FlipTS: Failure-to-Success Mathematical Self-Refinement via Trajectory-Aware Sampling in End-to-End Online RL

Anonymous ACL submission

Abstract

Self-Refinement has emerged as a promising paradigm for improving Large Language Model (LLM) performance by iteratively improving its response at inference time. Existing training methods typically treat refinement as a decoupled task and rely on pre-generated responses, thereby detaching training from the model's actual reasoning trajectories and hindering performance. In this paper, we propose **FlipTS** (**F**ailure-to-Success **L**earning for self-ref**I**nement with **P**otential-guided **T**rajectory-aware **S**ampling), an end-to-end reinforcement learning (RL) framework that optimizes the entire self-refinement loop online. We identify a fundamental bottleneck in this end-to-end setting: the sparsity of informative refinement behaviors *Failure-to-Success* (*F2S*). **FlipTS** addresses this by utilizing a non-stationary Bayesian model to estimate data potential and applying trajectory-aware sampling to enrich the training process with valuable refinement signals. Experiments on Qwen3-4B-Instruct and LLaMA-3.1-8B-Instruct across challenging mathematical benchmarks demonstrate that **FlipTS** consistently outperforms both offline and online RL baselines. Notably, it exhibits robust generalization to scientific reasoning and safety tasks without domain-specific tuning. Our code will be open-sourced.

1 Introduction

LLMs have demonstrated remarkable versatility across various tasks, particularly when equipped with a *Self-Refinement* mechanism. *Self-Refinement* (see Figure 1) allows the model to iteratively improve its responses and correct errors, effectively exploiting additional computation at inference time (Madaan et al., 2023; Shinn et al., 2023; Khatri et al., 2025). However, its efficacy is constrained by the model's limited verification and reasoning capabilities (Zhang et al., 2025). Consequently, it is imperative to enhance this self-correction proficiency through explicit training.

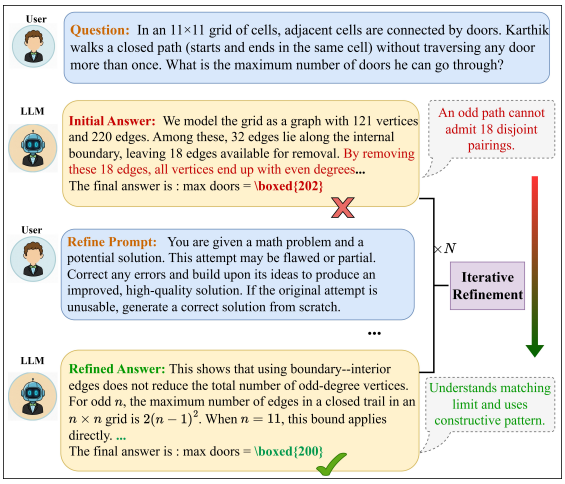

Figure 1: An example of *Self-Refinement*. The model diagnoses flaws in its initial answer and iteratively corrects them to achieve ground truth.

Recent advancements highlight the potential of outcome-based end-to-end RL (Guo et al., 2025; Wang et al., 2025; Xue et al.). Despite these successes, the *Self-Refinement* domain has yet to fully leverage this paradigm, specifically by treating the multi-turn *Self-Refinement* process as a unified trajectory. Existing approaches (Zeng et al., 2025; Huang et al., 2024; Hu et al., 2025) typically formulate refinement as an isolated, single-step task, relying predominantly on synthetic trajectories or offline generated responses for training. While straightforward, such methods fundamentally suffer from a distribution mismatch between training data and actual inference trajectories. Consequently, the resulting policies tend to overfit to fixed patterns and fail to generalize sufficiently to perform spontaneous self-correction.

Applying end-to-end RL to *Self-Refinement* poses non-trivial challenges, primarily stemming from the sparsity of high-value behaviors. Specifically, instances where the model successfully self-corrects, termed *Failure-to-Success* (*F2S*), are exceptionally rare under naive sampling. Instead,

positive rollouts are predominantly composed of *Success-to-Success* (*S2S*) instances, where the model generates a correct initial response and merely maintains it. While *S2S* trajectories provide positive reward signals, they lack explicit error-correction actions, rendering them ineffective for optimizing the model's refinement capabilities.

To address these challenges, we propose **FlipTS** (**F**ailure-to-Success **L**earning for self-ref**I**nement with **P**otential-guided **T**rajectory-aware **S**ampling), a novel framework that applies end-to-end RL to *Self-Refinement*. Our approach generates and optimizes entire multi-turn self-refinement trajectories online, ensuring fully on-policy optimization and effectively eliminating the distribution shift.

Furthermore, we tackle the sparsity issue with a potential-guided trajectory-aware sampling strategy. We model the occurrence of successful correction as a Bernoulli distribution conditioned on the specific query and model state. Leveraging the Beta-Bernoulli conjugate prior for Bayesian estimation, we employ Thompson Sampling to actively prioritize queries with high refinement potential. **FlipTS** achieves superior performance while adhering to simple outcome-based rewards, effectively circumventing the need for manual reward engineering and mitigating the risk of reward hacking. Our method and subsequent analysis underscore the pivotal role of rollout distribution in RL training: sufficient exposure to high-value trajectories serves as both a prerequisite and a powerful catalyst for the emergence of advanced behaviors.

To evaluate the efficacy of **FlipTS**, we conduct extensive training and evaluate models across challenging benchmarks: AIME25, HMMT (Feb 2025), BRUMO25, and CMIMC25 (Balunović et al., 2025). Experimental results demonstrate that **FlipTS** consistently outperforms competitive baselines, including step-level RL trained on offline data and trajectory-level RL with dynamic filtering. Our work offers three key contributions:

1. We present an end-to-end RL framework for multi-turn *Self-Refinement*, enabling models to evolve intrinsic correction capabilities with complete on-policy trajectories rather than offline curated data.

2. We identify the efficiency bottleneck arising from the sparsity of high-value behaviors and propose a potential-guided Thompson Sampling strategy to address it.

3. We demonstrate that our method enhances performance on challenging mathematical benchmarks, while significantly improving out-of-distribution generalization in both scientific reasoning and trust-and-safety evaluations.

2 Related Work

Self-Correction and Self-Refinement in LLMs. LLMs can boost their reasoning skills simply by critiquing and refining their own outputs, no extra training required. Early approaches primarily focused on inference-time prompting strategies. (Madaan et al., 2023) demonstrated that LLMs can improve their answers through iterative feedback loops without updating model parameters. Moreover, (Wang et al., 2024; Shridhar et al., 2024) further revealed that *Self-Refinement* can serve as an intrinsic mechanism for continuous reasoning enhancement and model self-evolution. However, studies like (Huang et al., 2023; Liu et al., 2025) suggest that LLMs often struggle to self-correct intrinsic reasoning errors without specific training, particularly in mathematical reasoning.

Online vs. Offline RL Learning for Reasoning. Existing training-based *Self-Refinement* methods are predominantly *offline*: they rely on static datasets synthesized before training and cannot adapt to the model's evolving error patterns. One line of work uses SFT on teacher-generated refinement trajectories, e.g., (Hu et al., 2025; Zhang et al., 2024) trains on math problems with refined answers produced by a stronger model. Another employs offline RL, optimizing aggregation or selection policies over pre-collected multi-solution traces, as in (Lee et al., 2024; Zeng et al., 2025). Consequently, these methods suffer from distribution shift and tend to imitate fixed, off-policy behaviors rather than correcting their own on-policy errors. In contrast, our fully online RL framework updates exclusively from on-policy trajectories, closing the *Self-Refinement* learning loop and enabling the model to initiate corrections autonomously.

Data Efficiency and Trajectory Selection. A key challenge in online RL for reasoning is the scarcity of informative feedback signals. Existing methods struggle to adapt to the evolving policy: Kumar et al. (2024) attempts to mitigate sparsity via static, offline pre-screening, which inevitably leads to a growing data-policy mismatch due to distributional shifts as training progresses. Conversely,

while methods like DAPO (Yu et al., 2025) perform online trajectory filtering, they require prohibitively expensive large-scale rollouts. To efficiently navigate this trade-off between stability and cost, we introduce an online Bayesian mechanism, inspired by Qu et al. (2025), that dynamically estimates the informativeness of trajectories. By applying Thompson Sampling, our approach actively prioritizes *Failure-to-Success* trajectories, as these are most likely to yield a high learning signal. This maximizes information gain and sample efficiency without incurring significant inference overhead.

3 Methodology

3.1 End-to-End RL for *Self-Refinement*

We treat the iterative *Self-Refinement* process as a multi-turn generation task. For a given question q, the process initiates by generating an initial draft solution. In subsequent turns, the model conditions on the original question and the previous solution to produce a refined response. This cycle repeats for a fixed number of turns T, yielding a trajectory of reasoning steps where each iteration aims to improve the correctness of the preceding output.

MDP Formulation. We formalize this process as a Markov Decision Process (MDP). A complete episode is represented as a trajectory τ containing a sequence of states, actions, and immediate rewards:

$$\tau = \big((s_1, a_1, r_1), (s_2, a_2, r_2), \ldots, (s_T, a_T, r_T)\big).$$

Here, the state and action spaces are defined dynamically based on the turn t. At the initial turn ($t = 1$), the state is the question itself, $s_1 = q$. For refinement turns ($t > 1$), the state encapsulates the context of the previous attempt, defined as $s_t = (q, a_{t-1})$. The action a_t corresponds to the generated solution at step t. Crucially, r_t denotes the immediate reward received after action a_t, which evaluates the quality of the current step.

RLVR with Trajectory-level GRPO. Following the spirit of Reinforcement Learning with Verifiable Rewards (RLVR), we define the immediate reward r_t based on the objective correctness of the solution. Formally,

$$r_t = \begin{cases} 1, & \text{if } a_t \text{ is correct,} \\ -1, & \text{otherwise.} \end{cases}$$

Specifically, we extract the answer within `\boxed` from a_t and evaluate it against the ground truth using the `math_verify`[1] toolkit of version 0.7.0. The trajectory reward $R(\tau)$ is defined as the reward of the final output, i.e., $R(\tau) = r_T$.

We employ Group Relative Policy Optimization (GRPO) (Shao et al., 2024) for RL training. For each question q, a group of G trajectories $\tau_i = \{(s_t^{(i)}, a_t^{(i)}, r_t^{(i)})\}_{t=1}^{T}$ is sampled from an old policy π_{old}. The advantage for the i-th trajectory is computed by normalizing the rewards within the group:

$$A_i = \frac{R(\tau_i) - \text{mean}(\{R(\tau_j)\}_{j=1}^{G})}{\text{std}(\{R(\tau_j)\}_{j=1}^{G})}.$$

Then the trajectory-level GRPO objective is formulated as follows, omitting the KL divergence constraint:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}_{q\sim\mathcal{D},\{\tau_i\}\sim\pi_{\text{old}}(\cdot|q)}\Big[\frac{1}{GT}\sum_{i=1}^{G}\sum_{t=1}^{T} \frac{1}{|a_t^{(i)}|}\sum_{l=1}^{|a_t^{(i)}|}\min\{\rho_{t,l}^{(i)}A_i, \text{clip}(\rho_{t,l}^{(i)}, 1\pm\epsilon)A_i\}\Big], \quad (1)$$

where ϵ is the clip ratio and $\rho_{t,l}^{(i)}$ is the importance sampling ratio for l-th token in solution $a_t^{(i)}$,

$$\rho_{t,l}^{(i)} = \frac{\pi_\theta\big(a_{t,l}^{(i)} \mid s_t^{(i)}, a_{t,<l}^{(i)}\big)}{\pi_{\text{old}}\big(a_{t,l}^{(i)} \mid s_t^{(i)}, a_{t,<l}^{(i)}\big)}.$$

3.2 Sparsity of High-value Behaviors

Applying end-to-end RL to complex processes is fundamentally limited by the intrinsic sparsity of high-value behaviors. Trajectories that exhibit desirable capabilities, such as error recovery, are rare compared to trivial solutions. Without sufficient exposure to these informative examples, the optimization algorithm lacks the gradient signals needed to reinforce targeted skills.

This bottleneck is particularly acute in the iterative *Self-Refinement* process. In this context, *Failure-to-Success* (*F2S*) instances are extremely rare under natural sampling, leaving the model with few positive examples from which to learn effective error-correction strategies. Two factors compound this sparsity. First, LLMs exhibit a strong *self-consistency bias* (Xu et al., 2024), favoring initial responses over critical revision. Second, *misaligned problem difficulty* further reduces useful

[1] https://github.com/huggingface/Math-Verify

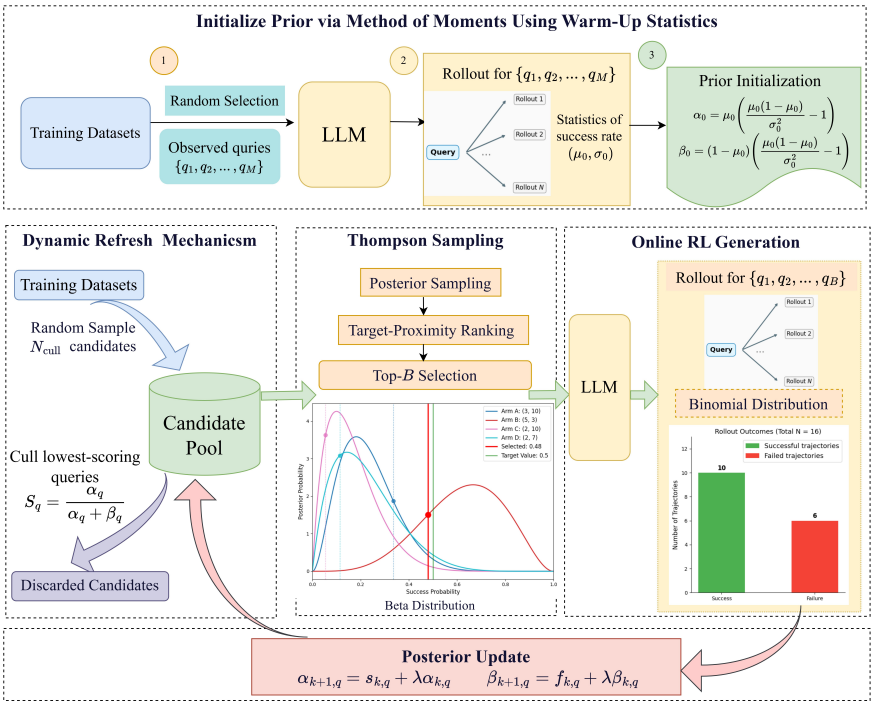

Figure 2: Framework of the Proposed Method.

signals: tasks are often too easy (solved on the first attempt) or too hard (unsolvable even with refinement). As shown in Figure 3, with a horizon of $T = 2$, valid corrections appear in only about 5% trajectories. This extreme sparsity dilutes the training signal and hampers policy optimization.

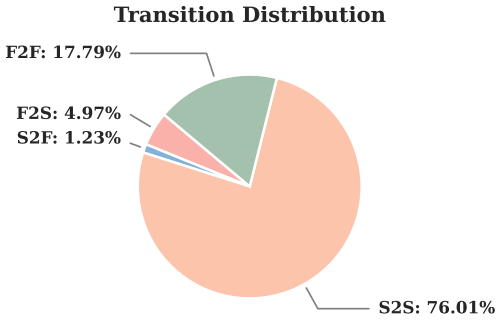

Figure 3: Proportion of four transition types during first 50 training steps of Trajectory GRPO.

3.3 Active Data Selection via Thompson Sampling

To address the sparsity of critical refinement signals, we propose an active data selection strategy that prioritizes trajectories exhibiting *F2S* transitions, where initially incorrect solutions are corrected in subsequent turns. For a horizon $T = 2$, these trajectories are identified by a reward pattern of $r_1 = -1$ followed by $r_2 = 1$. We formulate data selection as a **non-stationary multi-armed bandit (NS-MAB)** problem, in which each question q is treated as an arm. The expected reward reflects the likelihood of inducing *F2S* behavior and evolves as the model improves during training.

Assume that for each question q at training step k, the occurrence of an *F2S* transition follows a Bernoulli process governed by an unknown probability parameter $\xi_{k,q}$. To model the uncertainty of this parameter, we adopt a standard Beta-Bernoulli conjugate framework. Specifically, we maintain a belief distribution over $\xi_{k,q}$ at training step k as:

$$\xi_{k,q} \sim \text{Beta}(\alpha_{k,q}, \beta_{k,q}),$$

where $\alpha_{k,q}$ and $\beta_{k,q}$ represent the sufficient statistics (or effective pseudo-counts) for observed successes and failures, respectively.

Empirical Prior Initialization. To mitigate cold-start issues, we derive an informative global prior using data collected during a warm-up phase (the first W steps). In this phase, we select questions uniformly and compute the empirical *F2S* probability $\hat{\xi}_i$ for each observed question q_i using their rollouts. For simplicity, we model the population distribution as a Beta distribution and estimate its hyperparameters via the Method of Moments. Given the sample mean μ_0 and variance σ_0^2 of the collected $\{\hat{\xi}_i\}$, the prior parameters are derived as:

$$\alpha_0 = \mu_0 \kappa, \quad \beta_0 = (1 - \mu_0)\kappa, \tag{2}$$

where $\kappa = \mu_0(1-\mu_0)/\sigma_0^2 - 1$. We then initialize the belief state for all questions using $\alpha_{0,q} = \alpha_0$ and $\beta_{0,q} = \beta_0$.

Non-stationary Posterior Update. At training step k, we update the posterior parameters using new observations. For a question q in the current batch, let $s_{k,q}$ and $f_{k,q}$ denote the counts of trajectories with and without *F2S* transitions, respectively. The parameters are updated via

$$\begin{aligned} \alpha_{k+1,q} &= s_{k,q} + \lambda \alpha_{k,q}, \\ \beta_{k+1,q} &= f_{k,q} + \lambda \beta_{k,q}, \end{aligned} \tag{3}$$

where $\lambda \in (0, 1]$ is an exponential decay factor that discounts outdated evidence to handle non-stationarity. posterior parameters for questions not in the batch remain unchanged.

The Selection Criterion. At training step k, we construct the training batch $\mathcal{B}_k$ by leveraging the posterior to identify questions whose estimated *F2S* probabilities are closest to a target ξ_{target}. Using Thompson Sampling, we draw

$$\tilde{\xi}_{k,q} \sim \text{Beta}(\alpha_{k,q}, \beta_{k,q})$$

for each question q, and select the top-B questions minimizing the absolute deviation:

$$\mathcal{B}_k = \text{Top-B}_{q \in \mathcal{D}} \left(-\left| \tilde{\xi}_{k,q} - \xi_{\text{target}} \right| \right). \tag{4}$$

3.4 Scalable Exploration via a Dynamic Candidate Pool

Applying Thompson Sampling directly to a massive dataset $\mathcal{D}$ with $|\mathcal{D}| \gg B$ is impractical for two reasons. First, in a large candidate set, high-variance priors can produce extreme samples, causing top-B selection to be dominated by statistical outliers rather than promising trajectories. This effectively reduces exploration to near-random sampling; a theoretical analysis is provided in **Theorem 1** (Appendix B). Second, the vast search space leads to sparse and infrequent observations per question. Over such long revisit intervals, the model evolves significantly, making the posterior parameters $\alpha_{k,q}$ and $\beta_{k,q}$ outdated and misaligned with the model's current capabilities.

To address these issues, we restrict Thompson Sampling to a fixed-size, dynamically maintained candidate pool $\mathcal{C} \subset \mathcal{D}$ (where $B < M \ll |\mathcal{D}|$). The pool is managed through two key phases:

1). Initialization. Upon the completion of the warm-up phase (at step W), the pool is initialized by uniformly sampling M unique questions from the full dataset $\mathcal{D}$.

2). Dynamic Refresh and Replenishment. To prevent the pool from stagnating in local optima, we employ a saturation-based update strategy. Let $\mathcal{O} \subseteq \mathcal{C}$ be the set of *observed* questions selected at least once. A refresh is triggered when the fraction of observed questions exceeds a threshold η (i.e., $|\mathcal{O}|/M \geq \eta$). At this point, we compute the posterior mean $S_q = \alpha_q/(\alpha_q + \beta_q)$ for all observed items and remove a fraction γ of the pool corresponding to questions with the largest absolute deviation $|S_q - \xi_{\text{target}}|$. These removed items are replaced with new questions sampled from the global dataset $\mathcal{D}$, introducing exploration while filtering out questions far from the target.

We detail the complete training workflow in Appendix H.

4 Experiments

4.1 Experimental Settings

We implement **FlipTS** and various baselines leveraging the `verl-agent` framework (Feng et al., 2025). Training is conducted on **Qwen3-4B-Instruct-2507** (Team, 2025) and **Llama3.1-8B-Instruct** (Dubey et al., 2024), using a dataset composed of **DeepMath** (He et al., 2025) and **AM** (Ji et al., 2025) with difficulty levels 1 and 2.

To assess the performance on challenging mathematical reasoning, we evaluate Qwen models on **AIME25**, **HMMT (Feb 2025)**, **BRUMO25**, and **CMIMC25** (Balunović et al., 2025). For these benchmarks, we report the mean accuracy across 128 independent sampled trajectories (Acc_{avg}@128). For Llama models, we evaluate on **MATH500** (Lightman et al., 2024) using Acc_{avg}@8. To test the out-of-distribution (OOD) generalization, we also include **GPQA-Diamond** (Rein et al., 2024), **Super-GPQA** (Du et al., 2025), **TruthfulQA** (Lin et al., 2022), and **SafetyBench** (Zhang et al., 2023), reporting the Pass@1 accuracy. Please refer to Appendix C for detailed statistics of the datasets and benchmarks.

We evaluate our method against three competitive baselines: (1) **Step-level GRPO**: Employs single-turn GRPO on refinement tasks with offline-generated initial solution. (2) **Trajectory GRPO**: The end-to-end RL algorithm as described in Section 3.1 without sampling. (3) **Dynamic-filter GRPO**: A variant of Trajectory GRPO that dynamically filters out trajectories with zero advan-

Table 1: $\mathbf{Acc}_{\text{avg}}@128$ (%) of Qwen3-4B-Instruct under 10-turn *Self-Refinement* across four benchmarks.

Method	AIME25	HMMT (Feb 2025)	BRUMO25	CMIMC25	Average ↑	Rollouts ↓
w/o training	51.41	36.17	63.31	35.82	46.68	N/A
Step-level GRPO	53.65	37.50	63.41	37.66	47.99	1×
Trajectory GRPO	57.63	40.08	63.75	38.05	49.88	1×
Dynamic-filter GRPO	58.83	42.32	65.81	39.02	51.49	4×
FlipTS	**61.74**	**42.97**	**66.41**	**41.86**	**53.24**	1×

tage. Notably, while Dynamic-filter GRPO relies on costly online rollouts for sample filtering, our approach updates the candidate pool without additional rollout expenses and incurs negligible memory overhead. We refer to Appendix D for more details about these baselines.

Training uses a learning rate of $1e{-6}$, batch size 16, group size 16, max output length 8192, and trajectory length $T = 2$ in a strict on-policy setting without KL constraints. For **FlipTS**, we set $\xi_{\text{target}} = 0.5$, pool size $M = 2000$, refresh threshold $\eta = 0.3$, and culling ratio $\gamma = 0.25$. It is worth noting that the configuration was selected without exhaustive tuning. The superior performance achieved under such conditions highlights the inherent efficacy of our framework compared to the baselines. For additional sensitivity analyses, we refer to Appendix F.

4.2 Preliminary Analysis of Self-Refinement

Figure 4 shows the evaluation performance of **FlipTS** under iterative self-refinement, measured by $\mathbf{Acc}_{\text{avg}}@128$. Although trained with a short horizon ($T = 2$), evaluation accuracy consistently improves over up to 10 turns. This indicates that the training effectively instills the *atomic capability* of self-correction, which involves verifying and refining a previous solution and generalizes to longer horizons. The largest improvement occurs at the first refinement step, where many coarse initial errors are corrected in a single iteration, while subsequent steps gradually resolve finer-grained reasoning errors, resulting in substantial cumulative improvement through multi-turn refinement. In this work, we perform iterative self-refinement over 10 turns (Turn-10) for evaluation.

4.3 Comparative Results and Discussion

Table 1 summarizes the performance of different training strategies applied to Qwen3-4B-Instruct on four challenging mathematical reasoning benchmarks using 10-turn self-refinement.

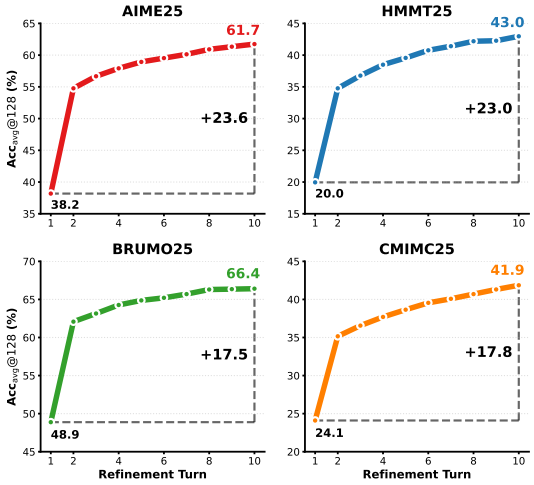

Figure 4: Acc-Avg@128 of FlipTS on AIME25, HMMT25, BRUMO25, and CMIMC25 across refinement turns. Turn 1 denotes the initial response, and subsequent turns correspond to iterative self-refinement. Across all benchmarks, performance improves monotonically with additional refinement, with the most pronounced gain appearing at the first refinement step.

Step-level GRPO provides only marginal gains over the base model, indicating limitations of offline-synthesized solutions. Trajectory GRPO improves accuracy through multi-step reasoning but is restricted by uniform sampling. Dynamic-filter GRPO further boosts performance via online filtering, but at a substantial computational cost (4× rollouts). In contrast, FlipTS achieves the highest accuracy while requiring only the baseline rollouts (1×). By selectively reinforcing high-value trajectories using trajectory-value guided sampling and Bayesian estimation, FlipTS balances exploration and exploitation efficiently, achieving superior performance without extra computational overhead.

FlipTS offers two main advantages: (1) It outperforms offline baselines, including standard Trajectory GRPO, without increasing computational cost; (2) It surpasses online filtering approaches, such as Dynamic-filter GRPO, while avoiding extra

rollouts. Overall, this demonstrates that targeted trajectory refinement is more effective than uniform sampling or discriminative filtering, achieving both higher accuracy and computational efficiency.

Improvement on LLama3.1-8B-Instruct. Table 2 highlights a fundamental shift in *Self-Refinement* dynamics, moving from stochastic degradation to principled error correction. In its vanilla state, LLama3.1-8B-Instruct lacks the metacognitive grounding to evaluate its own reasoning, causing accuracy to decline. This **refinement collapse** indicates that without internal verification, iterative turns merely act as a recursive noise generators that compound hallucinations.

Table 2: Accuracy (%) of Llama3.1-8B-Instruct on MATH500 under 10-turn *Self-Refinement*. **Init** denotes single-step inference before refinement; **Turn-10** is the final output after 10 refinement steps.

Method	Init	Turn-10
w/o training	41.95	37.68
Trajectory GRPO	50.00	52.53
FlipTS	**54.20**	**55.48**

Trajectory GRPO effectively reverses this degenerative trend by transforming the model from a passive generator into an active self-refinement. By internalizing a verification primitive, the model secures a positive refinement trajectory, demonstrating that reinforcement learning can reconfigure its policy to treat multi-turn reasoning as goal-oriented optimization. Building on this self-correction capability, FlipTS further enhances policy search by balancing exploration with guided exploitation. Its strategic trajectory sampling transforms self-refinement from an unstable heuristic into a reliable and scalable mechanism.

4.4 Which Trajectories Drive Effective *Self-Refinement*?

In this subsection, we justify the use of the *Failure-to-Success* (*F2S*) rate as the sampling criterion. Prior methods, such as DAPO (Yu et al., 2025) and MoPPS (Qu et al., 2025), typically select questions based on the *Final Success* rate, aiming to discard trivial cases with zero advantage and retain questions of moderate difficulty. However, for learning effective self-correction, we argue that moderate difficulty alone is insufficient. We compare the proposed *F2S* sampling strategy with a strategy guided by *Final Success* (*Final-S*) rate. As shown in Figure 5, the *F2S* strategy consistently outperforms the *Final-S* strategy across all four benchmarks, achieving a higher average accuracy of 53.24% compared to 51.54%. To shed light on the source of this improvement, we analyze the training dynamics in Figure 6:

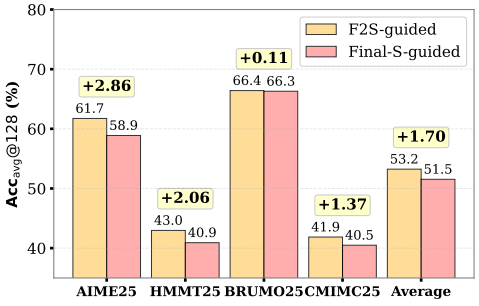

Figure 5: Comparison of *F2S* and *Final-S* sampling strategies on mathematical competition benchmarks.

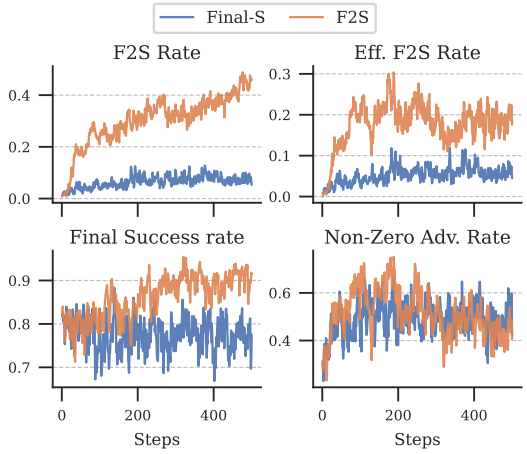

Figure 6: Training dynamics of *F2S* and *Final-S* sampling. (a) F2S Rate: proportion of *F2S* trajectories per batch. (b) Effective F2S Rate: proportion of *F2S* trajectories with non-zero advantage. (c) Final Success Rate: proportion of trajectories achieving the correct final answer. (d) Non-zero Advantage Rate: proportion of trajectories with non-zero advantage.

(1) Refinement Signals (Fig 6a & b): The *F2S*-guided strategy maintains a substantially higher proportion of *F2S* trajectories, whether measured by raw counts or when excluding zero-advantage samples. Our method effectively increases the *F2S* rate by more than three-fold, with negligible additional computational overhead. This creates a training distribution enriched with explicit correction signals, thereby enhancing the model's *Self-Refinement* capabilities.

(2) Difficulty and Efficiency (Fig 6c & d): The *Final-S* strategy favors prompts with lower final

Table 3: Accuracy (%) of Qwen3-4B-Instruct under 10-turn *Self-Refinement* on three challenging OOD benchmarks. "Init" denotes single-step inference before refinement; "Turn-10" is the final output after 10 refinement steps. All results are averaged over 128 evaluation samples per benchmark.

Method	TruthfulQA		GPQA-Diamond		SuperGPQA	
	Init	Turn-10	Init	Turn-10	Init	Turn-10
w/o training	82.03	84.43	42.42	63.13	36.80	45.20
Trajectory GRPO	80.38	84.30	53.79	65.15	**40.20**	**47.20**
FlipTS	**84.05**	**86.08**	**54.42**	**65.85**	40.10	46.20

accuracy, targeting harder problems. However, as shown in Figure 6d, both strategies result in comparable non-zero-advantage rates, suggesting that merely increasing prompt difficulty does not inherently yield more informative training signals.

These observations reveal that increasing the density of successful self-correction behaviors is more vital than simply escalating task difficulty. While reducing zero-advantage samples ensures efficiency, our method goes further by enriching the distribution with explicit refinement signals, creates a more behaviorally informative environment for *Self-Refinement*.

4.5 Out-of-Distribution Performance

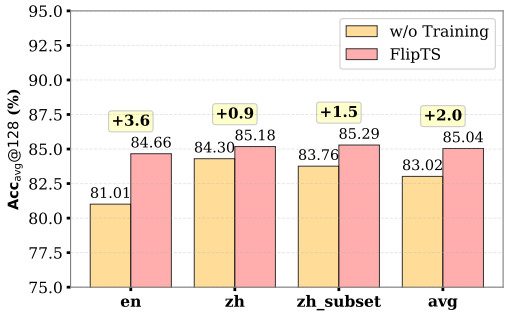

Figure 7: SafetyBench Evaluation (single-step): w/o training vs FlipTS

We evaluate the transferability of the learned *Self-Refinement* behavior on OOD benchmarks covering factuality, scientific reasoning, and safety. As shown in Table 3, **FlipTS** consistently improves the performance of Qwen3-4B-Instruct under 10-turn *Self-Refinement*, with gains observed at both the initial and final turn.

On TruthfulQA (Lin et al., 2022), **FlipTS** achieves the highest accuracy, outperforming the untrained baseline after refinement by a clear margin in factual verification. For scientific reasoning, the gains are more pronounced: on **GPQA-Diamond** (Rein et al., 2024), trajectory-level training yields over 20% absolute improvement over the untrained initialization, while on **SuperGPQA** (Du et al., 2025), it consistently improves the final refined performance, demonstrating strong generalization under OOD conditions. We further assess the impact of this training on model safety and alignment. Figure 7 reports single-step accuracy (Turn=1) on **SafetyBench** (Zhang et al., 2023), a challenging multilingual OOD benchmark covering offensiveness, bias, health, ethics, and morality. Despite training being limited to mathematical reasoning, our method improves average accuracy from 83.02% to $\mathbf{85.04}\%$, consistently enhancing safety alignment across all categories. These results show that iterative self-refinement via End-to-End RL transfers effectively to OOD tasks, enhancing the model's safety and trustworthiness.

5 Conclusion

This work introduces **FlipTS**, an end-to-end online RL framework that enhances iterative *Self-Refinement* in LLMs and addresses key limitations of offline methods. FlipTS combines Bayesian trajectory-value estimation with targeted Thompson Sampling to prioritize promising trajectories, effectively mitigating the sparsity of informative refinement behaviors. Importantly, FlipTS overcomes the issue of **refinement collapse**, where untrained models degrade over multiple self-refinement steps due to the lack of an internal verification mechanism. Through principled training and targeted trajectory sampling, FlipTS stabilizes iterative refinement, enabling the model to reliably correct its own errors and progressively improve reasoning. Results demonstrate that FlipTS achieves strong performance on challenging mathematical reasoning benchmarks and generalizes effectively to OOD tasks, highlighting its robust and transferable reasoning capabilities.

6 Limitations

Our analysis highlights the importance of rollout distribution in RL training, particularly in terms of the model's exposure to high-value trajectories. In the present work, training is restricted to a short horizon of $T = 2$, consisting of an initial response followed by a single refinement step, primarily due to limitations in available computational resources. This setting allows us to observe basic self-refinement behavior. However, how extending RL training to longer, multi-turn rollouts would influence trajectory coverage or reasoning dynamics remains a worthwhile direction for investigation. Systematic investigation of such settings may help clarify under what conditions rollout length affects the emergence and utilization of high-value trajectories.

Our current work focuses on enhancing the efficiency of sequential, iterative *Self-Refinement*. Moving forward, exploring parallel self-evolving strategies is crucial for scalability. Future research will investigate integrating our trajectory-aware sampling principles with parallel refinement techniques. In particular, we aim to develop policies that can concurrently generate and evaluate multiple candidate solutions or reasoning chains, while supporting efficient online RL training.

Furthermore, we address the bottleneck of sparse high-value trajectories through active online sample selection. While this strategy effectively prioritizes informative data, the optimization process still relies on relatively coarse trajectory-level rewards. In future work, we will integrate our sampling framework with finer-grained step-level reward shaping and dynamic advantage optimization to jointly improve online sample selection and credit assignment, thereby alleviating reward sparsity and enabling more robust self-correction in complex reasoning scenarios.

References

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. 2025. Matharena: Evaluating llms on uncontaminated math competitions. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmark*.

Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, King Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, and 1 others. 2025. Supergpqa: Scaling llm evaluation across 285 graduate disciplines. *arXiv preprint arXiv:2502.14739*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2025. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, and 1 others. 2025. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*.

Ting-Yao Hu, Hema Swetha Koppula, Hadi Pouransari, Cem Koc, Oncel Tuzel, and Raviteja Vemulapalli. 2025. Learning from self critique and refinement for faithful llm summarization. *arXiv preprint arXiv:2512.05387*.

Audrey Huang, Adam Block, Dylan J Foster, Dhruv Rohatgi, Cyril Zhang, Max Simchowitz, Jordan T Ash, and Akshay Krishnamurthy. 2024. Self-improvement in language models: The sharpening mechanism. *arXiv preprint arXiv:2412.01951*.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xanyuan Peng, and Denny. 2023. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*.

Yunjie Ji, Shuo Zhao, Xiaohao Tian, and 1 others. 2025. How difficulty-aware staged reinforcement learning enhances llms' reasoning capabilities: A preliminary experimental study. *arXiv preprint arXiv:2504.00829*.

Devvrit Khatri, Lovish Madaan, Rishabh Tiwari, Rachit Bansal, Sai Surya Duvvuri, Manzil Zaheer, Inderjit S Dhillon, David Brandfonbrener, and Rishabh Agarwal. 2025. The art of scaling reinforcement learning compute for llms. *arXiv preprint arXiv:2510.13786*.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, and 1 others. 2024. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*.

Kyungjae Lee, Dasol Hwang, Sunghyun Park, Youngsoo Jang, and Moontae Lee. 2024. Reinforcement learning from reflective feedback (rlrf): Aligning and improving llms via fine-grained self-reflection. *arXiv preprint arXiv:2403.14238*.

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

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 3214–3252.

Liping Liu, Chunhong Zhang, Likang Wu, Chuang Zhao, Zheng Hu, Ming He, and Jianping Fan. 2025. Instruct-of-reflection: Enhancing large language models iterative reflection capabilities via dynamic-meta instruction. *arXiv preprint arXiv:2503.00902*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Yun Qu, Qi Wang, Yixiu Mao, Vincent Tao Hu, Björn Ommer, and Xiangyang Ji. 2025. Can prompt difficulty be online predicted for accelerating rl finetuning of reasoning models? *arXiv preprint arXiv:2507.04632*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Kumar Shridhar, Koustuv Sinha, Andrew Cohen, Tianlu Wang, Ping Yu, Ramakanth Pasunuru, Mrinmaya Sachan, Jason Weston, and Asli Celikyilmaz. 2024. The art of llm refinement: Ask, refine, and trust. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5872–5883.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Yifei Wang, Yuyang Wu, Zeming Wei, Stefanie Jegelka, and Yisen Wang. 2024. A theoretical understanding of self-correction through in-context alignment. *Advances in Neural Information Processing Systems*, 37:89869–89912.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, and 1 others. 2025. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*.

Wenda Xu, Guanglei Zhu, Xuandong Zhao, Liangming Pan, Lei Li, and William Wang. 2024. Pride and prejudice: LLM amplifies self-bias in self-refinement. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15474–15492, Bangkok, Thailand. Association for Computational Linguistics.

Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun MA, and Bo An. Simpletir: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. In *NeurIPS 2025 Fourth Workshop on Deep Learning for Code*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Yongcheng Zeng, Xinyu Cui, Xuanfa Jin, Guoqing Liu, Zexu Sun, Dong Li, Ning Yang, Jianye Hao, Haifeng Zhang, and Jun Wang. 2025. Evolving llms' self-refinement capability via synergistic training-inference optimization. *arXiv preprint arXiv:2502.05605*.

Qingjie Zhang, Di Wang, Haoting Qian, Yiming Li, Tianwei Zhang, Minlie Huang, Ke Xu, Hewu Li, Liu Yan, and Han Qiu. 2025. Understanding the dark side of LLMs' intrinsic self-correction. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27066–27101, Vienna, Austria. Association for Computational Linguistics.

Xiaoying Zhang, Baolin Peng, Ye Tian, Jingyan Zhou, Lifeng Jin, Linfeng Song, Haitao Mi, and Helen Meng. 2024. Self-alignment for factuality: Mitigating hallucinations in llms via self-evaluation. *arXiv preprint arXiv:2402.09267*.

Zhexin Zhang, Leqi Lei, Lindong Wu, Rui Sun, Yongkang Huang, Chong Long, Xiao Liu, Xuanyu Lei, Jie Tang, and Minlie Huang. 2023. Safetybench: Evaluating the safety of large language models with multiple choice questions. *arXiv preprint arXiv:2309.07045*.

A Notations and Symbols

Table 4 summarizes the primary symbols, variables, and parameters used in the **FlipTS** framework, categorized by trajectory dynamics, RL optimization, and adaptive sampling mechanism.

Notation	Description
Problem & Trajectory Formulation	
q, y	Input query and its corresponding ground-truth answer.
τ	Multi-turn *Self-Refinement* trajectory $\{(s_t, a_t, r_t)\}_{t=1}^{T}$.
T	Maximum number of turns per episode.
s_t, a_t, r_t	State, action (solution), and verifiable reward at turn t.
RL Optimization & Dynamics	
$\pi_\theta, \mathcal{D}$	Policy network and the global training dataset.
$R(\tau), A_i$	Trajectory-level return and group-relative advantage.
G	Group size (number of sampled trajectories per query).
$\mathcal{J}_{\text{GRPO}}$	Objective function for trajectory-level GRPO.
Active Selection via Thompson Sampling	
$\xi_{k,q}$	*F2S* transition probability for question q at step k.
α, β	Shape parameters (pseudo-counts) of the Beta distribution.
λ	Decay factor addressing non-stationarity of the policy.
ξ_{target}	Target *F2S* probability for prioritizing trajectories.
$\mathcal{C}, M$	Candidate pool and its capacity ($M = \lvert\mathcal{C}\rvert$).
$\mathcal{O}$	Set of observed questions within the candidate pool.
η	Saturation threshold for triggering a pool refresh.
γ	Fraction of candidates to be culled during replenishment.
S_q	Posterior mean used for candidate filtering.

Table 4: Summary of notations and definitions.

B Theoretical Analysis of Thompson Sampling Collapse Due to Large-Scale Datasets

In this section, we provide a formal justification for the adoption of Dynamic Candidate Pool. We analyze the asymptotic behavior of Thompson Sampling when the dataset size N is much larger than the sampling batch B, and show that, despite the presence of informative samples, the selection mechanism effectively degenerates into uniform random sampling: a phenomenon we term *Thompson Sampling Collapse*.

Theorem 1 (Asymptotic Collapse of Thompson Sampling Precision). Let $\mathcal{D}$ be a dataset of cardinality N, partitioned into two disjoint sets:

- $\mathcal{O}$: The set of **Observed** samples with cardinality $|\mathcal{O}| = K \ll N$.
- $\mathcal{U}_N$: The set of **Unobserved** samples with cardinality $|\mathcal{U}_N| = N - K \approx N$.

Statement: As $N \to \infty$ with fixed batch size B, the expected number of Observed samples in the batch approaches zero, causing the batch precision to converge to the prior sparsity ρ.

Proof. **1. The Budget Constraint**
The selection of the top-B samples defines a selection window

$$\mathcal{W}_N = [x^* - \delta_N, x^* + \delta_N].$$

The total count B comprises selected observed samples (M_{obs}) and unobserved samples (M_{un}). Using the Law of Large Numbers to approximate discrete counts with probability integrals:

$$\begin{aligned} B &= \underbrace{K \int_{\mathcal{W}_N} f_{post}(x)\,dx}_{\mathbb{E}[M_{obs}]} \\ &+ \underbrace{(N-K) \int_{\mathcal{W}_N} f_{prior}(x)\,dx}_{\mathbb{E}[M_{un}]} \end{aligned} \tag{5}$$

2. Asymptotic Shrinkage of δ_N
As $N \to \infty$, the unobserved component $(N - K)$ dominates the budget equation. To keep the total sum finite (B), the integration interval δ_N must shrink to zero, i.e., $\delta_N \to 0$. By the Mean Value Theorem for small intervals, we can approximate:

$$B \approx 2\,\delta_N \Big[K \cdot f_{\text{post}}(x^*) + (N-K) \cdot f_{\text{prior}}(x^*)\Big].$$

Since $N \gg K$, the contribution from the unobserved portion dominates:

$$B \approx 2\,\delta_N (N-K)\, f_{\text{prior}}(x^*),$$

Thus, the batch density is governed almost entirely by the prior of the unobserved population.

3. Asymptotic Dilution of Informative Samples
We now determine the expected number of high-value observed samples that successfully enter the batch by substituting δ_N back into the expectation term:

$$\begin{aligned} \mathbb{E}[M_{obs}] &\approx K \cdot 2\delta_N \cdot f_{post}(x^*) \\ &\approx \tfrac{K \cdot B}{N} \cdot \underbrace{\frac{f_{post}(x^*)}{f_{prior}(x^*)}}_{\text{Density Ratio}}. \end{aligned} \tag{6}$$

The observed samples typically exhibit **Posterior Concentration** (low variance) due to evidence updates, whereas unobserved samples follow a diffuse

prior. This implies that the local density at the target is topically higher for observed samples:

$$f_{post}(x^*) \geq f_{prior}(x^*) > 0.$$

The density ratio remains a finite constant. As the dataset size scales ($N \to \infty$), the linear growth of N in the denominator overwhelms the density benefit:

$$\lim_{N \to \infty} \mathbb{E}[M_{obs}] = 0. \tag{7}$$

Therefore, the batch becomes statistically saturated with unobserved samples ($\mathcal{B} \approx \mathcal{B}_{un} \subset \mathcal{U}_N$). Since the unobserved set is a random subset inheriting the global signal sparsity, the final batch precision reverts to ρ. □

Remark 1. This theoretical insight directly motivates our sampling design. In practice, datasets are large ($N \sim 10^5$–10^7), while batch sizes are small ($B \sim 2^4$–2^7). Under this regime ($N \gg B$), Theorem 1 shows that standard Thompson Sampling suffers from **asymptotic dilution of informative samples**, effectively degenerating into uniform random sampling and failing to leverage newly discovered high-value trajectories. To mitigate this, we introduce a **Candidate Pool** of manageable size (e.g., $M = 2000$), ensuring the ratio B/M remains sufficiently large. This preserves the concentration of the posterior density $f_{\text{post}}(x^*)$ around informative samples, thereby maintaining the efficacy of Thompson Sampling for active exploitation.

C Dataset Details and Statistics

In this section, we provide detailed documentation of the datasets used for training, validation, and benchmarking. The primary domains cover competition-level mathematics, scientific reasoning, and AI safety/trustworthiness.

C.1 Training Dataset

Our training corpus is constructed by uniformly mixing the following mathematical reasoning datasets: (1) **DeepMath** (He et al., 2025): Contains 103k samples; (2) **AM-Math-Difficulty-RL** (Ji et al., 2025): We utilize the subset of diffuculty level-1 (98k) and level-2 (101k). The total size of the mixed training dataset is approximately 302k samples.

C.2 Evaluation Benchmarks

We employ a diverse set of benchmarks to evaluate mathematical capability and out-of-distribution (OOD) generalization. Table 5 summarizes the statistics, domains, and evaluation settings for each dataset. Note that for TruthfulQA, we adhere to the new, improved multiple-choice version[2] recommended by the authors. For SuperGPQA, we use a fixed random subset of 1,000 questions for evaluation.

D Baseline Details

We detail the baselines and summarize their key differences and characteristics in Table 6.

Step-level GRPO. This baseline treats solution generation and refinement as decoupled single-turn tasks. We randomly sample 20,000 questions and use the original model to generate one initial solution for each. The training is conducted via single-turn GRPO on a dataset comprising two tasks: (1) generating a correct solution directly from the question, and (2) generating a refined solution given the pre-generated initial response as context. This baseline serves to isolate the benefits of end-to-end online optimization.

Trajectory GRPO. This baseline implements the end-to-end RL framework detailed in Section 3.1, optimizing self-refinement with complete multi-turn trajectory. Unlike our proposed method, it omits the Thompson Sampling mechanism and instead adheres to the original dataset distribution for rollout generation.

Dynamic-filter GRPO. This baseline integrates the dynamic filtering spirit of DAPO into the Trajectory GRPO framework. Instead of following the original DAPO formulation strictly, it utilizes online rejection sampling to filter out trajectories with non-positive advantages. This ensures a fair comparison by applying DAPO's adaptive sampling logic to the same trajectory-level search space as our method. However, it incurs significant compute overhead from oversampling and rejection.

E Computation Budget

Our experimental framework and algorithms are implemented as modifications to the open-source `verl-agent` library, utilizing RL training for model optimization. All experiments were conducted on a dedicated compute node equipped with 8× NVIDIA GPUs (80GB VRAM), operating with

[2] https://www.alignmentforum.org/posts/Bunfwz6JsNd44kgLT/new-improved-multiple-choice-truthfulqa

Table 5: Statistics of evaluation datasets.

Category	Dataset	Size (Questions)	Evaluation Setting
Validation	AIME24[3] AIME25, HMMT (Feb 2025) (Balunović et al., 2025)	30 each	Acc_{avg}@8
Math Benchmark	AIME25, HMMT (Feb 2025), BRUMO25, CMIMC25 (Balunović et al., 2025)	30 each	Acc_{avg}@128
	MATH500 (Lightman et al., 2024)	500	Acc_{avg}@8
Science	GPQA-Diamond (Rein et al., 2024)	198	Pass@1
	SuperGPQA (Subset) (Du et al., 2025)	1,000	Pass@1
Safety & Trust	SafetyBench (Zhang et al., 2023)	11,435	Pass@1
	TruthfulQA (Lin et al., 2022)	817	Pass@1

[3] https://huggingface.co/datasets/AI-MO/aimo-validation-aime

Table 6: Comparison of baseline methodologies across core algorithmic dimensions. E2E RL: End-to-End RL optimization of self-refinement; Sample Pruning: Online filtering of zero-advantage trajectories.

Method	E2E RL Paradigm	Thompson Sampling	Sample Pruning	Compute Efficiency	Gradient Fidelity
Step-level GRPO	×	×	×	High	Moderate
Trajectory GRPO	✓	×	×	High	Moderate
Dynamic-filter GRPO	✓	×	✓	Low[†]	High
FlipTS (Ours)	✓	✓	×	High	High

[†] Dynamic-filter GRPO increases the rollout-to-update ratio through extra online sampling and rejection.

CUDA 12.2. The computational cost for training to approximately 800 update steps is as follows:

- **Qwen3-4B-Instruct:** Completing 800 update steps requires approximately 3.6 days of continuous wall-clock time, largely due to the iterative generation and self-refinement cycles.
- **Llama-3.1-8B-Instruct:** Under identical configurations, the training process for 800 steps requires approximately 2.5 days.

F Sensitivity Analysis of Hyperparameters

In this section, we conduct several experiments to investigate the impact of key hyperparameters and design choices within our dynamic candidate pool framework. To provide a comprehensive view of the training dynamics, we visualize the validation scores throughout the optimization process. The validation set comprises three challenging benchmarks: AIME24, AIME25, and HMMT (Feb 2025), evaluated under a self-refinement process with a horizon of $T = 2$. We report performance using the Average Accuracy over 8 sampled trajectories (denoted as Acc_{avg}@8).

F.1 Target *Failure-to-Success* Probability

We investigate the impact of the target *Failure-to-Success* probability ξ_{target}. We evaluate the model's performance with $\xi_{\text{target}} \in \{0.2, 0.5, 0.8\}$, and the corresponding validation scores are presented in Figure 8. The setting of $\xi_{\text{target}} = 0.8$ consistently yields superior performance compared to lower values of 0.5 and 0.2. This empirical evidence validates the hypothesis that exposing the model to a higher density of Failure-to-Success transitions is essential for mastering *Self-Refinement*. Thus, prioritizing these trajectories through the sampling mechanism serves as a crucial driver for performance improvements.

Figure 8: Validation scores for target Failure-to-Success probability $\xi_{\text{target}} \in \{0.2, 0.5, 0.8\}$ over training steps.

F.2 Size of Candidate Pool M

To analyze the sensitivity of our method to the pool size M, we compare the validation scores of $M \in \{2000, 20000, 100000\}$ in Figure 9. The

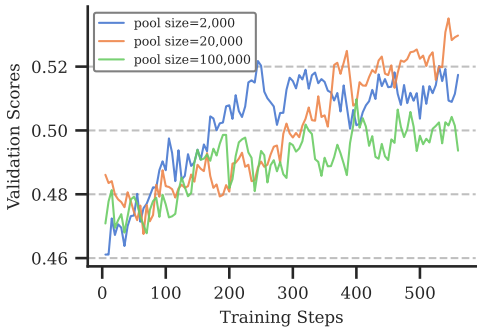

Figure 9: Validation scores for pool sizes $M = 2,000$ and $M = 20,000$ over training steps.

results indicate a trade-off between convergence speed and final performance. A smaller pool ($M = 2,000$) accelerates early-stage training by encouraging rapid exploitation, whereas a moderately larger pool ($M = 20,000$) exhibits a slower initial learning curve but fosters more thorough exploration, ultimately yielding superior asymptotic performance. This suggests that M can be adjusted to balance computational efficiency against peak accuracy depending on the available budget. Notably, an excessively large pool ($M = 100,000$) leads to significant performance degradation. This observation empirically validates the necessity of the candidate pool mechanism itself, aligning with our discussion in Section 3.4 and corroborate the theoretical proofs provided in Appendix B.

G Artifact Licenses and Usage Compliance

This work strictly complies with the licenses and terms of service for all utilized artifacts. Our usage is confined to academic, non-commercial research.

- Permissive Licenses (MIT, Apache-2.0, CC-BY-4.0, ODC-By): We adhere to the attribution and open-source terms for DeepMath, TruthfulQA, SafetyBench, GPQA-Diamond, SuperGPQA, and AIME24. Our codebase utilizes verl-agent and `math_verify` under the Apache-2.0 license.

- Non-Commercial Licenses (CC-BY-NC-4.0, CC-BY-NC-SA-4.0): We utilize AM-Math-Difficulty-RL, AIME25, HMMT (Feb 2025), BRUMO25, and CMIMC25 strictly for non-commercial research purposes, ensuring no violation of the "NC" (Non-Commercial) or "SA" (Share-Alike) clauses.

Attribution is provided for all cited assets to maintain transparency and reproducibility in line with community standards.

Furthermore, we confirm that none of the utilized datasets contain personally identifiable information (PII). While SafetyBench inherently contains sensitive or potentially offensive queries by design, such content serves a specific research purpose and does not target real-world individuals. The remaining mathematical datasets are free of offensive content.

H Algorithm

Algorithm 1 summarizes our proposed method, **FlipTS**, which integrates dynamic Bayesian posterior estimation with targeted Thompson sampling to guide end-to-end RL for *Self-Refinement*. It maintains a candidate pool of questions and continuously updates belief parameters based on observed trajectory outcomes. By sampling trajectories whose estimated refinement potential is closest to a target probability ξ_{target}, **FlipTS** focuses training on the model's zone of proximal development. A refresh-and-cull mechanism ensures pool diversity and mitigates posterior staleness under non-stationary learning dynamics.

I Self-Refinement Prompt Template

Our self-refinement process is structured as a multi-turn dialogue. The specific prompt configurations for each stage of the process are detailed below.

System Message (All Turns) For every turn in the conversation (both initial generation and subsequent refinements), the system message remains constant to enforce step-by-step reasoning and correct output formatting:

> **System:**
> `Please reason step by step, and put your final answer within \boxed{}.`

Turn 1: Initial Generation In the first turn, the user message consists solely of the math problem. The model generates the initial candidate solution based on this input.

> **User:**
> `{question}`

Subsequent Turns: Refinement For all subsequent turns, the user message follows a structured

refinement template. This template feeds the problem and the previous candidate solution (which may be flawed) back into the model, instructing it to critique and improve the answer.

User:

```
You are given a math problem and
a potential solution.
This attempt may be flawed or
partial.
Correct any errors and build upon
its ideas to produce an improved,
high-quality solution.
If the original attempt is
unusable, generate a correct
solution from scratch.
End with the final result in
\boxed{}.

Problem:
{question}

Candidate solution (may contain
mistakes):
-- Candidate --
{candidate_answer}

Now refine the candidate into an
improved solution.
Provide clear reasoning and
end with the final answer in
\boxed{}.
```

Here, `{question}` is the original problem statement, and `{candidate_answer}` is the model's output from the immediate previous turn.

Algorithm 1: FlipTS

Input: Dataset $\mathcal{D}$; Batch size B; Decay factor λ; Target probability ξ_{target}; Warm-up steps W; Pool size M; Refresh threshold η; Culling fraction γ; Initial Policy π_{θ_0}.

Output: Optimized Policy π_{θ_N}

for $k = 1, 2, \ldots, N$ **do**

  `// Thompson Sampling for Questions`

  if $k \leq W$ **then**

    Sample batch $\mathcal{B}_k$ uniformly from $\mathcal{D}$;

  end

  else

    Sample a proxy probability $\tilde{\xi}_{k,q} \sim \text{Beta}(\alpha_{k,q}, \beta_{k,q})$ for each $q \in \mathcal{C}$;

    Select batch $\mathcal{B}_k$ as the top-B questions in $\mathcal{C}$ minimizing $|\tilde{\xi}_{k,q} - \xi_{target}|$ (Eq. 4);

  end

  `// RL Training (Trajectory-level GRPO)`

  Generate trajectories $\{\tau_i\}_{i=1}^{B \times G}$ using π_{θ_k} for questions in $\mathcal{B}_k$;

  Update θ_k to θ_{k+1} by maximizing $\mathcal{J}_{GRPO}(\theta)$ (Eq. 1);

  `// Beyasian Estimation`

  Count the number of trajectories with and without *F2S* transition s_q, f_q; compute $\hat{\xi}_q = s_q/G$;

  if $k = W$ **then**

    Compute prior parameters α_0, β_0 using the empirical probabilities $\{\hat{\xi}_q\}$ from the first W steps (Eq. 2); initialize $\alpha_{k,q} \leftarrow \alpha_0, \beta_{k,q} \leftarrow \beta_0, \forall q \in \mathcal{D}$;

  end

  if $k > W$ **then**

    Update $\alpha_{k+1,q} \leftarrow s_q + \lambda \alpha_{k,q}$, $\beta_{k+1,q} \leftarrow f_q + \lambda \beta_{k,q}$ for each $q \in \mathcal{B}_k$ (Eq. 3);

  end

  `// Dynamic Pool Mechanism`

  if $k = W$ **then**

    Initialize candidate pool $\mathcal{C}$ by uniformly sampling M questions from $\mathcal{D}$;

    Initialize observed questions in pool $\mathcal{O} = \emptyset$;

  end

  if $k > W$ **then**

    Update observed questions: $\mathcal{O} \leftarrow \mathcal{O} \cup \mathcal{B}_k$;

    if $|\mathcal{O}|/M \geq \eta$ **then**

      Calculate scores $S_q = \alpha_{k,q}/(\alpha_{k,q} + \beta_{k,q})$ for $q \in \mathcal{O}$;

      Identify cull set $\mathcal{Q}_{rem} \subset \mathcal{O}$ with $\lfloor \gamma M \rfloor$ questions with largest $|S_q - \xi_{\text{target}}|$;

      Uniformly sample a new set $\mathcal{Q}_{new}$ from $\mathcal{D} \setminus \mathcal{C}$ with $|\mathcal{Q}_{rem}| = |\mathcal{Q}_{new}|$;

      Update pool $\mathcal{C} \leftarrow (\mathcal{C} \setminus \mathcal{Q}_{rem}) \cup \mathcal{Q}_{new}$; $\mathcal{O} \leftarrow \mathcal{O} \setminus \mathcal{Q}_{rem}$

    end

  end

end

return π_{θ_N}