WHAT MAKES A BATCH RELIABLE? BALANCING MULTI-DOMAIN RLVR WITH TAYLORMIX

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-domain reinforcement learning with verifiable rewards (RLVR) can produce unstable policy updates even with self-generated responses. Our experiments identify batch composition as a key factor in update stability. Selecting high-scoring cases independently can overlook the risks of using them together, while concentrating training in favored domains can limit learning opportunities for other capabilities. These limitations motivate TAYLORMIX, a Taylor-motivated selector centered on **balance**. Before rollout, it balances domain priorities with continued opportunities across capabilities through KL-regularized allocation, and individual usefulness against composition risk through companion-dependent case selection. Our theoretical analysis characterizes this balance by quantifying the continuous-allocation trade-off and establishing conditional geometric-penalty reduction at a fixed selection prefix. Under matched training budgets, TAYLORMIX achieves the highest category means among the compared methods in mathematical reasoning, code, and general capability on two evaluated backbones. Ablations support both levels of balance, linking capability gains to preserved domain opportunities and geometric selection; the latter also yields fewer large gradient spikes and zero-advantage rollout groups in the observed trajectories. Our anonymous code is available at https://anonymous.4open.science/r/TaylorMix-D280.

1 INTRODUCTION

Multi-domain Reinforcement learning with verifiable rewards (RLVR) provides an effective approach to improving the various capabilities of large language models through automatically verifiable feedback (Lambert et al., 2025; DeepSeek-AI, 2025; Yu et al., 2025a). However, reports on several strong open-source models, including INTELLECT-2 (Prime Intellect Team et al., 2025) and MiniMax-M1 (MiniMax, 2025), and Qwen3-30B (Zheng et al., 2025a) document training instabilities from gradient-norm spikes to model collapse. Such instabilities can compromise training progress and waste substantial rollout and optimization resources. Notably, even in GRPO-based training (Shao et al., 2024), self-generated responses with probability-ratio clipping do not ensure stable updates. Our experiments identify batch composition—the cases selected together to generate these responses—as a key factor in this instability: what matters is not only the individual responses, but also which responses contribute to the same update. This finding raises a fundamental question: which training cases should generate the responses that contribute to the same policy update?

A simple approach is to rank training examples by their individual scores and select the highest-scoring ones (Gao et al., 2025; Gu et al., 2026; Koh & Liang, 2017; Pruthi et al., 2020; Park et al., 2023). Yet selecting examples by their individual scores alone can overlook the risks of using them together: even high-scoring examples may form a high-risk batch, as illustrated schematically by the large joint penalty in Figure 1. Beyond individual example selection, domain-level allocation offers another way to organize training data: first assign sampling quotas to domains based on their expected training value (Xie et al., 2023; Ye et al., 2024; Albalak et al., 2023), then select high-scoring examples within each domain. However, allocating domain shares solely by estimated value can favor boundary solutions, leaving some domains with no training opportunities, while case-score-based selection within each quota still leaves the risks of combining examples unaddressed. Together, these limitations motivate a strategy that preserves learning opportunities across domains while accounting for which examples are used together.

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.

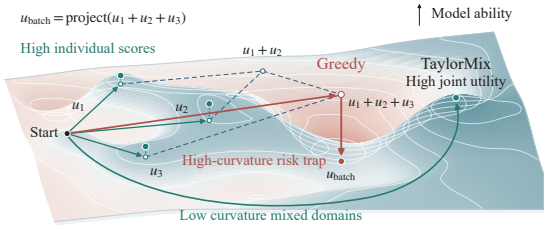

Figure 1: The schematic illustration of batch-composition risk. Individually promising directions combine at the hollow endpoint; the red descent into a high-curvature risk trap illustrates the composition risk of greedy selection.

To address these limitations, our approach centers on **balance**. Specifically, a Taylor expansion of the update improvement motivates this set-level view: first-order gains can be offset by a second-order correction that captures interaction risk among the combined update directions. Building on this view, we introduce TAYLORMIX, which coordinates domain allocation and case selection before rollout. At the domain level, TAYLORMIX **balances** prioritizing promising domains, guided by reward feedback and target alignment, with preserving learning opportunities across capabilities through KL-regularized allocation. Within each domain's quota, it **balances** individual usefulness against a geometric composition penalty, revising each candidate's priority as other cases are selected. Each candidate is thus evaluated for what it adds to the emerging batch, making the risks of combining individually useful cases an explicit part of selection. Together, these mechanisms determine where to allocate learning opportunities and how to combine them into effective policy updates. This design makes this **balance** explicit in each selection: a candidate's value depends on both its individual usefulness and its interaction with the examples already chosen. A lower-scoring example can therefore become a more valuable addition when the reduction in composition risk outweighs the difference in individual usefulness.

Under matched training budgets, TAYLORMIX achieves the highest category means in mathematical reasoning, code, and general capability on both evaluated backbones. The results also illustrate how this balance operates in practice: KL-regularized allocation keeps every domain represented throughout the recorded allocation sequence, whereas removing the regularizer consistently leaves a domain out. Within these allocations, balancing individual usefulness against composition risk improves benchmark performance and yields fewer large gradient spikes and zero-advantage rollout groups in the observed trajectories.

Contributions. Our contributions are threefold. First, we formulate multi-domain RLVR batch construction around joint update value, linking domain opportunities and case combinations to learning across capabilities and making companion-dependent value an explicit criterion for data selection. Second, we introduce TAYLORMIX, a Taylor-motivated selector balancing domain priority with coverage and individual usefulness with a geometric composition penalty. We quantify the cost of preserving coverage and establish fixed-prefix geometric-penalty reduction, making these trade-offs explicit. Third, experiments show that TAYLORMIX sustains learning opportunities and enables effective training across all evaluated domains, improving mathematical reasoning, code, and general capability on two backbones under matched training budgets. Controlled analyses connect these gains to domain coverage and geometric selection; the latter also yields fewer large gradient spikes and degenerate rollout groups in observed trajectories.

2 RELATED WORK

Data selection and domain mixing. Influence Functions (Koh & Liang, 2017) and TracIn (Pruthi et al., 2020) assess training-example influence using inverse-Hessian information and checkpoint gradient alignments, respectively. GLISTER (Killamsetty et al., 2021b) connects subset selection to validation improvement, while GRAD-MATCH (Killamsetty et al., 2021a) constructs weighted subsets that approximate training or validation gradients. TRAK (Park et al., 2023) scales attribution through projected gradient features. At the domain level, DoReMi (Xie et al., 2023) learns mixture weights with a proxy model, while Skill-it! (Chen et al., 2023) and Efficient Online Data Mixing (Albalak et al., 2023) adapt sampling through skill dependencies and bandit feedback, respectively. Data Mixing Laws (Ye et al., 2024) predicts language-modeling loss from mixture proportions, and AutoScale (Kang et al., 2024) extrapolates allocations optimized at smaller scales to larger training budgets. Dynamic Gradient Alignment (Fan et al., 2024) uses target-gradient align-

ment to reweight domains, and R&B (Ge et al., 2025) combines semantic regrouping with mixture optimization based on domain-gradient interactions. Our domain-level baselines cover two allocation strategies: Domain2Vec (Zhang et al., 2025) derives training-free mixtures from dataset-vector alignment, while AC-ODM (Ma et al., 2025) learns an actor–critic policy for online domain mixing.

Online curricula for RL post-training. RL post-training places data selection inside an evolving policy loop. DeepSeekMath (Shao et al., 2024) introduces GRPO, while DAPO (Yu et al., 2025b) uses dynamic sampling to filter out prompt groups with no reward variation after rollout. At the distribution level, DUMP (Wang et al., 2025) uses UCB to schedule heterogeneous distributions, and Self-Evolving Curriculum (Chen et al., 2025) learns a category-sampling policy through a non-stationary bandit formulation. At the prompt level, Prompt Curriculum Learning (Gao et al., 2025) uses a learned value model to select prompts of intermediate difficulty for the current policy, while Actor-Curator (Gu et al., 2026) learns a neural curator targeting expected policy improvement under partial feedback. GRESO (Zheng et al., 2025b) uses historical reward dynamics to skip prompts predicted to be uninformative before rollout, whereas CoDaPO (Zhou et al., 2026) uses rollout confidence and empirical difficulty to reweight policy updates and resample learnable questions. Complementing these algorithmic developments, Rethinking Easy-to-Hard (Mordig et al., 2026) reports no robust advantage for difficulty-based ordering over random sampling on synthetic deductive-reasoning benchmarks. These works provide complementary views of training-data value, from individual examples to domain mixtures and changing policy states. We build on these perspectives around a central observation: individually promising cases may not form a useful batch.

3 Preliminaries and Problem Definition

Setup. Let $\mathcal{D} = \bigsqcup_{c=1}^{K} \mathcal{D}_c$ contain K capability domains with verifiers v_c, which may be shared. At step t, policy π_{θ_t} generates $y_i \sim \pi_{\theta_t}(\cdot \mid x_i)$ for each selected prompt $x_i \in \mathcal{B}_t$. Its domain verifier returns $r_i = v_c(x_i, y_i)$, and GRPO applies $\theta_{t+1} = \theta_t - \eta g_t(\mathcal{B}_t; \omega_t)$ (Shao et al., 2024), where ω_t denotes rollout randomness. Responses, rewards, and learning signals change with the policy.

Before rollout, the selector observes $\mathcal{S}_t = (\theta_t, \mathcal{H}_t, \{\mathcal{C}_{c,t}\}_{c=1}^{K}, B)$, where $\mathcal{C}_{c,t} \subseteq \mathcal{D}_c$ contains eligible candidates, B is the batch size, and $\mathcal{H}_t$ contains only feedback and cached representations from completed updates. The current batch's responses, rewards are not yet available.

How should learning opportunities be allocated? Allocation determines how many cases each domain contributes, balancing current usefulness with continued opportunities across capabilities. The nonnegative integer quotas satisfy $\mathbf{n}_t = (n_{1,t}, \ldots, n_{K,t}), \qquad \sum_{c=1}^{K} n_{c,t} = B.$

Which cases should be learned from together? Quotas specify counts, not case identities. We must choose $\mathcal{B}_{c,t} \subseteq \mathcal{C}_{c,t}, |\mathcal{B}_{c,t}| = n_{c,t}, \mathcal{B}_t = \bigsqcup_{c=1}^{K} \mathcal{B}_{c,t}$. Because the selected cases contribute to one update, their value depends on their companions, not just their individual promise. Thus, batch construction requires allocating opportunities across domains and combining cases within them, using only pre-rollout information. Section 4 presents the resulting conditional-selection rules.

4 Method

Individually promising cases need not form a useful training batch. TaylorMix follows a **balance** principle: across domains, it favors currently promising domains while preserving learning opportunities for other capabilities; within each domain, it weighs individual usefulness against a geometric composition penalty. Before rollout at step t, we allocate B slots across K domains, then select cases within each quota based on those already chosen there. The policy generates responses for the selected cases and is updated with GRPO; feedback informs the next step.

4.1 Balancing Domain Opportunities

To support progress across capabilities, we balance priority for currently promising domains with continued learning opportunities for others. Target alignment and reward feedback from completed updates define each domain's cost Cost_c, with lower values favoring allocation. A positive reference mixture q specifies baseline shares, and KL regularization moderates how far current costs shift allocation from this reference. Specifically, we balance a domain-cost surrogate motivated by first-

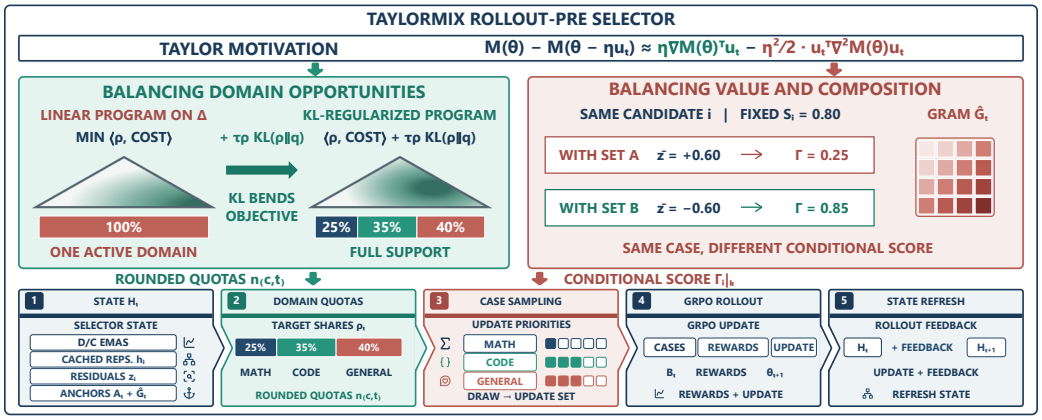

Figure 2: TAYLORMIX balances current domain preference with retained opportunities, then individual value with composition within each quota. The schematic holds a candidate's score fixed while its companions change its conditional score. GRPO feedback informs the next step.

order Taylor alignment with regularization toward the reference mixture:

$$\rho_t^\star = \arg\min_{\rho \in \Delta^{K-1}} \underbrace{\sum_c \rho_c \, \mathrm{Cost}_c}_{\text{first-order Taylor-motivated surrogate}} + \tau_\rho \, \mathrm{KL}(\rho \| q). \tag{1}$$

For finite costs and $\tau_\rho > 0$, the unique solution is

$$\rho_{c,t}^\star = \frac{q_c \exp(-\mathrm{Cost}_c / \tau_\rho)}{\sum_{d=1}^{K} q_d \exp(-\mathrm{Cost}_d / \tau_\rho)}. \tag{2}$$

The exponential factor favors lower-cost domains, while τ_ρ controls the strength of this preference: larger values keep shares closer to q, and smaller values concentrate allocation more strongly on lower-cost domains. Every continuous share remains positive. Largest-remainder rounding converts $B\rho_{c,t}^\star$ into domain c's quota $n_{c,t}$; slots left unfilled by insufficient eligible cases are reassigned to domains with spare capacity. Appendix A.1 gives the cost signals and normalization. These quotas allocate learning opportunities across domains; within each quota, we next balance individual usefulness with case composition.

4.2 BALANCING INDIVIDUAL VALUE AND COMPOSITION

Within each allocated domain quota, we balance individual usefulness with composition risk. Taylor expansion motivates a set utility with individual and interaction terms. For candidates in $\mathcal{C}_{c,t}$, S_i combines target relevance, reward-based learning room, and an individual geometric penalty; z_i records low-dimensional deviation from the domain prototype. Specifically, for a nonempty set $\mathcal{B} \subseteq \mathcal{C}_{c,t}$ with mean residual $\bar{z}_\mathcal{B}$, we define the utility using the anchor Gram matrix $\widehat{G}_t \succeq 0$:

$$\mathcal{V}(\mathcal{B}) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} S_i - \underbrace{\frac{1}{2} \bar{z}_\mathcal{B}^\top \widehat{G}_t \bar{z}_\mathcal{B}}_{\mathcal{R}(\mathcal{B}) \text{ : second-order Taylor-motivated penalty}}. \tag{3}$$

This balances average individual score against a composition penalty whose cross terms depend on the cases selected together.

Let $\mathcal{B}_{c,t}^{(k)}$ contain the k selected cases, with mean $\bar{z}_{c,t}^{(k)}$. For this fixed set, expanding the change in $\mathcal{V}$ gives a conditional score that ranks the exact next-addition utility increments:

$$\Gamma_{i|k} = S_i - \frac{k}{k+1} z_i^\top \widehat{G}_t \bar{z}_{c,t}^{(k)} - \frac{z_i^\top \widehat{G}_t z_i}{2(k+1)}. \tag{4}$$

The cross term captures companion interactions; the diagonal term accounts for the candidate's own contribution to the enlarged-set penalty. To retain exploration, we draw from the remaining window

$\mathcal{C}_{c,t}^{(k)}$ with probability

$$\Pr(i_{k+1} = i \mid \mathcal{B}_{c,t}^{(k)}, \mathcal{C}_{c,t}^{(k)}) = \frac{\exp(\Gamma_{i|k}/T_s)}{\sum_{j \in \mathcal{C}_{c,t}^{(k)}} \exp(\Gamma_{j|k}/T_s)}. \tag{5}$$

Here $T_s > 0$ controls exploration. Starting from a zero mean, each draw removes the chosen ID and updates the mean until the quota is filled without replacement.

Selection occurs before rollout using cached residuals and anchor outer products. Within each quota, S_i, z_i, and $\widehat{G}_t$ stay fixed; only the selected-set mean and remaining window change, so rescoring requires no new responses. Appendix A.2 gives the construction.

4.3 Training and Next-Step Feedback

The quotas and selected cases form the training batch $\mathcal{B}_t = \cup_c \mathcal{B}_{c,t}$. The policy generates responses to these cases for the unchanged GRPO update. Observed rewards refresh domain costs and case scores for the next step, keeping both levels of balance responsive to training progress. Representations and geometric state refresh on their schedules. Selected IDs leave persistent pools; unselected IDs keep their order, and pools with insufficient remaining cases reset before selection. Algorithm 1 and Appendix A.3 give the complete execution and state interfaces.

Together, TaylorMix builds batches to support progress across capabilities by balancing current domain priorities with continued learning opportunities, and individual usefulness with within-domain geometric composition penalties. We next characterize the allocation cost of retaining domain opportunities and when companion-dependent selection reduces the expected composition penalty at a fixed selection prefix.

5 Analysis

The method applies balance at two levels: favoring promising domains without eliminating other opportunities, then weighs individual value against composition within each quota. The two results give these choices an optimization basis, comparing allocation with its reference mixture and conditional selection with an individual-score sampler.

5.1 Balancing Cost Responsiveness and Domain Opportunities

Concentrating training in currently favored domains can limit learning opportunities for other capabilities. Supporting progress across capabilities therefore calls for balancing responsiveness to current domain priorities with continued opportunities for others. Specifically, Theorem 1 characterizes this balance for the allocation in Eq. (2) by comparing its average domain cost with that of the reference mixture q and quantifying the continuous share retained by every domain.

Theorem 1 (Cost-responsive domain opportunities). For finite costs, $q_c > 0$, and $\tau_\rho > 0$, let $D = \max_c \mathrm{Cost}_c - \min_c \mathrm{Cost}_c$ be the spread of domain costs. The shares from Eq. (1) satisfy

$$\begin{aligned} \sum_c \rho_{c,t}^\star \,\mathrm{Cost}_c &\le \sum_c q_c \,\mathrm{Cost}_c, \\ \rho_{c,t}^\star &\ge \frac{q_c}{q_c + (1 - q_c)e^{D/\tau_\rho}} > 0. \end{aligned} \tag{6}$$

These bounds make the domain-level balance concrete: retaining continuous opportunities across domains is compatible with improving or preserving average cost relative to q. With costs and q fixed, increasing τ_ρ weakens cost-based reweighting and strengthens the share lower bound toward q_c; smaller positive values give current priorities greater influence. The temperature thus adjusts the strength of current preferences and the protection of opportunities for other capabilities. After rounding and supply handling, Table 2 reports nonzero quotas for all domains at each observed step.

Retaining domain opportunities is part of constructing useful training batches. Within each quota, individually promising cases can still form a high-penalty combination. We next examine how companion-dependent selection balances individual value with the geometric composition penalty.

5.2 BALANCING INDIVIDUAL VALUE AND COMPOSITION

Within each quota, TAYLORMIX seeks a balance between favoring individually promising cases and limiting the geometric penalty of combining them. We examine this balance by comparing two samplers with the selected set $\mathcal{B}$ and remaining window held fixed. The baseline $p_i^0 \propto e^{S_i/T_s}$ uses individual scores alone, while TAYLORMIX's conditional sampler p_i adjusts these preferences using the composition terms in Eq. (4). Both use the same scores S_i and temperature T_s. Specifically, Theorem 2 quantifies the effect on the expected composition penalty and characterizes the value–composition balance in the selection objective.

Theorem 2 (Composition-aware next-case selection). Fix a selected set $\mathcal{B}$ of k cases and a finite, nonempty remaining window. Assume that S_i, z_i, and $\widehat{G}_t$ are fixed and finite, with $\widehat{G}_t \succeq 0$ and $T_s > 0$.

Expected composition penalty. For the composition penalty $\mathcal{R}$ defined in Eq. (3), the expected penalty after adding one case satisfies

$$\mathbb{E}_{i\sim p}\mathcal{R}(\mathcal{B}\cup\{i\}) \le \mathbb{E}_{i\sim p^0}\mathcal{R}(\mathcal{B}\cup\{i\}). \tag{7}$$

The inequality is strict whenever $\mathcal{R}(\mathcal{B}\cup\{i\})$ differs across candidates.

Value–composition balance. The sampler balances individual value and composition through $\mathcal{V}$, which rewards a higher average individual score and penalizes a higher composition penalty. Among all probability distributions r for drawing the next case from the remaining window, p uniquely maximizes

$$\mathbb{E}_{i\sim r}\big[\mathcal{V}(\mathcal{B}\cup\{i\}) - \mathcal{V}(\mathcal{B})\big] + \frac{T_s}{k+1}\mathsf{H}(r), \tag{8}$$

where $\mathsf{H}(r) = -\sum_i r_i \log r_i$ is the entropy of r. The expected utility gain expresses the value–composition balance, while entropy encourages exploration. We use the convention $\mathcal{V}(\varnothing) = 0$.

The first result concerns the penalty after a candidate joins the set, not its penalty alone. The second makes the value–composition balance explicit: the optimized utility includes both individual scores and the enlarged-set penalty. A lower-scoring case receives higher priority when its composition improvement outweighs the difference in average individual score (Appendix B.3). Thus neither individual score nor penalty alone determines the best next addition.

Together, the results characterize balance in domain allocation and within-domain selection. Appendices B and C provide the exact identities, sensitivity bounds, and complete proofs. Subsequently, experiments will assess whether this balance supports effective learning across capabilities.

6 EXPERIMENTS

We evaluate performance across mathematical reasoning, code, and general capability, then ablate the selector components. Section 7 examines coverage and training dynamics. Appendix D supplies the detailed evaluation protocol.

6.1 EVALUATION SETUP AND BASELINES

- **Backbones.** We evaluate Qwen3.5-2B (Qwen Team, 2026) and Qwen2.5-1.5B (Yang et al., 2024).
- **Benchmarks.** Mathematical reasoning uses MATH-500 (Lightman et al., 2023) and AIME 2024/2025 (Mathematical Association of America, 2026); code capability uses LiveCodeBench (LCB) (Jain et al., 2024) for generation, execution, and test-output, together with HumanEval (Chen et al., 2021) for functional code generation; general capability uses Arena-Hard v2.0 and its hard and creative subsets (Li et al., 2024; LMArena, 2025). Dataset details are in Appendix D.1.
- **Metrics.** Qwen3.5-2B reports AIME Avg@8, whereas Qwen2.5-1.5B reports Pass@64, so comparisons remain within each backbone and metric. Appendix D.2 defines all pass rates, the code-capability mean, and order-balanced Arena scoring; values are percentages.

Baselines. We compare three groups of baselines:

Table 1: Performance after RLVR training across model backbones and task families. Results marked with * indicate statistically significant improvements over the best baseline at $p < 0.05$.

Mathematical reasoning

Method	Qwen3.5-2B MATH-500 Pass@1	Qwen3.5-2B AIME24 Avg@8	Qwen3.5-2B AIME25 Avg@8	Qwen3.5-2B Mean	Qwen2.5-1.5B MATH-500 Pass@1	Qwen2.5-1.5B AIME24 Pass@64	Qwen2.5-1.5B AIME25 Pass@64	Qwen2.5-1.5B Mean
Base	71.6	20.42	18.75	36.92	40.4	10.00	6.67	19.02
Vanilla GRPO	74.8	15.83	15.42	35.35	51.8	3.33	3.33	19.49
Domain2Vec	74.0	17.50	14.17	35.22	49.6	10.00	6.67	22.09
AC-ODM	75.2	23.75	22.08	40.34	52.0	10.00	3.33	21.78
GRESO	70.4	25.83	20.83	39.02	50.8	13.33	3.33	22.49
CoDaPO	75.6	20.00	15.42	37.01	49.6	10.00	3.33	20.98
TAYLORMIX	**77.2***	**25.83***	**22.92***	**41.98***	**58.6***	**16.67***	**13.33***	**29.53***

Code capability

Method	Qwen3.5-2B Code Gen.	Qwen3.5-2B Test Output	Qwen3.5-2B Code Exec.	Qwen3.5-2B HumanEval Pass@1	Qwen3.5-2B Mean	Qwen2.5-1.5B Code Gen.	Qwen2.5-1.5B Test Output	Qwen2.5-1.5B Code Exec.	Qwen2.5-1.5B HumanEval Pass@1	Qwen2.5-1.5B Mean
Base	12.50	13.78	17.42	49.39	23.27	0.25	3.85	17.90	48.17	17.54
Vanilla GRPO	18.00	30.61	20.23	60.98	32.46	2.75	4.71	19.62	46.95	18.51
Domain2Vec	15.40	31.94	17.72	60.98	31.51	1.75	4.94	20.46	45.73	18.22
AC-ODM	17.50	28.96	20.25	55.49	30.55	0.75	5.81	18.37	45.73	17.67
GRESO	15.25	32.56	22.11	61.59	32.88	2.50	5.81	19.21	48.17	18.92
CoDaPO	14.25	25.11	24.11	62.20	31.42	2.75	5.23	19.83	43.90	17.93
TAYLORMIX	**19.75***	**32.77***	**27.77***	**65.24**	**36.38***	**3.00***	**6.94***	**24.22***	**51.22**	**21.35***

General capability

Method	Qwen3.5-2B Arena WinScore	Qwen3.5-2B Arena-Hard	Qwen3.5-2B Arena-Creative	Qwen3.5-2B Mean	Qwen2.5-1.5B Arena WinScore	Qwen2.5-1.5B Arena-Hard	Qwen2.5-1.5B Arena-Creative	Qwen2.5-1.5B Mean
Base	20.53	19.75	27.36	22.55	6.01	5.18	6.49	5.89
Vanilla GRPO	23.53	22.79	28.66	24.99	5.90	5.32	6.41	5.88
Domain2Vec	22.26	20.17	28.95	23.79	6.56	6.55	6.57	6.56
AC-ODM	23.95	21.04	29.75	24.91	8.09	7.89	8.52	8.17
GRESO	24.76	21.33	33.78	26.62	7.97	7.98	7.94	7.96
CoDaPO	24.22	19.78	33.09	25.70	7.74	7.61	8.01	7.79
TAYLORMIX	**26.84***	**22.80***	**37.21***	**28.95***	**8.56***	**8.26***	**9.86***	**8.89***

- **Reference baselines.** **Base** is the unadapted checkpoint before RLVR training. **Vanilla GRPO** (Shao et al., 2024) uses the same GRPO update and budget without rollout-pre data selection, testing whether standard RLVR alone explains the gains.

- **Domain-level selection. Domain2Vec** (Zhang et al., 2025) derives a static training mixture from dataset representations. **AC-ODM** (Ma et al., 2025) learns online domain weights with an actor–critic controller. These methods test domain allocation without case-level geometric selection.

- **Case-level selection. GRESO** (Zheng et al., 2025b) skips prompts using online reward dynamics. **CoDaPO** (Zhou et al., 2026) uses confidence and empirical difficulty to reweight updates and resample learnable questions. These methods test case selection without explicit domain allocation.

Selector settings and public implementation. Appendix E.4 gives the corresponding sketch, temperature, window, and history settings. Appendix D.3 describes each method and its source. Except for Base, comparisons use the same base model family, training pool, reward or verifier interface, GRPO optimizer, training budget, and evaluation prompts. Within each comparison, generation and evaluation procedures are matched.

Table 2: Performance and domain coverage with and without KL regularization.

Base / sequence and KL	AIME24 Avg@8 / Pass@64	AIME25 Avg@8 / Pass@64	MATH-500 Pass@1	LCB Code generation	LCB Code execution	LCB Test output	LCB Mean	Arena-Hard	Arena-Creative
Qwen3.5-2B, Base	20.42	18.75	71.6	12.50	17.42	13.78	14.57	19.75	27.36
Qwen3.5-2B, without KL	24.58	22.08	76.2	17.75	20.43	27.85	22.01	**23.04**	33.65
Qwen3.5-2B, with KL	**25.83**	**22.92**	**77.2**	**19.75**	**27.77**	**32.77**	**26.76**	22.80	**37.21**
Qwen2.5-1.5B, Base	10.00	6.67	40.4	0.25	17.90	3.85	7.33	5.18	6.49
Qwen2.5-1.5B, without KL	10.00	**13.33**	53.8	0.25	21.50	2.26	8.00	5.20	7.33
Qwen2.5-1.5B, with KL	**16.67**	**13.33**	**58.6**	**3.00**	**24.22**	**6.94**	**11.39**	**8.26**	**9.86**

Sequence and KL	Entropy Mean ↑	Effective domains Mean ↑	Minimum share Mean ↑	Maximum share Mean ↓	All-domain Active steps ↑	Zero-domain Slots / step ↓
Data Mixing Laws, without KL	0.379	1.46	0.000	0.874	0/56	1.00
same cost sequence, with KL	**0.927**	**2.77**	**0.194**	**0.490**	**56/56**	**0.00**

Table 3: Performance after RLVR training when each TaylorMix component is removed.

Removed component	AIME24 Avg@8	AIME25 Avg@8	MATH-500 Pass@1	LCB Code generation	LCB Code execution	LCB Test output	LCB Mean	Arena-Hard	Arena-Creative
without inner candidate selection	22.92	20.83	76.20	19.25	17.95	27.36	21.52	9.97	14.61
without low-cost sketch abstraction	24.58	22.08	76.20	17.25	17.12	30.04	21.47	10.41	22.05
without curvature	15.42	20.00	71.20	11.25	20.43	22.85	18.18	16.70	28.30
Full TAYLORMIX	**25.83**	**22.92**	**77.20**	**19.75**	**27.77**	**32.77**	**26.76**	**22.80**	**37.21**

6.2 OVERALL PERFORMANCE

We evaluate whether TAYLORMIX's two-level balance translates into joint capability gains: balancing current domain priorities with continued opportunities for other capabilities, and individual usefulness with composition risk within each quota. We compare TAYLORMIX with the baselines on both backbones under the matched training and evaluation setup. Table 1 summarizes the results.

Can TAYLORMIX improve multiple capabilities together? TAYLORMIX achieves the strongest or tied-strongest results across mathematical reasoning, code, and general capability. Its advantages over both domain-level and case-level baselines extend across the capability families. These joint improvements support the intended outcome of the two-level balance: making learning opportunities effective across capabilities through their allocation and the cases selected to use them.

Can TAYLORMIX remain effective across different backbones? The joint gains hold on both evaluated backbones despite their different starting capability profiles. This consistency supports the applicability of the combined allocation–selection design across the tested starting policies. We next use ablations to examine how retaining domain opportunities and accounting for case composition contribute to these gains.

6.3 ABLATION STUDY

We conduct a component-wise ablation on one backbone using the main comparison's endpoint evaluation setup. Table 3 compares the starting checkpoint, the full selector, and variants without within-domain selection, the low-cost representation, or geometric selection.

Can TAYLORMIX make effective use of domain opportunities? The full selector develops a stronger cross-capability profile than the starting checkpoint during RLVR post-training. Removing within-domain selection or its low-cost representation weakens this profile. These results support complementing domain-level balance with effective within-domain selection: the cases filling each quota influence how well the allocated opportunities translate into capability gains. Candidate selection and its representation thus help make learning opportunities effective across capabilities.

Can TAYLORMIX turn composition awareness into capability gains? Removing geometric selection reduces post-training performance across capability families. This provides performance evidence for the value–composition balance in TAYLORMIX, which weighs individual usefulness against the geometric penalty of the cases selected together. The stronger performance of the full selector supports accounting for composition when turning individual learning opportunities into a training batch. Section 7 next examines the corresponding effects on domain coverage and training dynamics under controlled comparisons.

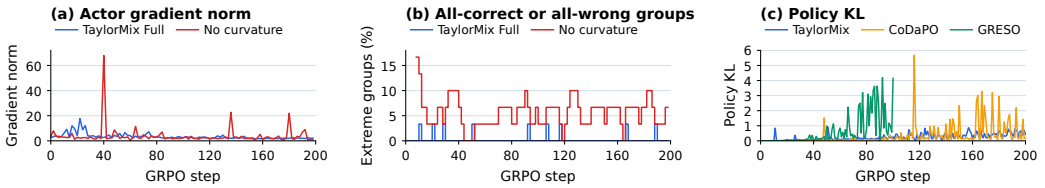

Figure 3: Training dynamics during RLVR: (a) actor gradient norm and (b) all-correct or all-wrong rollout groups for TaylorMix and its geometry-removed variant; (c) policy KL for TaylorMix, CoDaPO, and GRESO.

Table 4: Performance with and without geometric case selection.

Base and curvature	AIME24 Avg@8 / Pass@64	AIME25 Avg@8 / Pass@64	MATH-500 Pass@1	LCB Code generation	LCB Code execution	LCB Test output	LCB Mean	Arena-Hard	Arena-Creative
Qwen3.5-2B, without curvature	15.42	20.00	71.2	11.25	20.43	22.85	18.18	16.7	28.3
Qwen3.5-2B, with curvature	**25.83**	**22.92**	**77.2**	**19.75**	**27.77**	**32.77**	**26.76**	**22.80**	**37.21**
Qwen2.5-1.5B, without curvature	**13.33**	**13.33**	52.8	2.50	19.33	4.58	8.80	7.35	8.22
Qwen2.5-1.5B, with curvature	**16.67**	**13.33**	**58.6**	**3.00**	**24.22**	**6.94**	**11.39**	**8.26**	**9.86**

7 Analysis of Coverage and Training Dynamics

We test TaylorMix's two-level balance through domain coverage and the contribution of geometric selection beyond relevance and reward feedback.

Can TaylorMix balance domain priorities and coverage? We compare allocation with and without KL regularization under matched domain-cost signals and training settings (Table 2). Without KL, cost-only allocation excludes a domain from the training batches. With KL, every domain receives a nonzero integer quota at every observed allocation step. The regularized models also achieve stronger overall performance across Math, Code, and General on both backbones. These findings support TaylorMix's domain-level balance: favoring current priorities while retaining opportunities for other capabilities to improve.

Can TaylorMix balance individual value and composition risk? We hold domain allocation and the linear score fixed and compare the full selector with its geometry-removed variant under matched rollout and training settings. The full selector achieves a stronger capability profile on both backbones (Table 4), alongside fewer large gradient spikes and all-correct or all-wrong rollout groups in the observed trajectories (Figure 3). These findings support balancing individual usefulness with composition risk: retaining geometric selection yields capability gains alongside more stable and informative training in the observed runs.

With domain allocation and the linear score held fixed, retaining geometric selection improves the capability profile on both backbones. The observed trajectories also show fewer large gradient spikes and fewer all-correct or all-wrong rollout groups, which provide no group-relative learning signal. Together, these observations support balancing individual usefulness with composition risk: accounting for how cases combine yields capability gains alongside fewer disruptive updates and uninformative rollout groups in the observed runs.

8 Conclusion

We identified batch composition as a key factor in multi-domain RLVR stability: individually promising cases did not necessarily form a useful training batch. TaylorMix implemented a Taylor-motivated **balance** principle before rollout. Across domains, KL-regularized allocation balanced current priorities with continued learning opportunities; within each quota, companion-dependent selection balanced individual usefulness against a geometric composition penalty. Under matched training budgets, TaylorMix improved mathematical reasoning, code, and general capability over representative domain- and case-level baselines on two evaluated backbones. Controlled analyses linked these gains to preserved domain opportunities and geometric selection; the latter also yielded fewer large gradient spikes and zero-advantage rollout groups in the observed trajectories. Together, these findings supported two-level **balance** as a practical principle for turning allocated opportunities into effective learning across capabilities.

AI USE STATEMENT

AI tools assisted with manuscript writing and LaTeX formatting, discovery, and research execution. The authors reviewed the manuscript and remain responsible for its contents.

ETHICS STATEMENT

This work studies data selection for multi-domain language-model post-training. Responsible use should consider biases in training data and reward signals, as well as potential misuse of improved model capabilities.

REPRODUCIBILITY STATEMENT

A reference implementation of the selector is available at: `https://anonymous.4open.science/r/TaylorMix-D280`. The experimental setup is described in Section 6; Appendices C and E provide theoretical proofs and implementation details, respectively.

REFERENCES

Alon Albalak, Liangming Pan, Colin Raffel, and William Yang Wang. Efficient online data mixing for language model pre-training. *arXiv preprint arXiv:2312.02406*, 2023. URL https://arxiv.org/abs/2312.02406.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Mayee F. Chen, Nicholas Roberts, Kush Bhatia, Jue Wang, Ce Zhang, Frederic Sala, and Christopher Ré. Skill-it! a data-driven skills framework for understanding and training language models. *Advances in Neural Information Processing Systems 36*, 2023. URL https://arxiv.org/abs/2307.14430.

Xiaoyin Chen, Jiarui Lu, Minsu Kim, Dinghuai Zhang, Jian Tang, Alexandre Piché, Nicolas Gontier, Yoshua Bengio, and Ehsan Kamalloo. Self-evolving curriculum for LLM reasoning. *arXiv preprint arXiv:2505.14970*, 2025. URL https://arxiv.org/abs/2505.14970.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Simin Fan, David Grangier, and Pierre Ablin. Dynamic gradient alignment for online data mixing. *arXiv preprint arXiv:2410.02498*, 2024. URL https://arxiv.org/abs/2410.02498.

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

Albert Ge, Tzu-Heng Huang, John Cooper, Avi Trost, Ziyi Chu, Satya Sai Srinath Namburi GNVV, Ziyang Cai, Kendall Park, Nicholas Roberts, and Frederic Sala. R&B: Domain regrouping and data mixture balancing for efficient foundation model training. *arXiv preprint arXiv:2505.00358*, 2025. URL https://openreview.net/forum?id=lRv2cfg3u5.

Zhengyao Gu, Jonathan Light, Raul Astudillo, Ziyu Ye, Langzhou He, Henry Peng Zou, Wei Cheng, Santiago Paternain, Philip S. Yu, and Yisong Yue. Actor-curator: Co-adaptive curriculum learning via policy-improvement bandits for RL post-training. *arXiv preprint arXiv:2602.20532*, 2026. URL https://arxiv.org/abs/2602.20532.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024. URL https://arxiv.org/abs/2403.07974.

Feiyang Kang, Yifan Sun, Bingbing Wen, Si Chen, Dawn Song, Rafid Mahmood, and Ruoxi Jia. AutoScale: Scale-aware data mixing for pre-training LLMs. *arXiv preprint arXiv:2407.20177*, 2024. URL https://openreview.net/forum?id=rujwIvjooA.

Krishnateja Killamsetty, Durga S, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *PMLR*, pp. 5464–5474, 2021a. URL https://proceedings.mlr.press/v139/killamsetty21a.html.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 8110–8118, 2021b. URL https://arxiv.org/abs/2012.10630.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *PMLR*, pp. 1885–1894, 2017. URL https://proceedings.mlr.press/v70/koh17a.html.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-Hard and BenchBuilder pipeline. *arXiv preprint arXiv:2406.11939*, 2024. URL https://arxiv.org/abs/2406.11939.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023. URL https://arxiv.org/abs/2305.20050.

LMArena. Arena-Hard-Auto: Arena-Hard v2.0 preview. GitHub repository, 2025. URL https://github.com/lmarena/arena-hard-auto. Arena-Hard v2.0 released April 23, 2025.

Jing Ma, Chenhao Dang, and Mingjie Liao. AC-ODM: Actor–critic online data mixing for sample-efficient LLM pretraining. *arXiv preprint arXiv:2505.23878*, 2025. URL https://arxiv.org/abs/2505.23878. ICML 2026 poster.

Mathematical Association of America. MAA invitational competitions, 2026. URL https://maa.org/maa-invitational-competitions/. Accessed 2026-07-18.

MiniMax. MiniMax-M1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*, 2025. URL https://arxiv.org/abs/2506.13585.

Maximilian Mordig, Andreas Opedal, Weiyang Liu, and Bernhard Schölkopf. Rethinking easy-to-hard: Limits of curriculum learning in post-training for deductive reasoning. *arXiv preprint arXiv:2603.27226*, 2026. URL https://arxiv.org/abs/2603.27226.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. TRAK: Attributing model behavior at scale. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *PMLR*, pp. 27074–27113, 2023. URL https://arxiv.org/abs/2303.14186.

Prime Intellect Team, Sami Jaghouar, Justus Mattern, Jack Min Ong, Jannik Straube, Manveer Basra, Aaron Pazdera, Kushal Thaman, Matthew Di Ferrante, Felix Gabriel, Fares Obeid, Kemal Erdem, Michael Keiblinger, and Johannes Hagemann. INTELLECT-2: A reasoning model trained through globally decentralized reinforcement learning. *arXiv preprint arXiv:2505.07291*, 2025. URL https://arxiv.org/abs/2505.07291.

Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems 33*, 2020. URL https://arxiv.org/abs/2002.08484.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

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

Zhenting Wang, Guofeng Cui, Yu-Jhe Li, Kun Wan, and Wentian Zhao. DUMP: Automated distribution-level curriculum learning for RL-based LLM post-training. *arXiv preprint arXiv:2504.09710*, 2025. URL https://arxiv.org/abs/2504.09710.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. DoReMi: Optimizing data mixtures speeds up language model pretraining. In *Advances in Neural Information Processing Systems 36*, 2023. URL https://arxiv.org/abs/2305.10429.

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. *arXiv preprint arXiv:2409.12122*, 2024. URL https://arxiv.org/abs/2409.12122.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Jiasheng Ye, Peiju Liu, Tianxiang Sun, Jun Zhan, Yunhua Zhou, and Xipeng Qiu. Data mixing laws: Optimizing data mixtures by predicting language modeling performance. *arXiv preprint arXiv:2403.16952*, 2024. URL https://arxiv.org/abs/2403.16952.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025a. URL https://arxiv.org/abs/2503.14476.

Qiying Yu et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025b. URL https://arxiv.org/abs/2503.14476.

Mozhi Zhang, Howe Tissue, Lu Wang, and Xipeng Qiu. Domain2Vec: Vectorizing datasets to find the optimal data mixture without training. *arXiv preprint arXiv:2506.10952*, 2025. URL https://arxiv.org/abs/2506.10952.

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

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

Zhanke Zhou, Xiangyu Lu, Chentao Cao, Brando Miranda, Tongliang Liu, Bo Han, and Sanmi Koyejo. The easy, the hard, and the learnable: Confidence and difficulty-adaptive policy optimization for LLM reasoning. *arXiv preprint arXiv:2606.07950*, 2026. URL https://arxiv.org/abs/2606.07950.

A Expanded Method and Signal Definitions

This appendix specifies the cost signals, mixture-form utility, and state interfaces used in Section 4. Algorithm 1 gives the execution procedure; Appendix E.4 collects the numerical settings.

A.1 Allocating Domain Quotas

Let p_c be the domain prototype, $p_\star$ the target prototype, and $\bar{R}_{c,t}$ the domain's running reward average. The allocation cost is

$$\mathrm{Cost}_c = -\mathrm{cos}_{c,t} + \mathrm{Reward}_{c,t}\,. \tag{A.1}$$

Here $\mathrm{cos}_{c,t}$ and $\mathrm{Reward}_{c,t}$ are the separately normalized prototype alignment and mapped reward $\psi_{\mathrm{cost}}(\bar{R}_{c,t})$, respectively; Appendix A.4 defines their arithmetic. For a fixed positive reference mixture q and $\tau_\rho > 0$, the continuous allocation is

$$\begin{aligned} \rho_t^\star &\in \arg\min_{\rho\in\Delta^{K-1}} \sum_c \rho_c\,\mathrm{Cost}_c + \tau_\rho\,\mathrm{KL}(\rho\|q), \\ \rho_{c,t}^\star &= \frac{q_c \exp(-\,\mathrm{Cost}_c\,/\tau_\rho)}{\sum_j q_j \exp(-\,\mathrm{Cost}_j\,/\tau_\rho)}. \end{aligned} \tag{A.2}$$

Equation (A.10) converts these shares to integer quotas; Algorithm 1 handles eligible-supply constraints.

A.2 Choosing Cases Together

For the candidate window $\mathcal{C}_{c,t}$, let S_i be the individual score, z_i the residual sketch, and $\widehat{G}_t \succeq 0$ the anchor Gram matrix. The set utility in Eq. (3) extends to mixture weights $x \in \Delta(\mathcal{C}_{c,t})$ as

$$\mathcal{V}_{c,t}(x) := \sum_{i\in\mathcal{C}_{c,t}} x_i S_i - \frac{1}{2}\left(\sum_{i\in\mathcal{C}_{c,t}} x_i z_i\right)^\top \widehat{G}_t \left(\sum_{i\in\mathcal{C}_{c,t}} x_i z_i\right), \qquad x \in \Delta(\mathcal{C}_{c,t}). \tag{A.3}$$

Assigning equal weights to the selected cases recovers the set utility. After k draws, the prefix is $\mathcal{B}_{c,t}^{(k)}$ with mean $\bar{z}_{c,t}^{(k)}$, and the remaining window is $\mathcal{C}_{c,t}^{(k)} = \mathcal{C}_{c,t} \setminus \mathcal{B}_{c,t}^{(k)}$. Using $\Gamma_{i|k}$ from Eq. (4) and $T_s > 0$, the next draw and window update are

$$\Pr(i_{k+1} = i \mid \mathcal{B}_{c,t}^{(k)}, \mathcal{C}_{c,t}^{(k)}) = \frac{\exp(\Gamma_{i|k}/T_s)}{\sum_{j\in\mathcal{C}_{c,t}^{(k)}} \exp(\Gamma_{j|k}/T_s)}, \qquad \mathcal{C}_{c,t}^{(k+1)} = \mathcal{C}_{c,t}^{(k)} \setminus \{i_{k+1}\}. \tag{A.4}$$

The individual score combines target relevance, reward-based learning room, and the sketch-space quadratic coordinate:

$$S_i = \lambda_{\mathrm{rel}}\widetilde{\mathrm{rel}}_i + \lambda_{\mathrm{learn}}\widetilde{\mathrm{learn}}_i - \lambda_{\mathrm{curv}}\widetilde{\mathrm{Curv}}_i^{\,sk}. \tag{A.5}$$

The tildes denote separate normalization over the original domain window; the weights are nonnegative and sum to one. Appendix A.4 defines the coordinates, normalization, and prefix update.

A.3 Updating from Training Feedback

Once each quota is filled, $\mathcal{B}_{c,t} = \mathcal{B}_{c,t}^{(n_{c,t})}$ and $\mathcal{B}_t = \cup_c \mathcal{B}_{c,t}$. Rewards from the GRPO update refresh domain and case histories for the next selector call; external representations, sketches, anchors, and Gram state refresh on their configured schedules. Equation (A.16) specifies the state interface, and Algorithm 1 specifies persistent-pool removal and reset.

A.4 Signal Arithmetic and State Interfaces

The raw domain alignment and feedback cost are

$$c_{c,t}^{\mathrm{raw}} = \frac{p_c^\top p_\star}{\|p_c\|_2\|p_\star\|_2}. \tag{A.6}$$

$$r_{c,t}^{\mathrm{raw}} = \psi_{\mathrm{cost}}(\bar{R}_{c,t}). \tag{A.7}$$

Each coordinate is normalized over its comparison set $\mathcal{S}$:

$$\mathrm{Norm}_{\mathcal{S}}(s)_i = \mathrm{clip}\left(\mathbf{1}\{\sigma_{\mathcal{S}}(s) \geq \epsilon\}\frac{s_i - \mu_{\mathcal{S}}(s)}{\sigma_{\mathcal{S}}(s) + \epsilon}, -a, a\right). \tag{A.8}$$

Here μ and σ are the mean and population standard deviation, $\epsilon > 0$ stabilizes the denominator, and a bounds the normalized coordinate. Near-constant coordinates become zero. For allocation, the comparison set is all K domains:

$$\mathrm{cos}_{c,t} = \mathrm{Norm}_{\{1,\dots,K\}}(c^{\mathrm{raw}})_c, \qquad \mathrm{Reward}_{c,t} = \mathrm{Norm}_{\{1,\dots,K\}}(r^{\mathrm{raw}})_c. \tag{A.9}$$

The shares are rounded into counts by assigning floors first, then the remaining slots in descending fractional-remainder order:

$$\widetilde{n}_{c,t} = B\rho_{c,t}^{\star}, \qquad n_{c,t} = \mathrm{round}_B(\widetilde{n}_{c,t}), \qquad \sum_c n_{c,t} = B. \tag{A.10}$$

Exact fractional ties follow fixed domain order in the selector reference. Quotas are capped at eligible supply; deficits are redistributed using the original shares restricted to domains with spare capacity.

For cached representation $h_i \in \mathbb{R}^d$, a fixed Gaussian projection compresses the deviation from the domain prototype:

$$z_i = P^{\top}(h_i - p_{c(i)}), \qquad P \in \mathbb{R}^{d\times m}, \quad P_{uv} \sim \mathcal{N}(0, 1/d), \quad m \ll d. \tag{A.11}$$

Shadow anchors A_t use the same projection and are excluded from the training batch. Their average outer products supply the Gram state:

$$\widehat{G}_t = \mathrm{EMA}\left(\frac{1}{|A_t|}\sum_{a\in A_t} z_a z_a^{\top}\right), \qquad \mathrm{Curv}_i^{sk} = z_i^{\top}\widehat{G}_t z_i. \tag{A.12}$$

Positive semidefinite initialization and nonnegative EMA weights preserve $\widehat{G}_t \succeq 0$. The quadratic coordinate aggregates squared inner products with the anchor sketches. The other individual coordinates are

$$\mathrm{rel}_i = \frac{h_i^{\top} p_{\star}}{\|h_i\|_2 \|p_{\star}\|_2}, \qquad \mathrm{learn}_i = \psi_{\mathrm{learn}}(\bar{R}_{i,t}). \tag{A.13}$$

The score in Eq. (A.5) normalizes these coordinates over the original candidate window. Scores, normalizations, sketches, and Gram state stay fixed within each quota; the diagonal correction in Eq. (4) uses the unnormalized quadratic coordinate, including at the first draw.

After drawing i_{k+1}, update only the prefix mean and remaining window:

$$\bar{z}_{c,t}^{(k+1)} = \frac{k\bar{z}_{c,t}^{(k)} + z_{i_{k+1}}}{k+1}, \qquad \mathcal{C}_{c,t}^{(k+1)} = \mathcal{C}_{c,t}^{(k)} \setminus \{i_{k+1}\}. \tag{A.14}$$

Initially $\bar{z}_{c,t}^{(0)} = 0$. After $n_{c,t}$ draws, the set $\mathcal{B}_{c,t}$ fills the quota. Selected IDs leave both temporary and persistent pools; unselected IDs retain their order. For history state $\mathcal{H}_t$ and pool $\mathcal{D} = \cup_c \mathcal{D}_c$, the full online interface is

$$(\mathcal{D}, \mathcal{H}_t) \xrightarrow{\textsc{TaylorMix}} \{n_{c,t}, \mathcal{B}_{c,t}\}_{c=1}^{K} \xrightarrow{\cup} \mathcal{B}_t \xrightarrow{\mathrm{GRPO}} (\theta_{t+1}, \mathcal{H}_{t+1}). \tag{A.15}$$

$$\mathcal{H}_{t+1} = \mathrm{Update}\left(\mathcal{H}_t, \mathcal{B}_t, \mathrm{Rollout}_{\theta_t}(\mathcal{B}_t)\right). \tag{A.16}$$

B Expanded Analysis and Exact Identities

We give the exact KL identities underlying the bounds in Section 5. Full proofs and auxiliary bounds are in Appendix C.

B.1 Coverage-Preserving Domain Allocation

For the continuous allocation in Eq. (A.2), the cost improvement over q equals a symmetric KL divergence:

Theorem 1 (Coverage–utility trade-off). Under the Method allocation in Eq. (A.2), assume finite Cost_c, positive q_c, and $\tau_\rho > 0$, and let $D = \max_c \mathrm{Cost}_c - \min_c \mathrm{Cost}_c$. Then

$$\sum_c (q_c - \rho^\star_{c,t})\,\mathrm{Cost}_c = \tau_\rho\big[\,\mathrm{KL}(\rho^\star_t \| q) + \mathrm{KL}(q \| \rho^\star_t)\big] \geq 0. \tag{B.1}$$

Every domain retains the quantitative share

$$\rho^\star_{c,t} \geq \frac{q_c}{q_c + (1-q_c)e^{D/\tau_\rho}} > 0. \tag{B.2}$$

The share floor applies before rounding and supply adjustment. Appendix C.1 gives a sufficient condition for positive integer quotas and a sensitivity bound for changing costs.

B.2 Prefix-Conditional Batch Selection

Fix domain c, step t, and prefix $\mathcal{B}^{(k)}_{c,t}$. For $k > 0$, let $x^{(k)}$ assign weight $1/k$ to each selected case. Adding $j \in \mathcal{C}^{(k)}_{c,t}$ gives $x^{(k+1)}_j = (kx^{(k)} + e_j)/(k+1)$, where e_j is the unit vector at j. Using Eq. (A.3), define $\Delta_{j|k} = \mathcal{V}_{c,t}(x^{(k+1)}_j) - \mathcal{V}_{c,t}(x^{(k)})$ and the next-set penalty

$$\mathcal{R}_{j|k} = \frac{1}{2}\left(\frac{k\bar{z}^{(k)}_{c,t} + z_j}{k+1}\right)^\top \widehat{G}_t \left(\frac{k\bar{z}^{(k)}_{c,t} + z_j}{k+1}\right). \tag{B.3}$$

At $k = 0$, use $\bar{z}^{(0)}_{c,t} = 0$, $x^{(0)} = 0$, and $\mathcal{V}_{c,t}(x^{(0)}) = 0$. The comparison below holds the remaining window, individual scores, and temperature fixed.

Theorem 2 (Conditional penalty and utility). For the Method prefix $\mathcal{B}^{(k)}_{c,t}$ and its finite nonempty remaining window $\mathcal{C}^{(k)}_{c,t}$, hold finite S_i, z_i, and $\widehat{G}_t \succeq 0$ fixed. For $T_s > 0$, compare the conditional sampler p in Eq. (A.4) with the same-window sampler $p^0_j \propto e^{S_j/T_s}$ that omits the batch correction. Then

$$\mathbb{E}_{p^0}\mathcal{R}_{j|k} - \mathbb{E}_p\mathcal{R}_{j|k} = \frac{T_s}{k+1}\big[\,\mathrm{KL}(p\|p^0) + \mathrm{KL}(p^0\|p)\big] \geq 0. \tag{B.4}$$

Moreover, p uniquely maximizes $F_k(r) = \mathbb{E}_r\Delta_{j|k} + \frac{T_s}{k+1}\mathsf{H}(r)$, where $\mathsf{H}(r) = -\sum_j r_j \log r_j$. For every distribution r on the remaining window,

$$F_k(p) - F_k(r) = \frac{T_s}{k+1}\,\mathrm{KL}(r\|p) \geq 0. \tag{B.5}$$

The penalty reduction is strict exactly when the next-set penalties differ across candidates. Equation (D.4) connects the utility increment to the implemented score through a candidate-independent offset, yielding the entropy coefficient $T_s/(k+1)$. These are conditional, one-step statements at a fixed prefix.

B.3 Set Notation and the Main-Text Conclusions

For a nonempty $\mathcal{B} \subseteq \mathcal{C}_{c,t}$, let $x_\mathcal{B}$ assign weight $1/|\mathcal{B}|$ to its members and zero elsewhere. Then

$$\mathcal{V}(\mathcal{B}) = \mathcal{V}_{c,t}(x_\mathcal{B}), \qquad \bar{z}_\mathcal{B} = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} z_i, \qquad \mathcal{R}(\mathcal{B}) = \frac{1}{2}\bar{z}_\mathcal{B}^\top \widehat{G}_t \bar{z}_\mathcal{B}. \tag{B.6}$$

Domain and step indices are suppressed only in the set notation. Set $\mathcal{V}(\varnothing) = \mathcal{R}(\varnothing) = 0$ and $\bar{z}_\varnothing = 0$. At Method's current prefix $\mathcal{B} = \mathcal{B}^{(k)}_{c,t}$, the quantities used above are exactly

$$\Delta_{i|k} = \mathcal{V}(\mathcal{B}\cup\{i\}) - \mathcal{V}(\mathcal{B}), \qquad \mathcal{R}_{i|k} = \mathcal{R}(\mathcal{B}\cup\{i\}). \tag{B.7}$$

Dropping the nonnegative KL terms yields the cost inequality in Eq. (6) and the penalty inequality in Eq. (7); the continuous-share floor is unchanged. The objective in Eq. (8) is $F_k(r)$. Equation (D.5) gives the sampler's exponential-tilt form.

When a lower-scoring case is the better addition. For two candidates i, j in the same remaining window, the candidate-independent terms cancel:

$$\Gamma_{i|k} - \Gamma_{j|k} = S_i - S_j - (k+1)\big[\mathcal{R}(\mathcal{B} \cup \{i\}) - \mathcal{R}(\mathcal{B} \cup \{j\})\big]. \tag{B.8}$$

Thus, if $S_i < S_j$, candidate i has both a larger exact utility increment and a larger conditional sampling probability precisely when

$$\mathcal{R}(\mathcal{B} \cup \{j\}) - \mathcal{R}(\mathcal{B} \cup \{i\}) > \frac{S_j - S_i}{k+1}. \tag{B.9}$$

This condition quantifies when the enlarged-set penalty advantage outweighs the difference in individual score.

Framework schematic. The right panel of Figure 2 illustrates the same conditional score with one-dimensional coordinates. Fix $k = 1$, $S_i = 0.80$, $z_i = 1$, and $\widehat{G}_t = 1$. Two alternative selected sets have means $+0.60$ and -0.60, giving $\Gamma_{i|1} = 0.80 - 0.30 - 0.25 = 0.25$ and $\Gamma_{i|1} = 0.80 + 0.30 - 0.25 = 0.85$, respectively. The candidate and individual score are identical; only its companions change. These are illustrative states, not successive draws or experimental measurements.

C Proofs and Auxiliary Bounds

We prove the allocation and conditional selection guarantees in Section 5, using the quantities and construction rules defined in Method.

C.1 Proof of Theorem 1

Proof. Fix step t and write $C_c = \mathrm{Cost}_c$ and $\rho = \rho_t^\star$. The domain objective is

$$J(r) = \langle r, C \rangle + \tau_\rho \,\mathrm{KL}(r \| q)$$

over the probability simplex. Let $Z_\rho = \sum_c q_c e^{-C_c/\tau_\rho}$. Finite costs, $q_c > 0$, and $\tau_\rho > 0$ give

$$\rho_c = \frac{q_c e^{-C_c/\tau_\rho}}{Z_\rho} > 0, \qquad \log \frac{\rho_c}{q_c} = -\frac{C_c}{\tau_\rho} - \log Z_\rho. \tag{D.1}$$

Substitution into J gives, for every probability distribution r,

$$J(r) - J(\rho) = \tau_\rho \,\mathrm{KL}(r \| \rho) \geq 0. \tag{D.2}$$

Equality holds exactly at $r = \rho$, proving uniqueness.

Utility and coverage. Multiply the log-ratio identity in Eq. (D.1) by $\rho_c - q_c$ and sum. Since $\sum_c (\rho_c - q_c) = 0$,

$$\mathrm{KL}(\rho \| q) + \mathrm{KL}(q \| \rho) = -\frac{\langle \rho - q, C \rangle}{\tau_\rho},$$

which proves Eq. (B.1). If $c_\star \in \arg\min_c C_c$, comparison with $r = e_{c_\star}$ in Eq. (D.2) also gives

$$0 \leq \langle \rho, C \rangle - C_{c_\star} \leq \tau_\rho \log(1/q_{c_\star}).$$

Thus the cost of retaining coverage relative to pure greedy allocation is bounded. To establish the share floor, write

$$\rho_c = \frac{q_c}{q_c + \sum_{d \neq c} q_d e^{(C_c - C_d)/\tau_\rho}} \geq \frac{q_c}{q_c + (1 - q_c) e^{D/\tau_\rho}} = L_c.$$

Equality is attained when $C_c - C_d = D$ for every $d \neq c$, so the bound is sharp given the cost range.

Integer quotas. Largest-remainder rounding first assigns $\lfloor B\rho_c \rfloor$ slots and then distributes the residual budget. Hence $\sum_c n_{c,t} = B$ and $n_{c,t} \geq \lfloor B\rho_c \rfloor \geq \lfloor BL_c \rfloor$ before any supply adjustment. If $BL_c \geq 1$ for every domain and eligible supply meets the quotas, all integer quotas are positive. Method's two clipped coordinates imply $D \leq 4a$. For uniform q and $B > K > 1$, the explicit sufficient condition

$$\tau_\rho \geq \frac{4a}{\log((B-1)/(K-1))}$$

follows by substituting $L_c \geq [1 + (K-1)e^{4a/\tau_\rho}]^{-1}$. □

Allocation stability. At fixed q, τ_ρ, allocation satisfies

$$\|\rho(C') - \rho(C)\|_1 \leq \min\left\{2, \frac{\operatorname{osc}(C'-C)}{2\tau_\rho}\right\}, \qquad \operatorname{osc}(v) := \max_c v_c - \min_c v_c. \tag{D.3}$$

Temperature therefore controls sensitivity as well as concentration. For fixed q, τ_ρ, set $\delta = C' - C$ and consider $C(s) = C + s\delta$, $s \in [0,1]$. Differentiating the normalized exponential tilt yields

$$\frac{d\rho_c(s)}{ds} = -\frac{\rho_c(s)}{\tau_\rho}\left(\delta_c - \mathbb{E}_{\rho(s)}\delta\right).$$

For a random variable $X \in [a_0, b_0]$ with mean μ, $(X - a_0)(b_0 - X) \geq 0$ implies $\operatorname{Var}(X) \leq (\mu - a_0)(b_0 - \mu) \leq (b_0 - a_0)^2/4$. Cauchy–Schwarz therefore gives

$$\left\|\frac{d\rho(s)}{ds}\right\|_1 = \frac{\mathbb{E}_{\rho(s)}|\delta - \mathbb{E}_{\rho(s)}\delta|}{\tau_\rho} \leq \frac{\operatorname{osc}(\delta)}{2\tau_\rho}.$$

Integration over $s \in [0,1]$, together with the maximum ℓ_1 distance of two probability distributions, proves Eq. (D.3).

C.2 Proof of Theorem 2

Proof. For this proof only, suppress the fixed domain and step indices: $G = \widehat{G}_t$, $\mathcal{V} = \mathcal{V}_{c,t}$, $R_j = \mathcal{R}_{j|k}$, and $\Delta_j = \Delta_{j|k}$. With $m(x) = \sum_i x_i z_i$, Method defines

$$\mathcal{V}(x) = \sum_{i \in \mathcal{C}} x_i S_i - \frac{1}{2} m(x)^\top G m(x).$$

The matrix G is positive semidefinite by construction: each anchor outer product satisfies $v^\top z_a z_a^\top v = (z_a^\top v)^2 \geq 0$, and nonnegative EMA combinations preserve this property from a positive semidefinite initialization. Thus $R_j \geq 0$ and $\mathcal{V}$ is concave in the mixture x.

Exact finite-step change. At a fixed prefix, let $\bar{S}_k = \sum_i x_i^{(k)} S_i$ and $m_k = m(x^{(k)})$. Direct expansion gives

$$\begin{aligned}\Delta_j &= \frac{k\bar{S}_k + S_j}{k+1} - \bar{S}_k - \frac{(km_k + z_j)^\top G(km_k + z_j)}{2(k+1)^2} + \frac{1}{2} m_k^\top G m_k \\ &= \frac{1}{k+1}\left[S_j - \frac{k}{k+1} z_j^\top G m_k - \frac{z_j^\top G z_j}{2(k+1)} - \bar{S}_k + \frac{2k+1}{2(k+1)} m_k^\top G m_k\right].\end{aligned}$$

The first three terms are Method's corrected score. Equivalently,

$$(k+1)\Delta_j = \Gamma_{j|k} - \bar{S}_k + \frac{2k+1}{2(k+1)} m_k^\top G m_k.$$

The last two terms are shared by all candidates. Consequently,

$$\Delta_j = \frac{\Gamma_{j|k} + b_k}{k+1}, \qquad b_k = -\bar{S}_k + \frac{2k+1}{2(k+1)} m_k^\top G m_k. \tag{D.4}$$

The offset b_k is independent of j, so the ordering of $\Gamma_{j|k}$ is exactly the ordering of the finite-step increments. At $k = 0$, the polynomial extension gives $\bar{S}_0 = 0$, $m_0 = 0$, $b_0 = 0$ and $\Delta_j = S_j - z_j^\top G z_j/2 = \Gamma_{j|0}$.

Conditional penalty reduction. Expand the next-set quadratic:

$$(k+1)R_j = \frac{k^2 m_k^\top G m_k}{2(k+1)} + \frac{k}{k+1} z_j^\top G m_k + \frac{z_j^\top G z_j}{2(k+1)}.$$

Its first term is candidate-independent, and therefore $\Gamma_{j|k} = S_j - (k+1)R_j + k^2 m_k^\top G m_k/[2(k+1)]$. Normalizing the exponentials gives

$$p_j = \frac{p_j^0 e^{-(k+1)R_j/T_s}}{\sum_\ell p_\ell^0 e^{-(k+1)R_\ell/T_s}}. \tag{D.5}$$

Multiply $\log(p_j/p_j^0)$ by $p_j - p_j^0$ and sum. The log normalizer cancels, leaving

$$\mathrm{KL}(p\|p^0) + \mathrm{KL}(p^0\|p) = \frac{k+1}{T_s}\left(\mathbb{E}_{p^0}R - \mathbb{E}_p R\right).$$

This is Eq. (B.4). Both distributions have full support, so equality holds exactly when $p = p^0$, equivalently when every remaining candidate has the same R_j.

Finite-step utility and exploration. By Eq. (D.4),

$$F_k(r) = \frac{b_k + \sum_j r_j \Gamma_{j|k} + T_s \mathsf{H}(r)}{k+1}.$$

Let $Z_s = \sum_j e^{\Gamma_{j|k}/T_s}$. Substituting $\log p_j = \Gamma_{j|k}/T_s - \log Z_s$ gives

$$(k+1)[F_k(p) - F_k(r)] = T_s \log Z_s - \sum_j r_j \Gamma_{j|k} - T_s \mathsf{H}(r) = T_s\,\mathrm{KL}(r\|p).$$

This proves Eq. (B.5) and uniqueness; $0 \log 0 = 0$ covers zero coordinates of r. Averaging Eq. (D.4) also gives $\mathbb{E}_p\Delta = (\sum_j p_j \Gamma_{j|k} + b_k)/(k+1)$. For M_k remaining candidates, choose r concentrated at a largest increment. Since $\mathsf{H}(p) \le \log M_k$,

$$0 \le \max_j \Delta_j - \mathbb{E}_p\Delta \le \frac{T_s \log M_k}{k+1}. \tag{D.6}$$

Thus the same temperature that controls exploration bounds its cost relative to the best next-set increment. □

C.3 Taylor Motivation and One-Step Expansion

For the batch-induced direction $u = g_t(\mathcal{B}_t; \omega_t)$ in Section 3, with M denoting the evaluation loss, write $\theta = \theta_t$, $g = \nabla M(\theta_t)$, and $H = \nabla^2 M(\theta_t)$. With $\Delta\theta = -\eta u$, the one-step utility is

$$\mathcal{T}(u) := M(\theta) - M(\theta - \eta u) = \eta g^\top u - \frac{\eta^2}{2} u^\top H u - R_3(u). \tag{D.7}$$

The linear term parallels GLISTER's validation-improvement approximation (Killamsetty et al., 2021b). Both terms evaluate the same complete-batch update, so isolated case scores need not capture composition. This motivates batch-aware selection. The generic expansion and remainder are given in Eqs. (D.8)–(D.9); Section 5 analyzes the resulting domain-cost and geometric-utility decision rules.

Expansion and remainder. For completeness, the generic expansion and the induced loss change are

$$M(\theta + \Delta\theta) = M(\theta) + g^\top \Delta\theta + \frac{1}{2}\Delta\theta^\top H \Delta\theta + \widetilde{R}_3(\Delta\theta). \tag{D.8}$$

and, with $\Delta\theta = -\eta u$ and $R_3(u) := \widetilde{R}_3(-\eta u)$,

$$M(\theta - \eta u) - M(\theta) = -\eta g^\top u + \frac{\eta^2}{2} u^\top H u + R_3(u). \tag{D.9}$$

Here $g = \nabla M(\theta)$, $H = \nabla^2 M(\theta)$, and the remainder collects the terms beyond the displayed quadratic expansion.

D EXPERIMENTAL DETAILS

This appendix expands the evaluation protocol in Section 6 and the geometric-selection intervention in Section 7.

D.1 EVALUATION DATASETS

We evaluate mathematical reasoning on MATH-500, a representative subset of the challenging MATH test set for multi-step competition-style problem solving (Lightman et al., 2023), and AIME 2024/2025, challenging integer-answer invitational competitions for students who excel on AMC 10 or AMC 12 (Mathematical Association of America, 2026; Yang et al., 2024; 2025). Code capability is evaluated with LiveCodeBench (LCB) (Jain et al., 2024), a continuously updated benchmark of programming problems that covers code generation, code execution, and test-output prediction while supporting evaluation on problems released after a model's training cutoff, and HumanEval (Chen et al., 2021), which measures functional correctness on 164 hand-written Python programming problems. General capability is evaluated on Arena-Hard v2.0 (Li et al., 2024; LMArena, 2025), which combines challenging real-world user queries, including open-ended software-engineering and mathematics requests, with creative-writing prompts sourced from Chatbot Arena.

D.2 METRIC DEFINITIONS AND AGGREGATION

MATH-500 Pass@1 is the percentage of one-response exact answers accepted by the common grader. AIME Avg@k averages correctness over k samples, whereas Pass@k counts problems solved by at least one sample. Qwen3-2B uses Avg@8 and Qwen2.5-1.5B uses Pass@64, so comparisons stay within each backbone and metric. Each LCB task reports Pass@1: a generated program must pass all tests, code execution must predict the literal output for a function and input, and test-output prediction must return the expected output for a test case. HumanEval Pass@1 is the percentage of problems for which the first generated completion passes all unit tests. The code-capability Mean in Table 1 averages the three LCB percentages and HumanEval Pass@1; LCB Mean in the ablation tables averages only the three LCB percentages. Our order-balanced Arena WinScore compares against a fixed reference and maps decisive win, slight win, tie, slight loss, and decisive loss to 1, $2/3$, $1/2$, $1/3$, and 0. We average both response orders over valid prompts and report percentages; Arena-Hard and Arena-Creative use the same rule on their respective subsets. Within each comparison, all methods use the same generation and evaluation procedure. Unless stated otherwise, values are percentages and larger is better.

D.3 BASELINE DESCRIPTIONS

The baselines cover standard RLVR, domain-level allocation, and case-level selection:

- **Standard baselines. Base** is the unadapted model and marks capability before RLVR training. **Vanilla GRPO** (Shao et al., 2024) uses the same GRPO update, reward computation, and training budget but removes Taylor-guided data selection, testing whether standard RLVR alone explains the gains.
- **Domain-level baselines. Domain2Vec** (Zhang et al., 2025) uses dataset vectors to determine a training-free mixture, testing static representation-based allocation. **AC-ODM** (Ma et al., 2025) uses an actor–critic controller to update online domain weights, testing learned domain control without case-level geometric selection.
- **Case-level baselines. GRESO** (Zheng et al., 2025b) uses reward dynamics to skip uninformative prompts before rollout. **CoDaPO** (Zhou et al., 2026) uses confidence and empirical difficulty to reweight updates and resample learnable questions. They test whether online case selection without explicit domain allocation is sufficient.

Training pool and reward source. The shared data configuration uses the frozen JSON mixture `Mix_AirRep_nonrepeated_sysprom.json`. Its input snapshot contains $18{,}035$ prompts: $5{,}582$ Math, $4{,}909$ Code, and $7{,}544$ General prompts, before tokenizer-dependent filtering and anchor exclusion. The source labels are `math`, `code`, and `wildchat-if`, respectively, and all entries carry the `train` split tag. Math and Code entries provide rule-based ground-truth fields; General entries specify model-based rewards. The configured General reward backbone is

Table 1: Per-case training time of TaylorMix and representative RLVR baselines under matched hardware (seconds). Best and second-best results are boldfaced and underlined, respectively.

Method	Generation	Reward	Selection	Update	Other	Total
Base	28.66	**3.24**	**1.60**	**4.22**	<u>0.22</u>	**37.94**
AC-ODM	23.31	5.48	<u>4.01</u>	<u>8.83</u>	0.35	41.97
CoDaPO	24.41	5.97	4.42	9.72	0.45	44.96
GRESO	**11.86**	18.64	15.44	36.54	**0.15**	82.63
TaylorMix	<u>18.82</u>	<u>3.60</u>	5.06	11.63	0.38	<u>39.49</u>

`Skywork/Skywork-Reward-Llama-3.1-8B-v0.2`. This training reward source is distinct from the pairwise Arena evaluation in Appendix D.2.

E Implementation and Computational Cost

This appendix covers trainer integration, runtime, storage, and reference-code conventions. Signal arithmetic and the selection procedure are specified in Appendix A.4 and Algorithm 1, respectively.

E.1 Rollout-Pre Execution Path

One selector call reads domain-specific case-ID queues, cached summaries, and configuration, then returns the ID list $\mathcal{B}_t$ before rollout. The trainer deterministically shards this list across workers; generation, reward computation, advantage construction, and GRPO optimization are unchanged. Selection requires neither speculative rollout nor optimizer-side selection state.

E.2 Time Analysis

Table 1 reports component costs estimated under a four-GPU setting and normalized by the number of prompts. Selection includes batched representation extraction and the remaining pre-rollout operations. The reported total is 39.49 seconds per case for TaylorMix, compared with 37.94 for Base and 41.97–82.63 for the listed selection baselines.

E.3 Computational Complexity and Storage

Let K be the number of domains, $n_t = \sum_c |\mathcal{C}_{c,t}|$ the number of candidates scored at step t, m the sketch dimension, and $|A_t|$ the number of shadow anchors. The following bounds assume that the required scalar coordinates and sketches are available; representation extraction is included in the measured runtime above.

Evaluating one cached cost per domain takes $O(K)$. Combining and normalizing the scalar linear coordinates takes $O(n_t)$, while evaluating one dense sketch-space quadratic coordinate per candidate takes $O(n_t m^2)$. Refreshing the shared Gram matrix from cached anchor information takes $O(|A_t|m^2)$, after which the matrix is reused across selection steps. Thus, the displayed allocation and score-construction arithmetic is $O(K + n_t m^2)$, with matrix refresh and weighted sampling kept outside this bound and included in the end-to-end runtime measurement.

Conditional rescoring reuses the fixed quadratic coordinates and recomputes interactions with the current prefix mean. Softmax normalization and sampling operate on the remaining window, so their total cost depends on both quota and window sizes and is not included in the score-construction bound. At the configured $m = 64$, the Gram matrix contains $4{,}096$ scalar entries; quadratic-coordinate construction is linear in n_t and matrix refresh is linear in $|A_t|$ for fixed m.

TaylorMix stores domain and target prototypes, case-level summaries, the fixed projection or its seed, shadow-anchor state, and the Gram matrix. Case-level state grows linearly with the training pool. The Gram matrix and cached anchor sketches require $O(m^2 + |A_t|m)$ values.

E.4 SHARED TRAINING CONFIGURATION

Table 7 specifies the shared sequence-8192 training setup. Batch sizes count prompts before response expansion, yielding twelve rollout sequences per batch. The prompt and response limits allow 10,240 tokens in total. Preprocessing enables overlength filtering and right-side truncation. The data-selection rule is the intervention across trained comparisons. In panel (b), the percentage of all-correct or all-wrong rollout groups is computed over 1 groups with 30 cases.

Table 7: Shared training hyperparameters. Generation settings in this table apply to training rollouts.

Setting	Value
Prompt batch / generation batch	6 / 6
Responses per prompt	2
Training steps	200
Optimizer mini-batch (prompts)	3
Micro-batch per GPU (sequences)	1
PPO epochs per batch	1
Maximum prompt / response tokens	2,048 / 8,192
Training temperature / top-p	1.0 / 1.0
Training top-k	Disabled (-1)
Learning rate	10^{-6}
Learning-rate warmup	10 steps
Weight decay	0.1
Gradient-norm clipping	1.0
Entropy coefficient	0
Policy KL-loss coefficient	0.001
Loss aggregation	Token mean
Checkpoint save steps	$20, 50, 100, 150, 200$

F EXECUTION PROCEDURE

Algorithm 1 preserves Method's selection operations. For domain c, N_c is the eligible supply after anchor exclusions. Its ordered pool $\mathcal{Q}_c$ is shuffled once at initialization and retained across steps. If the remaining pool cannot fill the quota, it is reset before opening a candidate window of at least $n_{c,t}$ cases. A window equal to its quota is taken in full; otherwise draws follow the conditional softmax.

Algorithm 1: TAYLORMIX Online Batch Composition

Input: State $\mathcal{H}_t$ with persistent remaining-ID pools $\{\mathcal{Q}_c\}$, domains $\{\mathcal{D}_c\}_{c=1}^{K}$, batch size B, prototypes $\{p_c\}$ and $p_\star$, prior q, temperatures τ_ρ, T_s
Output: Selected batch $\mathcal{B}_t$, updated remaining-ID pools, and post-rollout state

```
for c = 1, ..., K do
    compute c_{c,t}^raw and r_{c,t}^raw using Eqs. (A.6)–(A.7);
normalize raw domain signals and compute Cost_c using Eq. (A.1);
solve ρ_t^⋆ with Eq. (1) and round quotas {n_{c,t}} using Eq. (A.10);
redistribute quotas exceeding N_c to domains with spare capacity according to ρ_t^⋆;
B_t ← ∅;
for c = 1, ..., K with n_{c,t} > 0 do
    if |Q_c| < min(n_{c,t}, N_c) then
        Q_c ← shuffled full list of eligible IDs in D_c;
    C_{c,t} ← Q_c in full-pool mode, or its configured prefix of at least n_{c,t} IDs;
    foreach i ∈ C_{c,t} do
        compute rel_i, learn_i, Curv_i^{sk} and score S_i using Eqs. (A.13), (A.12), and (A.5);
    B_{c,t} ← ∅ and z̄_{c,t}^{(0)} ← 0;
    if n_{c,t} ≥ |C_{c,t}| then
        B_{c,t} ← C_{c,t};
    else
        for k = 0, ..., n_{c,t} − 1 do
            compute the corrected conditional score Γ_{i|k} in Section 4 for remaining i;
            sample i_{k+1} from Eq. (5);
            C_{c,t} ← C_{c,t} \ {i_{k+1}}; B_{c,t} ← B_{c,t} ∪ {i_{k+1}};
            z̄_{c,t}^{(k+1)} ← (k z̄_{c,t}^{(k)} + z_{i_{k+1}}) / (k + 1);
    Q_c ← Q_c \ B_{c,t}, retaining the order of unselected IDs;
    B_t ← B_t ∪ B_{c,t};
persist {Q_c} across steps; shuffle the output ID list and return B_t before rollout;
after rollout update H_{t+1} using Eq. (A.16), retaining the sampler state;
```