in \mathcal{T}_t^{(L)}} \sum_{i \in \mathcal{O}_\tau} q_{ik,\tau} s_{i,\tau}, \tag{45}$$

and

$$F_{k,t}^{(L)} = \sum_{\tau \in \mathcal{T}_t^{(L)}} \sum_{i \in \mathcal{O}_\tau} q_{ik,\tau} \left(n_{i,\tau} - s_{i,\tau}\right). \tag{46}$$

Rather than directly inheriting the complete Beta posterior from the previous step, the windowed variant reconstructs the state-level Beta parameters from a fixed prior and the rollout evidence within the current window:

$$a_{k,t}^{(L)} = a_0 + S_{k,t}^{(L)}, \qquad b_{k,t}^{(L)} = b_0 + F_{k,t}^{(L)}. \tag{47}$$

In our implementation, we set $a_0 = b_0 = 1$, corresponding to the uniform prior $\mathrm{Beta}(1,1)$. This is a standard weak prior for a Bernoulli success probability and does not favor any particular success rate before rollout evidence is observed. It also guarantees that the Beta parameters remain strictly positive. This property is particularly important under a finite window because some latent states may receive no rollout observations in the current window. When $S_{k,t}^{(L)} = F_{k,t}^{(L)} = 0$, the $\mathrm{Beta}(1,1)$ prior still yields a valid posterior and keeps the posterior mean and the digamma terms used in the variational updates well-defined and numerically stable.

Ideally, the prior mean

$$\mathbb{E}[\theta_{k,t}] = \frac{a_0}{a_0 + b_0} \tag{48}$$

should be close to the average success probability of the current policy. Such a prior would provide a more informative default estimate for latent states with few or no recent observations. However, obtaining a reliable estimate of the current policy's average ability would require evaluating a sufficiently large and representative subset of the training set at each stage, which would introduce substantial additional rollout cost. Estimating it only from the samples selected by the scheduler would also be potentially biased, because the selected samples are generally not representative of the full training distribution. We therefore adopt $\mathrm{Beta}(1,1)$ as a practical trade-off among prior neutrality, numerical stability, and computational efficiency.

The posterior mean success probability of latent state k is then

$$\mu_{k,t}^{(L)} = \frac{a_{k,t}^{(L)}}{a_{k,t}^{(L)} + b_{k,t}^{(L)}}. \tag{49}$$

The model-based prediction for sample x_i retains the same mixture form as in the main method:

$$\widehat{p}_{i,t}^{\,\mathrm{model}} = \sum_{k=1}^{K} q_{ik,t} \mu_{k,t}^{(L)}. \tag{50}$$

When the window covers all preceding training steps, the windowed formulation reduces to the cumulative-history variant. With a finite L, rollout observations older than L training steps no longer directly contribute to the current state-level success probabilities.

WINDOWED VARIATIONAL UPDATES.

The windowed variant retains the mean-field factorization

$$q_t(\mathbf{z}_t, \boldsymbol{\theta}_t) \approx \prod_{i=1}^{N} q_i(z_{i,t}) \prod_{k=1}^{K} q(\theta_{k,t}), \tag{51}$$

where $q_{ik,t} \equiv q_i(z_{i,t} = k)$ and

$$q(\theta_{k,t}) = \mathrm{Beta}\left(a_{k,t}^{(L)}, b_{k,t}^{(L)}\right). \tag{52}$$

The coordinate update for the latent-state assignments preserves the same three sources of information as the main method: $q_{ik}^{(r+1)} = \mathrm{Softmax}_k\left[\log A_{ik} + \beta \sum_{j=1}^{N} W_{ij} q_{jk}^{(r)} + \mathbb{I}\left(i \in \mathcal{O}_t^{(L)}\right) \ell_{ik,t}^{(L)}\right]$, where

$$\mathcal{O}_t^{(L)} = \bigcup_{\tau \in \mathcal{T}_t^{(L)}} \mathcal{O}_\tau. \tag{53}$$

For each sample i, we aggregate its rollout outcomes within the current window as

$$s_{i,t}^{(L)} = \sum_{\tau \in \mathcal{T}_t^{(L)}} s_{i,\tau}, \qquad n_{i,t}^{(L)} = \sum_{\tau \in \mathcal{T}_t^{(L)}} n_{i,\tau}, \tag{54}$$

and compute the corresponding expected log-likelihood as $\ell_{ik,t}^{(L)} = s_{i,t}^{(L)} \left[\psi\left(a_{k,t}^{(L)}\right) - \psi\left(a_{k,t}^{(L)} + b_{k,t}^{(L)}\right)\right] + \left(n_{i,t}^{(L)} - s_{i,t}^{(L)}\right) \left[\psi\left(b_{k,t}^{(L)}\right) - \psi\left(a_{k,t}^{(L)} + b_{k,t}^{(L)}\right)\right]$.

After updating the assignment probabilities, the state-level Beta factors are recomputed using only the rollout observations within the current window:

$$a_{k,t}^{(L)} = 1 + \sum_{\tau \in \mathcal{T}_t^{(L)}} \sum_{i \in \mathcal{O}_\tau} q_{ik} s_{i,\tau}, \tag{55}$$

and

$$b_{k,t}^{(L)} = 1 + \sum_{\tau \in \mathcal{T}_t^{(L)}} \sum_{i \in \mathcal{O}_\tau} q_{ik} \left(n_{i,\tau} - s_{i,\tau}\right). \tag{56}$$

We alternate the assignment and Beta-factor updates using the same stopping criterion as in the main method. Therefore, the windowed variant does not change the coordinate-ascent structure of the inference procedure; it only restricts the temporal support of the rollout evidence used in the likelihood and state-level Beta updates.

PERSISTENT ASSIGNMENT WARM START.

In the cumulative formulation, the complete estimator state $\mathcal{S}_t = \{q_t, a_t, b_t\}$ is passed between consecutive training steps. In the windowed variant, we retain only the previous assignment probabilities as the default initialization:

$$q_{ik,t}^{(0)} = q_{ik,t-1}. \tag{57}$$

The Beta parameters are not directly inherited from the previous step. Instead, they are reconstructed from the fixed $\mathrm{Beta}(1,1)$ prior and the rollout observations within the current window. The persistent initialization of q preserves continuity in the graph-structured latent assignments and provides a warm start for the finite-iteration variational procedure. Importantly, the inherited responsibilities are used only as an optimization initialization. They are not treated as additional successes or failures and therefore do not alter the strict temporal window applied to the state-level sufficient statistics. We additionally consider a reset variant that initializes q from the static soft assignment prior A at every step; unless otherwise specified, the persistent warm-start variant is used by default.

PERSISTENT SAMPLE-LEVEL HISTORY.

The sliding window is applied only to the shared state-level statistics. For each sample x_i, we separately retain its complete rollout history before step t:

$$S_{i,t}^{\mathrm{all}} = \sum_{\tau < t} s_{i,\tau}, \qquad C_{i,t}^{\mathrm{all}} = \sum_{\tau < t} n_{i,\tau}. \tag{58}$$

For a previously observed sample, its historical empirical success rate is

$$\overline{r}_{i,<t} = \frac{S_{i,t}^{\mathrm{all}}}{C_{i,t}^{\mathrm{all}}}. \tag{59}$$

We then retain the prediction rule from the main method:

$$\widehat{p}_{i,t} = \begin{cases} \widehat{p}_{i,t}^{\mathrm{model}}, & C_{i,t}^{\mathrm{all}} = 0, \\ \gamma \widehat{p}_{i,t}^{\mathrm{model}} + (1-\gamma)\overline{r}_{i,<t}, & C_{i,t}^{\mathrm{all}} > 0. \end{cases} \tag{60}$$

Consequently, the two components of the final prediction operate at complementary time scales. The model-based term uses recent rollout feedback to track the evolving success probabilities of the shared latent states, whereas the sample-level empirical term preserves all available direct evidence for each previously observed sample. The windowed mechanism therefore removes stale evidence from transferable state-level statistics without discarding the long-term historical experience associated with individual samples.

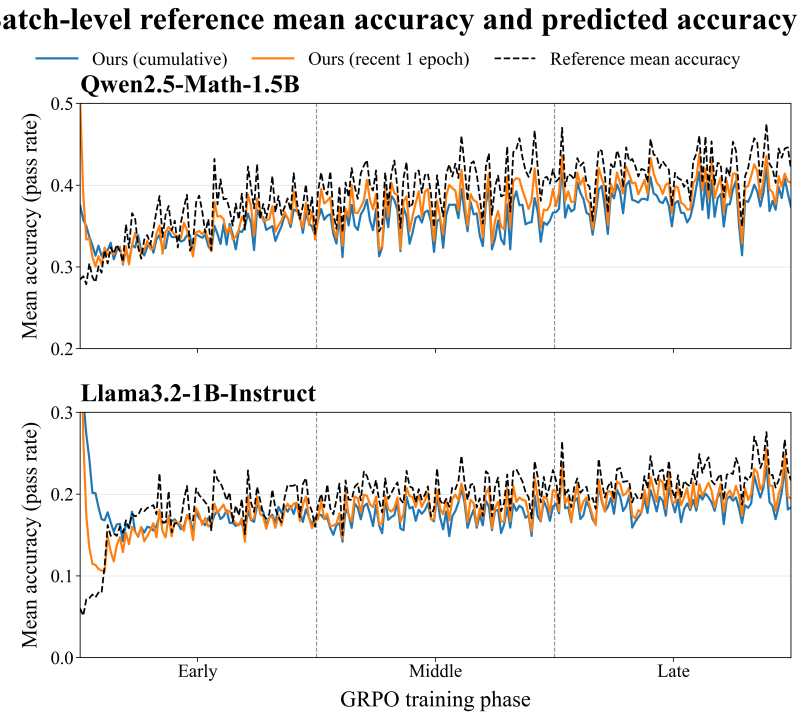


Figure 3: Batch-level reference accuracy and predicted accuracy of the cumulative and windowed estimators on Qwen-2.5-Math-1.5B and Llama-3.2-1B-Instruct.

F.2 Result and Discussion

We evaluate the windowed variant by replaying the full training logs of Qwen-2.5-Math-1.5B and Llama-3.2-1B-Instruct, and compare it with the cumulative estimator used in our main experiments. As shown in Table 2, the windowed variant achieves lower sample-level MAE on both models: from 0.183 to 0.172 on Qwen-2.5-Math-1.5B, and from 0.124 to 0.123 on Llama-3.2-1B-Instruct. This indicates that windowed forgetting makes the predicted success probability of each individual sample numerically closer to its reference rollout outcome. Fig. 3 further visualizes the batch-level reference accuracy and predicted accuracy over the full training process.

However, Table 2 also shows that the correlation metric r decreases after introducing the windowed mechanism: from 0.836 to 0.712 on Qwen-2.5-Math-1.5B, and from 0.776 to 0.729 on Llama-3.2-1B-Instruct. This suggests a trade-off: although the windowed estimator improves point-wise numerical accuracy, it weakens the ability to preserve the relative difficulty structure among samples. A likely reason is that the finite window reduces the amount of historical evidence available for state-level estimation, making the estimates more sensitive to sparse recent observations and sampling bias. For our difficulty-aware scheduling setting, stable relative ordering is more important than point-wise calibration alone. Therefore, we retain the cumulative estimator as the default method in the main experiments. Whether to use windowed forgetting should depend on the downstream objective: it is beneficial when current-policy calibration is prioritized, but less suitable when stable sample ranking is critical.

G Related Work on Graph-Based Learning and Probabilistic Inference

Graph-based learning uses sample relationships to regularize predictions and share sparse supervision. Spectral clustering extracts global partition structure from a similarity graph (Ng et al., 2001; Von Luxburg, 2007), whereas Gaussian-field propagation, local-and-global consistency, and man-

ifold regularization encourage predictions to vary smoothly over graph neighborhoods (Zhu et al., 2003; Zhou et al., 2003; Belkin et al., 2006). These approaches motivate relational evidence sharing, but RLVR difficulty estimation must additionally accommodate success probabilities that change with the policy and feedback restricted to scheduler-selected samples. Our formulation gives the graph two complementary roles: spectral clustering provides an initial assignment prior, while local graph interactions regularize subsequent latent-state inference.

Probabilistic graphical models represent relational dependence through coupled latent variables. Potts and Markov random-field priors encourage compatible neighboring states (Potts, 1952; Besag, 1974; Geman & Geman, 1984), while mean-field variational inference provides tractable coordinate updates (Zhang, 1992; McGrory et al., 2009; Wainwright & Jordan, 2008; Blei et al., 2017); stochastic and streaming methods extend inference to sequential observations (Hoffman et al., 2013; Broderick et al., 2013). Building on these foundations, we combine graph-regularized assignments with state-level Beta–Binomial models to pool sparse rollout feedback and update difficulty estimates online.

H Details of Datasets and Training

H.1 Datasets

NuminaMath.

NuminaMath is a large-scale mathematical reasoning dataset comprising approximately 860K problem–solution pairs. It covers problems ranging from Chinese high-school mathematics to U.S. and international mathematical olympiad competitions, with data primarily collected from online examination papers and mathematics discussion forums. Each problem is accompanied by a chain-of-thought solution and a verifiable final answer. Following the data-processing protocols of GVM and CurES, we extract approximately 150K problems from the original corpus for RL training Cui et al. (2025).

MATH.

MATH is a competition-level mathematical reasoning dataset containing 12.5K problems, each accompanied by a complete step-by-step solution and a final answer. The dataset provides predefined difficulty annotations ranging from Level 1 to Level 5 and category labels covering seven subjects: prealgebra, algebra, number theory, counting and probability, geometry, intermediate algebra, and precalculus. We use these annotations only as external references rather than as training supervision. Specifically, we compare the complete method with variants that remove the clustering or graph-based components and examine how well the learned representations and structural relationships align with the provided category and difficulty labels. This analysis assesses whether the proposed components capture semantic similarity among problems and distinguish problems at different difficulty levels.

MATH-500. MATH-500 is a representative subset of 500 problems selected from the original MATH benchmark. It retains the diverse mathematical subjects and difficulty levels of MATH while enabling more efficient and standardized evaluation. Each problem is accompanied by a reference solution and a verifiable final answer. We use MATH-500 to evaluate the models' general mathematical reasoning ability across a broad range of competition-level problems.

AIME 2024.

Our AIME 2024 evaluation set combines the 15 problems from AIME I and the 15 problems from AIME II, yielding 30 problems in total. These problems cover areas such as algebra, geometry, number theory, counting, and probability and generally require multi-step reasoning and nontrivial mathematical insight. Each problem has a unique integer answer between 000 and 999, allowing reliable rule-based evaluation. We use AIME 2024 to assess reasoning performance on challenging competition-level mathematics problems. The general AIME format consists of 15 questions per examination, each requiring an integer answer from 000 to 999.

AIME 2025.

Our AIME 2025 evaluation set similarly combines AIME I and AIME II, resulting in 30 problems. Each problem requires an integer answer between 000 and 999 and involves advanced high-school mathematics across multiple domains. Since these problems were released more recently than most commonly used mathematical reasoning benchmarks, AIME 2025 provides a challenging evaluation of the models' reasoning and generalization capabilities.

OLYMPIADBENCH.

OlympiadBench is a bilingual and multimodal scientific reasoning benchmark containing 8,476 problems collected from international olympiads, Chinese olympiads, and the Chinese college entrance examination. The final ACL 2024 version contains 6,142 mathematics problems and 2,334 physics problems, with every problem accompanied by an expert-annotated solution. The benchmark includes both open-ended problems and theorem-proving problems and covers English and Chinese questions with textual or visual information. Following the evaluation protocol of GVM and CurES, we use the corresponding mathematics portion of OlympiadBench to evaluate reasoning performance on challenging olympiad-level problems.

H.2 RATIONALE FOR INITIAL POLICY SELECTION

We use Qwen2.5-Math-1.5B and Llama-3.2-1B-Instruct as the initial policies for RLVR training, considering both potential data contamination and comparability with prior studies.

First, the relationship between model release dates and benchmark publication dates informs our selection (Wu et al., 2026). For benchmarks released after these model checkpoints, such as AIME 2025, this temporal separation reduces the possibility that the original checkpoints were directly exposed to the evaluation problems and provides a clearer temporal boundary for evaluation.

Second, we focus on the incremental contribution of difficulty-aware scheduling to subsequent RLVR training. For checkpoints from model families such as Qwen3.5 that have undergone dedicated reasoning-oriented post-training, their initial success-probability distributions and exploration behavior may already reflect the effects of prior reasoning-training data, reward mechanisms, and budget-allocation strategies. To maintain comparability with existing studies on curriculum learning and rollout budget allocation Yao et al. (2026); Gao et al. (2026); Zheng et al. (2026), we adopt the initial policy checkpoints used in the corresponding baseline settings. Starting from the same checkpoints allows us to analyze how the proposed difficulty estimator affects sample selection, rollout budget allocation, and subsequent training gains.