Difficulty-Adaptive Tree-Structured Policy Optimization for Expanding Reasoning Coverage in RLVR

Youngjun Yu **Sanghwan Jang** **Hwanjo Yu***
Pohang University of Science and Technology (POSTECH)
{colin31472, s.jang, hwanjoyu}@postech.ac.kr

Abstract

Reinforcement Learning with Verifiable Rewards (RLVR) has been central to the recent success of Large Reasoning Models. However, while RLVR significantly improves single-sample accuracy, it often fails to expand the model's intrinsic reasoning coverage (pass@k) due to limited exploration during training. To address this, we optimize the structural design of train-time rollouts to enhance pass@k. Our analysis identifies three key design principles: (1) difficulty-adaptive rollout can play an important role in expanding pass@k, beyond serving as an efficiency heuristic; (2) tree-based rollout outperforms parallel sampling in discovering correct answers; and (3) sentence-entropy-guided forking overcomes the *localization* phenomenon of token-level branching to maximize semantic diversity. Building on these insights, we propose **DATPO** (**D**ifficulty-**A**daptive Sentence-entropy-guided **T**ree-structured **P**olicy **O**ptimization). DATPO integrates difficulty-adaptive tree search with a *sibling-diversity* advantage term, explicitly promoting semantic diversity to expand reasoning coverage during training. Experiments on mathematical reasoning benchmarks demonstrate that DATPO outperforms baselines especially in pass@k, which directly translates to superior test-time scaling performance.[1]

1 Introduction

The paradigm of Large Reasoning Models (LRMs) has achieved remarkable success in eliciting rigorous problem-solving capabilities from foundation models (Jaech et al., 2024; Team et al., 2025). A key mechanism behind this is Reinforcement Learning with Verifiable Rewards (RLVR), exemplified by DeepSeek-R1 (Guo et al., 2025) through its successful application of the Group Relative Policy Optimization (GRPO) (Shao et al., 2024).

In parallel, LRM research has increasingly emphasized test-time scaling strategies, including CoT (Wei et al., 2022), majority voting (Wang et al., 2022), best-of-N (Stiennon et al., 2020), and Monte Carlo Tree Search (MCTS) (Ha et al., 2025). Fundamentally, the effectiveness of these methods is limited by the model's intrinsic reasoning coverage—the breadth of valid reasoning paths the model can explore, typically measured by pass@k. Without sufficient reasoning coverage, even large candidate sets are unlikely to include a correct reasoning path, causing test-time scaling to aggregate or select among recurring errors (Brown et al., 2024; Zhao et al., 2025).

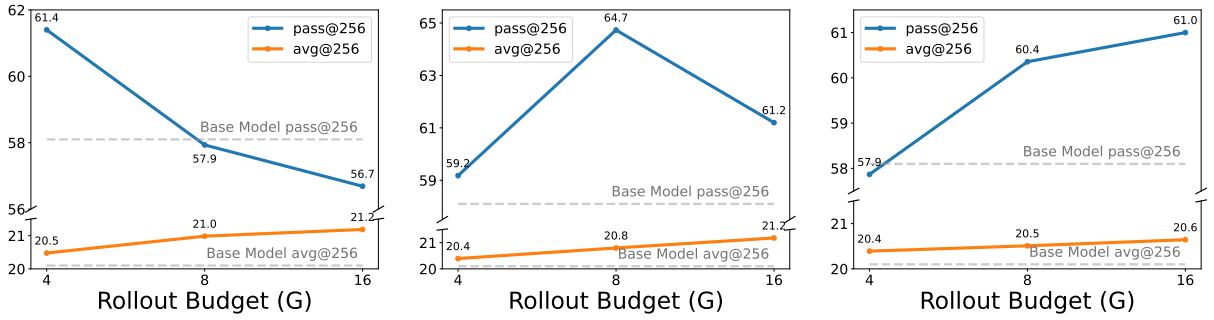
However, recent studies (Yue et al., 2025; Dang et al., 2025; Wu et al., 2025) argue that RLVR often fails to unlock new reasoning capabilities beyond the reasoning coverage of the base model. This limitation stems from a lack of explicit exploration strategies that steer the learning or sampling process toward underexplored regions. Without such guidance, the model merely exploits known solutions rather than discovering novel ones.

While recent studies introduce exploration strategies to expand reasoning coverage (Walder and Karkhanis, 2025; Yao et al., 2025; Wang et al., 2025), they largely focus on optimization-level interventions, leaving the train-time rollout as a fixed parallel structure. Tree-based frameworks (Hou et al., 2025; Liu et al., 2025a) have begun to move beyond standard parallel sampling. However, these methods primarily utilize tree structures for credit assignment or computational efficiency. Consequently, it remains underexplored which structural choices in train-time rollouts actually expand the model's reasoning coverage.

To bridge this gap, we conduct a systematic empirical analysis of train-time rollouts, revealing three core design principles to maximize reasoning coverage, extending beyond improvements in single-sample accuracy. First, **difficulty-adaptive**

*Corresponding author.
[1]Code: https://github.com/colin31472/DATPO

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 24084–24110
October 24-29, 2026 ©2026 Association for Computational Linguistics



(a) Models trained on **Easy** dataset (b) Models trained on **Medium** dataset (c) Models trained on **Hard** dataset

Figure 1: avg@256 and pass@256 performance across different rollout budgets for models trained on different difficulty subsets.

rollout is an important factor in expanding pass@k, rather than merely an efficiency heuristic. While increasing the rollout budget improves pass@k when training on hard problems, we observe that it can actually degrade pass@k when training on easy problems. Second, **tree-based rollout outperforms parallel sampling** in discovering correct answers by efficiently utilizing limited computational budgets. Finally, we observe that conventional token-guided forking suffers from *localization*, where high-entropy tokens cluster in narrow segments and trap exploration. To overcome this, **forking decisions must be elevated to a broader semantic level** to discover genuinely diverse reasoning paths.

Building on these insights, we propose **DATPO** (**D**ifficulty-**A**daptive Sentence-entropy-guided **T**ree-structured **P**olicy **O**ptimization). For train-time exploration, DATPO combines the structural advantages of tree-based search with difficulty-adaptive rollout to expand reasoning coverage. Forking points are selected using sentence-level entropy signals, broadening the granularity of uncertainty estimation to avoid *localization*. Furthermore, to maximize coverage-expanding benefits, we augment the advantage function with an annealed *sibling-diversity* term. This explicitly rewards semantically diverse reasoning branches, encouraging the model to explore a wider range of valid reasoning paths.

Extensive experiments on mathematical reasoning benchmarks demonstrate the effectiveness of DATPO, which achieves the highest average pass@k among the compared methods. Furthermore, this expanded reasoning coverage directly translates to superior test-time scaling with majority voting (maj@k).

In summary, our main contributions are as follows:

- We reveal key structural principles for expanding reasoning coverage: difficulty-adaptive rollout is a key factor, and tree-structured rollouts inherently outperform parallel sampling, with sentence-entropy forking further amplifying this structural advantage.
- We propose DATPO, a novel RLVR framework that integrates these structural insights with a diversity-augmented advantage to maximize coverage-expanding benefits. Experiments on mathematical reasoning benchmarks validate our approach, showing notable improvements in pass@k.

2 Analyzing the Structural Design of Train-time Rollouts

Despite extensive research on expanding reasoning coverage, the structural design of *train-time* rollouts remains underexplored. This section investigates optimal rollout structures to maximize not only single-sample accuracy (avg@k) but also broader reasoning coverage (pass@k). Our analysis is driven by three primary research questions: (1) **Difficulty-Adaptive Allocation:** How should compute budgets be allocated across problems of varying difficulty to expand reasoning coverage?; (2) **Topology of Search Space:** Which search space structure (tree-based or parallel rollout) more efficiently discovers correct reasoning trajectories?; and (3) **Optimal Forking Strategy:** Where should forking (branching) occur in tree-based rollouts to effectively promote diverse reasoning paths?

2.1 Necessity of Difficulty-Adaptive Rollout

Standard RLVR algorithms, notably GRPO (Shao et al., 2024) and its variants (Yu et al., 2025b; Liu et al., 2025b), typically generate a uniform number of rollouts regardless of problem difficulty. While existing difficulty-adaptive rollout strategies prioritize *computational efficiency* (Liu et al., 2025a; Liao et al., 2025; Zhang et al., 2025; Zheng et al., 2025a; Li et al., 2025b), we challenge this perspective by demonstrating that adaptive allocation is not merely a resource-saving heuristic but a crucial factor in expanding reasoning coverage.

2.1.1 Empirical Analysis

We first partitioned the training dataset into Easy, Medium, and Hard subsets based on the accuracy of the base model. Using GRPO (Shao et al., 2024), we conduct a total of 9 training runs. For each difficulty subset, we vary the rollout budget by adjusting the GRPO group size, $G \in \{4, 8, 16\}$. The resulting 9 models are evaluated on 3 different benchmarks, reporting both avg@256 and pass@256 metrics (detailed in Appendix B).

The aggregated results, as illustrated in Figure 1, reveal two key insights. First, **avg@256 shows consistent, yet marginal, improvements with increased rollout budgets (G) across all difficulties.** We attribute this to more stable exploitation; larger GRPO group sizes reduce variance in advantage estimation and yield more frequent meaningful learning signals, effectively solidifying the policy distribution around correct solutions.

Second, **the impact of increasing rollouts on pass@256 shifts from detrimental to beneficial depending on the difficulty of the training dataset.** For models trained on the *Easy* dataset (Figure 1a), larger budgets degrade pass@256. This occurs because the model over-exploits specific, easy templates, overfitting to familiar problems while failing entirely on harder ones. Since pass@256 requires only a single correct path, generating redundant successes for easy problems provides no coverage benefit. Consequently, despite marginal gains in avg@256, pass@256 decreases as the rollout budget increases. Conversely, for models trained on the *Hard* dataset (Figure 1c), where valid paths are scarce, larger budgets promote broader exploration without overfitting to narrow patterns, consistently improving pass@256. Between these opposing behaviors, models trained on the *Medium* dataset (Figure 1b) exhibit a non-monotonic trend, peaking at $G = 8$ by effectively balancing exploration and over-exploitation.

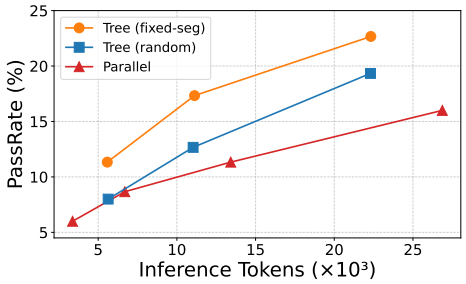


Figure 2: Comparison of PassRate against inference token consumption between parallel and two tree-structured sampling using different forking strategies.

These findings suggest that difficulty-adaptive rollout is not merely a heuristic for compute efficiency, but a strategic necessity for balancing the exploitation-exploration trade-off and maximizing the model's reasoning coverage. Comprehensive theoretical analysis is provided in Appendix A.

2.2 Tree vs. Parallel Rollout

In RLVR, the primary challenge in expanding reasoning coverage is the scarcity of positive learning signals on complex problems. These signals emerge only when the model discovers a correct reasoning path. To address this, we investigate the properties of train-time rollout structures by comparing the conventional *parallel rollout* with a *tree-structured rollout* strategy, aiming to determine which topology more efficiently discovers correct trajectories.

2.2.1 Tree Rollout Algorithm

We employ a generalized and simplified variant of the two-phase tree rollout strategy originally proposed by Hou et al. (2025). In the first phase, the model generates N independent base rollouts in parallel. Subsequently, in the second phase, we apply a forking point selection algorithm to identify K forking points within each base trajectory. From each selected forking point, we generate B additional branch rollouts, thereby expanding the exploration scope. Detailed algorithmic procedures are provided in Appendix C.1.

2.2.2 Empirical Analysis

To analyze cost-efficiency during inference, we compare standard parallel sampling against two tree-structured rollout variants by analyzing PassRate—the probability of obtaining at least one correct solution—as a function of token consumption during generation. These variants include

`fixed-seg`, which selects the K forking points at equidistant intervals, and `random`, which selects them uniformly at random (detailed in Appendix C.2).

As shown in Figure 2, both tree-structured variants achieve higher token efficiency than parallel sampling. Specifically, their performance curves demonstrate a steeper growth rate, yielding a higher PassRate for the same number of generated tokens. This enhanced efficiency stems from prefix sharing in tree structures, which avoids redundant regeneration of identical early segments and enables the exploration of alternative continuations within the same token budget (Tran et al., 2025; Hou et al., 2025).

Furthermore, the choice of forking strategy also affects performance within tree-structured methods. While both `fixed-seg` and `random` outperform parallel sampling, `fixed-seg` consistently achieves a higher PassRate under comparable token budgets. This demonstrates that the effectiveness of tree-based exploration depends not only on its structure but also on the placement of branching points.

2.3 Forking Point Selection in Tree Search

Section 2.2 shows that the performance of tree-structured rollouts varies substantially across different forking strategies. In this section, we investigate which forking strategies most effectively identify correct solutions and promote semantically diverse reasoning paths.

2.3.1 The Localization Phenomenon and Sentence-level Forking

Previous studies primarily select top-k high-entropy tokens as forking points (Hou et al., 2025; Zheng et al., 2025b; Cao et al., 2026), as they capture highly uncertain words that act as pivotal branching points (Wang et al., 2025; Cheng et al., 2025). However, we identify a critical limitation in this approach: the *localization* phenomenon (Figure 8). High-entropy tokens tend to densely cluster within narrow, highly uncertain segments of the reasoning trajectory. Consequently, the search budget is monopolized by repeated resampling within a localized cluster, which limits the structural reach of the search tree.

To mitigate localization, we propose a sentence-level forking strategy (`sent-entropy`), which expands the granularity of entropy estimation. In this approach, sentence entropy is calculated as the average entropy of its tokens, and the starting points of the top-k high-entropy sentences are selected as forking points. This strategy naturally evades localization while still targeting the most uncertain regions. We compare this approach with the conventional token-level method (`tok-entropy`) and three baselines: `random`, fixed-segment selection (`fixed-seg`), and Attention-based Tree Branching (`ATB`) (Liu et al., 2025a), which branches at steps receiving the highest attention weights. Detailed algorithmic procedures are provided in Appendix D.1.

Method	PassRate	SibDiv
`random`	10.0	0.0796
`fixed-seg`	13.3	0.0853
`ATB`	<u>14.7</u>	0.0874
`tok-entropy`	12.0	**0.1065**
`sent-entropy`	**17.3**	<u>0.1049</u>

Table 1: Comparison of **forking point selection strategies** when applied during inference. Best results are in **bold**, and second-best results are <u>underlined</u>.

2.3.2 Metrics for Measuring Diversity

While PassRate effectively captures task success, comprehensively evaluating these forking strategies also requires measuring whether the generated paths are semantically diverse. To directly quantify exploration diversity, we introduce an embedding-based metric, Sibling Diversity (SibDiv). We first partition the reasoning tree into contiguous text blocks bounded by forking points or the end of the trajectory. SibDiv then computes the average pairwise cosine distance (i.e., $1-$cosine similarity) between the embeddings of sibling blocks originating from the same forking point. By aggregating these values, this metric effectively evaluates how well the forking points promote semantically diverse reasoning branches. Formal definitions are provided in Appendix D.3.

2.3.3 Empirical Analysis

We evaluated each forking strategy by generating reasoning trees under a fixed inference budget, measuring PassRate and SibDiv (see Appendix D.4 for detailed experimental setups).

The results are summarized in Table 1. First, `tok-entropy` achieves the highest SibDiv because performing additional sampling at the model's most uncertain points naturally generates diverse immediate sibling blocks. However, due to the *local-*

ization phenomenon, this high SibDiv is strictly confined to a narrow segment of the reasoning path. The search fails to expand into meaningful structural differences across the entire reasoning tree, resulting in a low PassRate. By simply expanding the granularity of the entropy estimation, `sent-entropy` shows the highest PassRate while incurring minimal loss in SibDiv. In contrast, baselines such as `fixed-seg` and `ATB` inherently avoid localization, yielding higher PassRates than `tok-entropy`, but exhibit lower SibDiv since they do not utilize uncertainty signals. Ultimately, `sent-entropy` emerges as the most effective strategy by successfully translating high semantic diversity into a high PassRate.

3 Methodology

Building upon the empirical insights from Section 2, we propose **DATPO** (**D**ifficulty-**A**daptive Sentence-entropy-guided **T**ree-structured **P**olicy **O**ptimization) to expand a model's intrinsic reasoning coverage (pass@k). An overview of the proposed framework is illustrated in Figure 3.

3.1 Difficulty-Adaptive Tree Search at Train-time

Building on the tree algorithm proposed in Section 2.2.1, we introduce a difficulty-adaptive tree search to dynamically allocate the search budget. The procedure operates in two phases. First, we generate N independent base rollouts and estimate the empirical difficulty of the prompt through their average verifiable reward $V(\text{root}) \in [0, 1]$.

Second, we scale the tree expansion proportionally to this difficulty, where a lower $V(\text{root})$ indicates a more challenging problem, thus allocating more search budget. Given maximum budgets for forking points $K_{\max}$ and branch rollouts $B_{\max}$, the adaptive parameters are computed as:

$$\hat{K} = \lceil K_{\max}(1 - V(\text{root})) \rceil \quad (1)$$

$$\hat{B} = \lceil B_{\max}(1 - V(\text{root})) \rceil \quad (2)$$

Using the `sent-entropy` (Section 2.3.1), we identify $\hat{K}$ forking points within each base trajectory and generate $\hat{B}$ branches from each point.

This mechanism entirely bypasses expansion only when $V(\text{root}) = 1$. For harder problems, it expands the search space up to $N(1 + \hat{K}\hat{B})$ leaves. By scaling expansion based on difficulty, this strategy concentrates exploration on unsolved problems to maximize reasoning coverage.

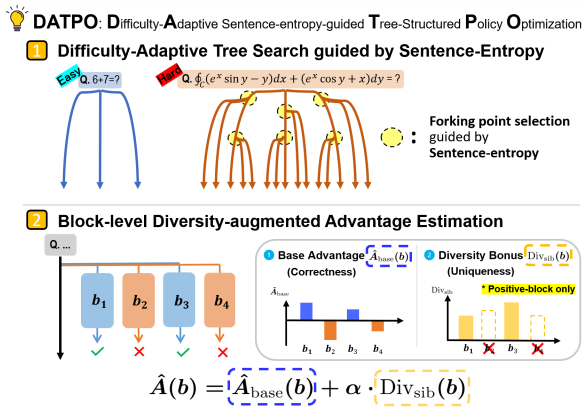


Figure 3: Overview of the proposed DATPO framework.

3.2 Block-level Diversity-augmented Advantage Estimation

To optimize the policy, we partition trajectories into contiguous *blocks* bounded by forking points or the end of the trajectory. Advantages are computed and assigned at this block level.

Since our tree search generates multiple branches from intermediate states, we reliably estimate state values using Monte Carlo (MC) returns (Kazemnejad et al., 2024). For a state s at any forking point, $\hat{V}_{MC}(s)$ is the average verifiable reward of all its descending terminal blocks. For a terminal state s_T without further rollouts, $\hat{V}_{MC}(s_T) = 0$.

For a block b spanning from $s_{\text{start}}^{(b)}$ to $s_{\text{end}}^{(b)}$, the base advantage is formulated as:

$$\hat{A}_{\text{base}}(b) = r(b) + \hat{V}_{MC}(s_{\text{end}}^{(b)}) - \hat{V}_{MC}(s_{\text{start}}^{(b)}) \quad (3)$$

where the verifiable reward $r(b) \in \{0, 1\}$ is assigned exclusively to terminal blocks; otherwise, it is 0. All tokens within b share this identical advantage. This block-level advantage assigns precise credit to intermediate steps, effectively facilitating implicit process supervision.

To maximize the coverage-expanding benefits, we augment the base advantage with a *sibling-diversity* term $\text{Div}_{\text{sib}}(b)$, inspired by the SibDiv metric. Specifically, sibling blocks refer to the child blocks generated from a shared forking point. $\text{Div}_{\text{sib}}(b)$ is defined as the average cosine distance between the embedding of block b and its sibling blocks, effectively encouraging the model to discover distinct reasoning paths. Crucially, we apply this bonus exclusively to blocks with positive base advantages to avoid incentivizing the exploration of incorrect paths. The augmented advantage is:

$$\hat{A}(b) = \hat{A}_{\text{base}}(b) + \mathbb{I}(\hat{A}_{\text{base}}(b) > 0) \cdot \alpha \cdot \text{Div}_{\text{sib}}(b) \quad (4)$$

Method	MATH500		AIME26		AIME25		AIME24		AMC23		Average	
	avg@8	pass@8	avg@64	pass@64	avg@64	pass@64	avg@64	pass@64	avg@64	pass@64	avg@k	pass@k
Qwen2.5-3B-Base												
Base	26.3	64.7	0.4	8.9	0.2	12.2	1.0	22.2	10.0	78.3	7.6	37.3
GRPO	61.8	79.8	1.7	21.1	0.7	24.4	3.6	30.0	35.9	85.8	20.7	48.2
Dr.GRPO	61.3	78.7	**3.0**	23.3	1.4	27.8	5.1	27.8	37.9	83.3	21.7	48.2
TreeRL	61.7	80.7	2.4	22.2	**1.7**	20.0	4.2	24.4	38.7	83.3	21.7	46.1
AttnRL	62.0	**82.1**	1.9	32.2	1.5	25.6	**5.2**	34.4	35.7	**90.8**	21.3	53.0
DATPO (Ours)	**63.5**	81.7	2.4	**33.3**	1.6	**33.3**	4.8	**35.6**	**39.8**	**90.8**	**22.4**	**54.9**
Qwen3-4B-Base												
Base	39.9	79.4	1.7	21.1	1.1	32.2	2.8	38.9	17.2	84.2	12.5	51.2
GRPO	75.5	87.4	7.1	32.2	10.3	38.9	9.6	41.1	47.9	**91.7**	30.1	58.3
Dr.GRPO	75.2	86.2	7.8	25.6	7.7	35.6	9.7	40.0	46.6	90.8	29.4	55.6
TreeRL	75.3	86.2	**7.9**	28.9	9.4	36.7	9.4	41.1	49.4	90.8	30.3	56.7
AttnRL	74.7	87.5	7.1	28.9	8.2	36.7	11.0	42.2	**52.5**	**91.7**	30.7	57.4
DATPO (Ours)	**76.4**	**87.9**	6.5	**33.3**	**10.9**	**42.2**	**14.0**	**46.7**	48.4	**91.7**	**31.3**	**60.4**

Table 2: Evaluation results on mathematical reasoning benchmarks. We report avg@k and pass@k for each dataset.

where $\mathbb{I}(\cdot)$ is the indicator function, and the coefficient α is linearly annealed during training to gradually decay the exploration incentive, allowing the model to refine its learned reasoning paths.

Finally, we optimize the policy directly across the generated tree topology. Given a set of contiguous blocks $\mathcal{B}$ generated for a prompt q, the DATPO objective is formulated as:

$$\mathcal{J}_{\text{DATPO}}(\theta) = \mathbb{E}_{q\sim\mathcal{Q},\mathcal{B}\sim\pi_{\theta_{\text{old}}}}\Bigg[\frac{1}{\sum_{b\in\mathcal{B}}|b|}\sum_{b\in\mathcal{B}}\sum_{t=1}^{|b|} \min\Big(\rho_{b,t}(\theta)\hat{A}(b), \text{clip}\big(\rho_{b,t}(\theta), 1-\epsilon, 1+\epsilon\big)\hat{A}(b)\Big)\Bigg], \tag{5}$$

where $|b|$ is the sequence length of block b, and $\hat{A}(b)$ is the block-level augmented advantage. Comprehensive algorithmic details and training procedures are provided in Appendix E.

4 Experiments

4.1 Experimental Setup

Models and Datasets We use Qwen2.5-3B-Base (Yang et al., 2024) and Qwen3-4B-Base (Yang et al., 2025) as base models. For training, we use the MATH dataset (Hendrycks et al., 2021).

Evaluation We evaluate the trained models on mathematical reasoning benchmarks, specifically MATH500 (Hendrycks et al., 2021), AIME26, AIME25, AIME24, and AMC23. Using a temperature of 1.0, we report avg@k as the primary evaluation metric across all benchmarks. Additionally, we report pass@k to assess the reasoning coverage of the trained models. For robustness, the pass@k results are computed by averaging over three independent evaluation runs.

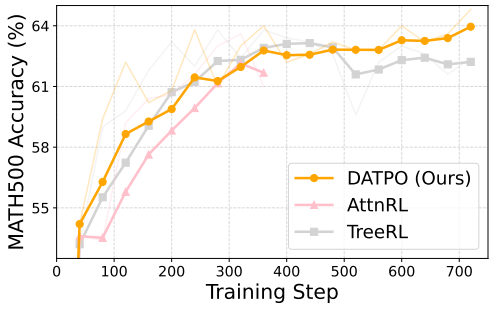


Figure 4: Learning curves of MATH500 accuracy over training steps for tree-based methods. (Smoothed)

Baselines We compare DATPO against the **Base** model and several advanced RLVR baselines. These include **GRPO** (Shao et al., 2024) (enhanced with clip-higher and token-level loss (Yu et al., 2025b)) and its variant, **Dr.GRPO** (Liu et al., 2025b). Furthermore, we evaluate two tree-based methods: **TreeRL** (Hou et al., 2025), which utilizes top-k high-entropy forking and combined local-global advantages, and **AttnRL** (Liu et al., 2025a), which leverages attention-based forking, adaptive sampling, and a one-step off-policy for enhanced efficiency. To ensure a fair comparison, we maintain a comparable total number of generated tokens per problem across all methods while adopting the tree-expansion hyperparameters reported in the original TreeRL and AttnRL papers. Comprehensive details for experiments are provided in Appendix F.

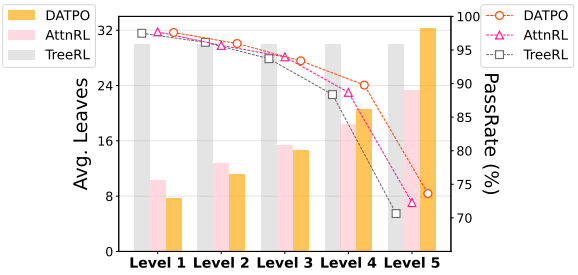


Figure 5: Comparison of average leaf count generated at train-time and PassRate across difficulty levels for tree-based methods.

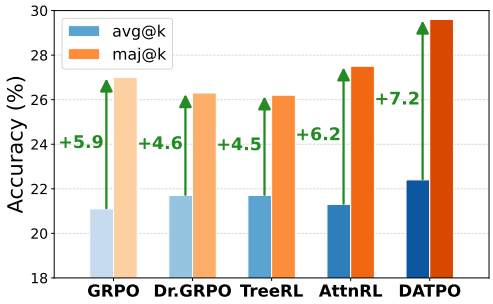


Figure 6: Test-time scaling performance using majority voting (maj@k). The green arrows indicate the performance improvement of maj@k over avg@k.

4.2 Main Results

Mathematical Reasoning Performance As reported in Table 2, DATPO achieves the best aggregate avg@k and pass@k across the evaluated benchmarks. Notably, while the improvements in single-sample accuracy (avg@k) are marginal compared to the strongest baseline, AttnRL (e.g., +1.1 and +0.6 on Qwen2.5-3B-Base and Qwen3-4B-Base, respectively), DATPO achieves substantial gains in pass@k, outperforming AttnRL by +1.9 and +3.0, respectively. This indicates that DATPO's difficulty-adaptive rollout and sibling-diversity term successfully expand the model's intrinsic reasoning coverage.

Analysis of Training Dynamics We first examine the MATH500 accuracy over training steps (Figure 4). While all three tree-based methods exhibit similar accuracy gains during the initial training steps, TreeRL and AttnRL fail to sustain this upward trend, eventually plateauing or even suffering performance degradation. In contrast, DATPO shows a consistent upward trend throughout the training. This stability can be attributed to the annealed sibling-diversity term, which injects semantic diversity early in training to prevent premature convergence.

Furthermore, we analyze the search behavior during training by comparing the average leaf count and PassRate across the difficulty levels of the MATH dataset (Figure 5). Note that these difficulty labels are used strictly for this analysis and were not provided to the model during training. Unlike TreeRL, which maintains a fixed leaf count across all difficulties, DATPO and AttnRL employ difficulty-adaptive rollouts. However, DATPO allocates fewer rollouts to easy problems and significantly more rollouts to hard problems compared to AttnRL. The latter's adaptive strategy relies heavily on an attention-based filtering mechanism that simply discards problems with below-average attention scores to maximize computational efficiency. Because this approach acts as a rigid cut-off, it fails to concentrate computational resources on the most challenging problems. Consequently, DATPO's highly adaptive resource allocation enables it to achieve a higher PassRate on harder problems (Levels 4 and 5) compared to other tree-based methods.

4.3 Test-time Scaling Performance

In this section, we investigate whether the substantial gains in pass@k directly translate into improved test-time scaling performance. To this end, we apply majority voting (maj@k) (Wang et al., 2022)—one of the simplest and most widely adopted test-time scaling strategies—to compare the methods applied to Qwen2.5-3B-Base. We evaluate the effectiveness of these methods across five mathematical reasoning benchmarks, applying $k = 8$ for MATH500 and $k = 64$ for the remaining datasets. For robustness, all maj@k results are computed as the average over three independent evaluation runs.

As illustrated by the average performance across five benchmarks in Figure 6, DATPO, which achieved the highest pass@k during training, also shows the largest maj@k gain, improving by +7.2 over its avg@k. This demonstrates that expanding a model's reasoning coverage during training directly enhances test-time scaling performance.

4.4 Ablation Studies

Effect of Forking Strategies Building upon the analysis during inference presented in Section 2.3, we further investigate the impact of different forking strategies during training. We compare DATPO's default `sent-entropy` strategy against

Method	MATH500	
	avg@8	pass@8
`random`	61.5	81.6
`fixed-seg`	61.1	80.6
`ATB`	62.1	80.7
`tok-entropy`	61.3	80.3
`sent-entropy`	**63.5**	**81.7**

Table 3: Ablation study on different forking strategies.

four alternatives: `random`, `fixed-seg`, `ATB`, and `tok-entropy`, evaluating their performance on the MATH500 benchmark. As reported in Table 3, the `sent-entropy` yields the highest performance. This confirms our hypothesis: the `sent-entropy` effectively guides exploration using the model's uncertainty, while avoiding *localization* that limits `tok-entropy`.

Effect of Sibling-Diversity To evaluate the effect of the sibling-diversity term, we conduct an ablation study by varying the coefficient α, and evaluate the performance on the MATH500 benchmark, as summarized in Table 4.

The results show that the $0.2 \rightarrow 0$ schedule is the most effective, surpassing both the baseline without diversity ($0 \rightarrow 0$) and a higher initial coefficient ($0.4 \rightarrow 0$). Furthermore, applying the diversity bonus to all blocks ($0.2 \rightarrow 0$, *all*) degrades performance compared to the baseline without diversity, highlighting the need to reward diversity exclusively on valid paths. Annealing is also critical: a constant α ($0.2 \rightarrow 0.2$) severely degrades avg@8 despite slight pass@8 gains. Conversely, annealing α to a negative value ($0.2 \rightarrow -0.2$) improves avg@8 but noticeably harms pass@8, confirming that penalizing diversity in the later stages of training restricts the model's reasoning coverage.

5 Related Work

Reinforcement Learning for LLM Reasoning RLVR has become the standard for post-training of reasoning LLMs. A representative method is Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which replaces the costly critic model of PPO (Schulman et al., 2017) with efficient group-based advantage estimation. Recent studies build on this framework to enhance stability and resolve optimization issues: DAPO (Yu et al., 2025b) prevents mode collapse through decoupled clipping and dynamic sampling, while Dr.GRPO (Liu et al., 2025b) corrects inherent structural biases in the advantage computation.

α	MATH500	
	avg@8	pass@8
$0 \rightarrow 0$	62.1	81.4
$0.2 \rightarrow 0$	**63.5**	81.7
$0.4 \rightarrow 0$	61.9	81.3
$0.2 \rightarrow 0$, *all*	61.6	81.3
$0.2 \rightarrow 0.2$	61.5	**82.0**
$0.2 \rightarrow$ -0.2	63.4	81.0

Table 4: Ablation study on the diversity coefficient α. The right arrow ($\rightarrow$) indicates the linear annealing schedule during training. The *all* applies the diversity term to all blocks, rather than exclusively to those with positive base advantages.

Exploration Strategies in RLVR RLVR often fails to expand a model's intrinsic reasoning capacity (pass@k) beyond its base capabilities (Yue et al., 2025; Dang et al., 2025; Wu et al., 2025). To overcome this exploration bottleneck, recent studies introduce explicit mechanisms to diversify search. PKPO (Walder and Karkhanis, 2025) directly optimizes pass@k to solve harder instances, while R1-zero-Div (Yao et al., 2025) and FOR (Yu et al., 2025a) use diversity-aware objectives and divergent reasoning flows to prevent trajectory collapse.

Tree-based Search in Reinforcement Learning Proven in RL milestones like AlphaGo (Silver et al., 2016), tree-based search is now adapted to LLMs to enable systematic exploration. TreeRL (Hou et al., 2025) pioneers this with an on-policy framework using entropy-guided branching. To address computational bottlenecks, AttnRL (Liu et al., 2025a) improves efficiency by leveraging difficulty-aware adaptive sampling within a one-step off-policy pipeline. While these methods use trees primarily for credit assignment or computational efficiency, we leverage them to expand the model's reasoning coverage.

6 Conclusion

In this work, we investigate the structural design of train-time rollouts in RLVR to expand the intrinsic reasoning coverage. Our analysis reveals that expanding reasoning coverage can benefit from moving beyond uniform parallel sampling toward difficulty-adaptive resource allocation and the struc-

tural advantages of sentence-entropy-guided tree search. Integrating these principles, we propose DATPO, which explicitly encourages semantic exploration through a sibling-diversity term. Experiments on mathematical benchmarks validate our approach, showing significant improvements in pass@k.

Limitations

While DATPO significantly expands reasoning coverage, calculating the sibling-diversity term requires additional forward passes through an external embedding model, which introduces computational overhead.

Additionally, while our block-level formulation effectively assigns process-level credit across the tree topology, it relies on small sample sizes. Deriving both the empirical difficulty and Monte Carlo state values from a limited number of rollouts (e.g., $N = 4, B = 4$) can inject noise into the estimations, occasionally yielding high variance during policy updates.

Finally, resource limitations restricted our empirical validation to the 3B-4B parameter regime, leaving DATPO's scalability to larger models ($\geq$ 7B) unverified. Furthermore, as our evaluation primarily focuses on mathematical reasoning, the framework's direct effectiveness in other rigorous domains, such as complex logical reasoning or code generation, requires further investigation.

Acknowledgments

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (No. RS-2019-II191906, Artificial Intelligence Graduate School Program (POSTECH); IITP-2026-RS-2026-25616370, AI Star Fellowship Support Program) and by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2024-00335873).

References

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

Lang Cao, Hui Ruan, Yongqian Li, Peng Chao, Wu Ning, Haonan Song, Renhong Chen, and Yitong Li. 2026. Treeadv: Tree-structured advantage redistribution for group-based rl. *arXiv preprint arXiv:2601.03703*.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335, Bangkok, Thailand. Association for Computational Linguistics.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. 2025. Reasoning with exploration: An entropy perspective. *arXiv preprint arXiv:2506.14758*.

Xingyu Dang, Christina Baek, J Zico Kolter, and Aditi Raghunathan. 2025. Assessing diversity collapse in reasoning. In *Scaling Self-Improving Foundation Models without Human Supervision*.

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

Rui Ha, Chaozhuo Li, Rui Pu, Litian Zhang, Xi Zhang, and Sen Su. 2025. DSG-MCTS: A dynamic strategy-guided Monte Carlo tree search for diversified reasoning in large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 10530–10544, Suzhou, China. Association for Computational Linguistics.

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

Zhenyu Hou, Ziniu Hu, Yujiang Li, Rui Lu, Jie Tang, and Yuxiao Dong. 2025. Treerl: Llm reinforcement learning with on-policy tree search. *arXiv preprint arXiv:2506.11902*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Zishang Jiang, Jinyi Han, Tingyun Li, Xinyi Wang, Sihang Jiang, Jiaqing Liang, Zhaoqian Dai, Shuguang Ma, Fei Yu, and Yanghua Xiao. 2025. Selective expert guidance for effective and diverse exploration in reinforcement learning of llms. *arXiv preprint arXiv:2510.04140*.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. 2024. Vineppo: Refining credit assignment in rl training of llms. *arXiv preprint arXiv:2410.01679*.

Yizhi Li, Qingshui Gu, Zhoufutu Wen, Ziniu Li, Tianshun Xing, Shuyue Guo, Tianyu Zheng, Xin Zhou, Xingwei Qu, Wangchunshu Zhou, and 1 others. 2025a. Treepo: Bridging the gap of policy optimization and efficacy and inference efficiency with heuristic tree-based modeling. *arXiv preprint arXiv:2508.17445*.

Ziniu Li, Congliang Chen, Tianyun Yang, Tian Ding, Ruoyu Sun, Ge Zhang, Wenhao Huang, and Zhi-Quan Luo. 2025b. Knapsack rl: Unlocking exploration of llms via optimizing budget allocation. *arXiv preprint arXiv:2509.25849*.

Mengqi Liao, Xiangyu Xi, Chen Ruinian, Jia Leng, Yangen Hu, Ke Zeng, Shuai Liu, and Huaiyu Wan. 2025. Enhancing efficiency and exploration in reinforcement learning for LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 1451–1463, Suzhou, China. Association for Computational Linguistics.

Runze Liu, Jiakang Wang, Yuling Shi, Zhihui Xie, Chenxin An, Kaiyan Zhang, Jian Zhao, Xiaodong Gu, Lei Lin, Wenping Hu, and 1 others. 2025a. Attention as a compass: Efficient exploration for process-supervised rl in reasoning models. *arXiv preprint arXiv:2509.26628*.

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

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*.

Nipun Sadvilkar and Mark Neumann. 2020. PySBD: Pragmatic sentence boundary disambiguation. In *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*, pages 110–114, Online. Association for Computational Linguistics.

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

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*.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, and 1 others. 2016. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, and 77 others. 2025. Kimi k1.5: Scaling reinforcement learning with llms. *Preprint*, arXiv:2501.12599.

Hieu Tran, Zonghai Yao, and Hong Yu. 2025. Exploiting tree structure for credit assignment in rl training of llms. *arXiv preprint arXiv:2509.18314*.

Christian Walder and Deep Karkhanis. 2025. Pass@ k policy optimization: Solving harder reinforcement learning problems. *arXiv preprint arXiv:2505.15201*.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, and 1 others. 2025. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Fang Wu, Weihao Xuan, Ximing Lu, Mingjie Liu, Yi Dong, Zaid Harchaoui, and Yejin Choi. 2025. The invisible leash: Why rlvr may or may not escape its origin. *arXiv preprint arXiv:2507.14843*.

Shangyu Xing, Siyuan Wang, Chenyuan Yang, Xinyu Dai, and Xiang Ren. 2025. Lookahead tree-based

rollouts for enhanced trajectory-level exploration in reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2510.24302*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Jian Yao, Ran Cheng, Xingyu Wu, Jibin Wu, and Kay Chen Tan. 2025. Diversity-aware policy optimization for large language model reasoning. *arXiv preprint arXiv:2505.23433*.

Fangxu Yu, Lai Jiang, Haoqiang Kang, Shibo Hao, and Lianhui Qin. 2025a. Flow of reasoning: Training llms for divergent reasoning with minimal examples. In *Forty-second International Conference on Machine Learning*.

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

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*.

Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, Meishan Zhang, Wenjie Li, and Min Zhang. 2024. mGTE: Generalized long-context text representation and reranking models for multilingual text retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1393–1412, Miami, Florida, US. Association for Computational Linguistics.

Yuheng Zhang, Wenlin Yao, Changlong Yu, Yao Liu, Qingyu Yin, Bing Yin, Hyokun Yun, and Lihong Li. 2025. Improving sampling efficiency in rlvr through adaptive rollout and response reuse. *arXiv preprint arXiv:2509.25808*.

Eric Zhao, Pranjal Awasthi, and Sreenivas Gollapudi. 2025. Sample, scrutinize and scale: Effective inference-time search by scaling verification. *arXiv preprint arXiv:2502.01839*.

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

Tianyu Zheng, Tianshun Xing, Qingshui Gu, Taoran Liang, Xingwei Qu, Xin Zhou, Yizhi Li, Zhoufutu Wen, Chenghua Lin, Wenhao Huang, and 1 others. 2025b. First return, entropy-eliciting explore. *arXiv preprint arXiv:2507.07017*.

A Theoretical Analysis of Difficulty-Adaptive Rollout

We provide a theoretical analysis of the empirical findings in Section 2.1. In particular, we explain why increasing the rollout budget G consistently improves the expected avg@k, while the behavior of pass@k can vary across difficulty subsets.

Notation. For a prompt x and a policy π_θ, let

$$p_x(\theta) := \Pr_{y\sim\pi_\theta(\cdot|x)}[r(y)=1]$$

denote the single-sample correctness probability under a binary verifiable reward $r(y) \in \{0,1\}$.

In empirical evaluations, metrics are computed over a finite set of k independent samples $\mathcal{Y} = (y^{(1)},\dots,y^{(k)}) \overset{\text{i.i.d.}}{\sim} \pi_\theta(\cdot \mid x)$. We define the empirical estimators as:

$$\widehat{\text{avg@k}}_x(\theta) := \frac{1}{k}\sum_{j=1}^{k} r\left(y^{(j)}\right),$$

$$\widehat{\text{pass@k}}_x(\theta) := \mathbb{I}\left(\sum_{j=1}^{k} r\left(y^{(j)}\right) \geq 1\right).$$

However, to rigorously analyze the optimization dynamics, we focus on their *expected values* with respect to the policy. Throughout this analysis, we omit the hat notation and define $\text{avg@k}_x(\theta)$ and $\text{pass@k}_x(\theta)$ strictly as these theoretical expectations:

$$\begin{aligned}\text{avg@k}_x(\theta) &:= \mathbb{E}_{\mathcal{Y}\sim\pi_\theta}\left[\widehat{\text{avg@k}}_x(\theta)\right] \\ &= p_x(\theta)\end{aligned}$$

and

$$\begin{aligned}\text{pass@k}_x(\theta) &:= \mathbb{E}_{\mathcal{Y}\sim\pi_\theta}\left[\widehat{\text{pass@k}}_x(\theta)\right] \\ &= 1-\big(1-p_x(\theta)\big)^k.\end{aligned}$$

By formulating these metrics as expectations, the theoretical avg@k rigorously simplifies to the single-sample success probability $p_x(\theta)$.

For the analysis below, given the current policy parameters θ, let

$$\theta_G^+ := \theta + \eta\hat{g}_G(x)$$

denote one GRPO update with rollout group size G, learning rate η, and stochastic gradient estimator $\hat{g}_G(x)$. We consider only the reward-advantage component of the GRPO update, excluding auxiliary regularization terms.

Theorem A.1 (One-step expected avg@k improvement with rollout budget). *Fix a prompt x and a policy π_θ. Let $p := p_x(\theta) \in (0,1)$, and let $S_G \sim \text{Binomial}(G,p)$ denote the number of correct responses in a rollout group size $G \geq 2$.*

Assume $p_x(\cdot)$ is twice continuously differentiable with a bounded Hessian, and the gradient estimator satisfies $\mathbb{E}[\|\hat{g}_G(x)\|^2] < \infty$. Further assume:

A1. *The update vanishes if all group rewards are identical: $\hat{g}_G(x) = 0$ almost surely if $S_G \in \{0, G\}$.*

A2. *The expected update direction on mixed-reward batches is a positive constant $c_x > 0$ independent of G:*

$$\begin{aligned}&\mathbb{E}[\langle\nabla_\theta p_x(\theta), \hat{g}_G(x)\rangle \mid 1 \leq S_G \leq G-1] \\ &\quad = c_x.\end{aligned}$$

Then, for sufficiently small η, the expected correctness probability satisfies:

$$\mathbb{E}[p_x(\theta_G^+)] = p_x(\theta) + \eta c_x \Psi_G(p) + O_G(\eta^2),$$

where $\Psi_G(p) := 1 - p^G - (1-p)^G$.

Consequently, for any finite set of rollout budgets $\mathcal{G}$, there exists $\eta_0 > 0$ such that for all $\eta \in (0,\eta_0)$ and any $G_1, G_2 \in \mathcal{G}$ with $G_2 > G_1$:

$$\mathbb{E}[p_x(\theta_{G_2}^+)] > \mathbb{E}[p_x(\theta_{G_1}^+)].$$

Furthermore, the first-order gain coefficient $\Psi_G(p)$ is strictly increasing in G with diminishing returns.

Proof. By definition, theoretical expected avg@k simplifies to the single-sample correctness probability $p_x(\theta)$. Let $E_G := \{1 \leq S_G \leq G-1\}$ be the event of a mixed-reward batch, which occurs with probability $\Pr(E_G) = 1-(1-p)^G-p^G = \Psi_G(p)$.

By A1, the update is zero outside E_G. Applying A2, the expected directional derivative is:

$$\begin{aligned}&\mathbb{E}[\langle\nabla_\theta p_x(\theta), \hat{g}_G(x)\rangle] \\ &\quad = \Pr(E_G)\mathbb{E}[\langle\nabla_\theta p_x(\theta), \hat{g}_G(x)\rangle \mid E_G] \\ &\quad = c_x\Psi_G(p).\end{aligned}$$

Since $p_x(\cdot)$ has a bounded Hessian on the update region, Taylor's theorem gives

$$p_x(\theta_G^+) = p_x(\theta) + \eta\,\langle\nabla_\theta p_x(\theta), \hat{g}_G(x)\rangle + R_G(\eta),$$

where, for some Hessian bound $L < \infty$, $|R_G(\eta)| \leq \frac{L}{2}\eta^2\|\hat{g}_G(x)\|^2$. Taking expectations

and using $\mathbb{E}[\|\hat{g}_G(x)\|^2] < \infty$ gives $\mathbb{E}[R_G(\eta)] = O_G(\eta^2)$, completing the first claim.

To establish monotonicity, we evaluate the marginal increase in $\Psi_G(p)$ for $p \in (0,1)$:

$$\Psi_{G+1}(p) - \Psi_G(p) = p(1-p)^G + (1-p)p^G > 0,$$

confirming $\Psi_G(p)$ strictly increases with G. For any $G_2 > G_1$, the first-order difference $\eta c_x(\Psi_{G_2}(p) - \Psi_{G_1}(p))$ is strictly positive and dominates the $O_{G_1,G_2}(\eta^2)$ remainder term for η strictly less than some threshold $\eta_0(G_1, G_2) > 0$. Taking the minimum η_0 across all pairs in the finite set $\mathcal{G}$ establishes the strict ordering simultaneously.

Finally, the second difference of $\Psi_G(p)$ is $-p^2(1-p)^G - (1-p)^2 p^G < 0$, confirming diminishing returns. □

Remark on Assumption 2. A2 serves as an idealized first-order approximation to isolate the core mechanism: larger groups increase the mixed-reward probability $\Psi_G(p)$, thereby providing nonzero relative reward signals more frequently. In practice, GRPO normalizes advantages relative to the group (e.g., the positive advantage scales as $\sqrt{(G-s)/s}$), meaning the exact expected magnitude c_x technically depends on G. However, this assumption keeps the mechanism analytically transparent and highlights a plausible first-order driver of the empirical monotonicity observed during training.

Remark on advantage-estimation variance. Our analysis isolates the *signal-availability* mechanism, demonstrating that a larger G yields more frequent informative updates. We do not explicitly model the variance reduction in advantage estimation to avoid complex distributional assumptions. Therefore, this theorem formalizes one core driver of the avg@k improvement, acknowledging that other statistical benefits also contribute.

Extension to Dataset-Level Notation. While Theorem A.1 characterizes the optimization dynamics of a single prompt, explaining the difficulty-dependent behavior requires dataset-level formalization. Let $p_{x,d}(G)$ denote the single-sample correctness probability on a test prompt $x \in \mathcal{D}$ for a policy trained on difficulty subset d using rollout budget G. We define the expected average correctness ($\mu_d(G)$) and pass@k over the dataset $\mathcal{D}$ as:

$$\mu_d(G) := \mathbb{E}_{x\sim\mathcal{D}}[p_{x,d}(G)],$$
$$\text{pass@k}_{\mathcal{D}}^{(d)}(G) := \mathbb{E}_{x\sim\mathcal{D}}\Big[1 - \big(1 - p_{x,d}(G)\big)^k\Big].$$

Theorem A.2 (Variance Penalty Decomposition of pass@k). *Fix $k \geq 2$. Let $f_k(p) := 1-(1-p)^k$ and define the Jensen gap*

$$J_{k,d}(G) := f_k\big(\mu_d(G)\big) - \textit{pass@}k_{\mathcal{D}}^{(d)}(G).$$

Then the following hold:

1. *$J_{k,d}(G) \geq 0$, with equality if and only if $p_{x,d}(G)$ is constant almost surely over $x \sim \mathcal{D}$.*
2. *For any two rollout budgets $G_2 > G_1$,*

$$\begin{aligned}&\textit{pass@}k_{\mathcal{D}}^{(d)}(G_2) - \textit{pass@}k_{\mathcal{D}}^{(d)}(G_1)\\ &= \underbrace{f_k\big(\mu_d(G_2)\big) - f_k\big(\mu_d(G_1)\big)}_{\textit{gain from higher avg@}k_{\mathcal{D}}}\\ &\quad - \underbrace{\big(J_{k,d}(G_2) - J_{k,d}(G_1)\big)}_{\textit{loss from test-set prompt heterogeneity}}.\end{aligned}$$

Moreover, if $\sigma_d^2(G) := \mathrm{Var}_{x\sim\mathcal{D}}[p_{x,d}(G)]$, then

$$\begin{aligned}J_{k,d}(G) = \frac{k(k-1)}{2}\big(1-\mu_d(G)\big)^{k-2}\sigma_d^2(G)\\ + O\big(\mathbb{E}_{x\sim\mathcal{D}}\left[|p_{x,d}(G) - \mu_d(G)|^3\right]\big).\end{aligned}$$

Hence, to second order, the penalty term is controlled by the cross-prompt dispersion of one-sample success probabilities on the test set.

Proof. For $k \geq 2$ and $p \in (0,1)$,

$$f_k'(p) = k(1-p)^{k-1} > 0,$$
$$f_k''(p) = -k(k-1)(1-p)^{k-2} < 0,$$

so f_k is strictly increasing and strictly concave on $(0,1)$.

By definition, the expected mean is

$$\mu_d(G) = \mathbb{E}_{x\sim\mathcal{D}}[p_{x,d}(G)],$$

and the expected pass rate is

$$\text{pass@k}_{\mathcal{D}}^{(d)}(G) = \mathbb{E}_{x\sim\mathcal{D}}[f_k(p_{x,d}(G))].$$

Applying Jensen's inequality to the concave function f_k,

$$\begin{aligned}\text{pass@k}_{\mathcal{D}}^{(d)}(G) &= \mathbb{E}_{x\sim\mathcal{D}}[f_k(p_{x,d}(G))]\\ &\leq f_k(\mathbb{E}_{x\sim\mathcal{D}}[p_{x,d}(G)])\\ &= f_k\big(\mu_d(G)\big).\end{aligned}$$

Thus $J_{k,d}(G) \geq 0$. Since f_k is strictly concave, equality holds if and only if $p_{x,d}(G)$ is constant almost surely.

Now rewrite

$$\text{pass@k}_{\mathcal{D}}^{(d)}(G) = f_k\big(\mu_d(G)\big) - J_{k,d}(G).$$

Subtracting the identities for G_2 and G_1 yields

$$\begin{aligned}&\text{pass@k}_{\mathcal{D}}^{(d)}(G_2) - \text{pass@k}_{\mathcal{D}}^{(d)}(G_1)\\ &= \Big(f_k\big(\mu_d(G_2)\big) - f_k\big(\mu_d(G_1)\big)\Big)\\ &\quad - \Big(J_{k,d}(G_2) - J_{k,d}(G_1)\Big),\end{aligned}$$

which gives the exact decomposition.

For the second-order approximation, apply Taylor's theorem around $\mu_d(G)$:

$$\begin{aligned}f_k(p_{x,d}(G)) &= f_k\big(\mu_d(G)\big)\\ &\quad + f_k'\big(\mu_d(G)\big)\big(p_{x,d}(G) - \mu_d(G)\big)\\ &\quad + \frac{1}{2}f_k''\big(\mu_d(G)\big)\big(p_{x,d}(G) - \mu_d(G)\big)^2\\ &\quad + O(|p_{x,d}(G) - \mu_d(G)|^3).\end{aligned}$$

Taking the expectation over $x \sim \mathcal{D}$, the linear term vanishes because $\mathbb{E}_{x\sim\mathcal{D}}[p_{x,d}(G) - \mu_d(G)] = 0$. Hence

$$\begin{aligned}\mathbb{E}_{x\sim\mathcal{D}}[f_k(p_{x,d}(G))] &= f_k\big(\mu_d(G)\big)\\ &\quad + \frac{1}{2}f_k''\big(\mu_d(G)\big)\sigma_d^2(G)\\ &\quad + O\Big(\mathbb{E}_{x\sim\mathcal{D}}\big[|p_{x,d}(G)\\ &\qquad - \mu_d(G)|^3\big]\Big).\end{aligned}$$

Rearranging gives

$$\begin{aligned}J_{k,d}(G) &= -\frac{1}{2}f_k''\big(\mu_d(G)\big)\sigma_d^2(G)\\ &\quad + O\Big(\mathbb{E}_{x\sim\mathcal{D}}\big[|p_{x,d}(G) - \mu_d(G)|^3\big]\Big),\end{aligned}$$

and substituting $f_k''(\mu) = -k(k-1)(1-\mu)^{k-2}$ completes the proof. □

Connection to Empirical Regimes. Rather than positing a universal law applicable to all training environments, Theorem A.2 provides a rigorous analytical lens to interpret the specific empirical observations reported in Section 2.1. Specifically, while the theorem formally establishes the mathematical *mechanism*—that the dynamics of $\text{pass@k}_{\mathcal{D}}$ are entirely governed by a trade-off between the mean improvement gain (Δf_k) and the cross-prompt variance penalty ($\Delta J_{k,d}$)—the actual increase of this variance ($\sigma_d^2(G)$) is an *empirical characteristic* dictated by the intrinsic difficulty of the dataset and specific training dynamics. Consequently, while these exact behavioral patterns may not identically manifest in every setup, bridging this theoretical mechanism with our specific findings allows us to formally characterize the three observed regimes as follows:

Easy: $\Delta_G J_{k,\text{Easy}}(G) > \Delta_G f_{k,\text{Easy}}(G)$

$$\implies \text{pass@k}_{\mathcal{D}}^{(\text{Easy})}(G+1) < \text{pass@k}_{\mathcal{D}}^{(\text{Easy})}(G),$$

Hard: $\Delta_G J_{k,\text{Hard}}(G) < \Delta_G f_{k,\text{Hard}}(G)$

$$\implies \text{pass@k}_{\mathcal{D}}^{(\text{Hard})}(G+1) > \text{pass@k}_{\mathcal{D}}^{(\text{Hard})}(G),$$

Medium: $\Delta_G J_{k,\text{Medium}}(G)$ crosses $\Delta_G f_{k,\text{Medium}}(G)$

$$\implies \text{pass@k}_{\mathcal{D}}^{(\text{Medium})}(G) \text{ is non-monotonic.}$$

where $\Delta_G J_{k,d}(G) = J_{k,d}(G+1) - J_{k,d}(G)$ and $\Delta_G f_{k,d}(G) = f_k(\mu_d(G+1)) - f_k(\mu_d(G))$.

Analytically, the second-order approximation reveals that the penalty term $J_{k,d}(G)$ is directly proportional to the cross-prompt variance $\sigma_d^2(G)$. This theoretically explains the paradoxical degradation in $\text{pass@k}_{\mathcal{D}}$ observed when training on the **Easy** dataset. When the rollout budget G is large, the policy over-exploits specific, easily solvable templates rather than learning generalizable reasoning skills. Evaluated on a general test set, this causes severe polarization: the success probability $p_{x,d}(G)$ converges to 1 for familiar easy prompts but remains near 0 for unseen or harder problems. Consequently, the surge in the variance penalty ($\Delta_G J_{k,\text{Easy}}(G)$) outweighs the marginal gains in mean performance ($\Delta_G f_{k,\text{Easy}}(G)$), resulting in a decrease in the overall $\text{pass@k}_{\mathcal{D}}$.

Conversely, on the **Hard** dataset, valid reasoning paths are scarce. An increased rollout budget promotes broader exploration, enabling the model to internalize more generalizable reasoning patterns. This uniformly improves the test-set mean $\mu_d(G)$ across prompts, maintaining low cross-prompt variance and keeping the associated penalty minimal.

B Experimental Details of Section 2.1

We train Qwen2.5-3B-Instruct (Yang et al., 2024) on the MATH (Hendrycks et al., 2021) dataset, which we partition into Easy, Medium, and Hard subsets based on the base model's empirical accuracy. Specifically, for each problem, we sample 12 independent solutions from the base model and categorize problems according to the number of

Training Configuration	Wall Clock (Hours)
Easy / $G = 4$	5.0
Easy / $G = 8$	9.2
Easy / $G = 16$	16.4
Medium / $G = 4$	5.4
Medium / $G = 8$	9.4
Medium / $G = 16$	16.6
Hard / $G = 4$	5.5
Hard / $G = 8$	9.9
Hard / $G = 16$	17.5

Table 5: Wall Clock Time for each difficulty subset and rollout budget.

correct samples. Problems for which all 12 samples are either entirely correct or entirely incorrect are excluded, as they provide no informative signal for distinguishing difficulty. Among the remaining problems, those with 1–4 correct samples are categorized as *Hard*, 5–8 as *Medium*, and 9–11 as *Easy*. This results in 2,217 Easy, 1,433 Medium, and 1,712 Hard problems. To ensure a balanced training distribution, we subsample each subset to 1,400 problems. The model was trained for 5 epochs on each difficulty-specific dataset.

Using GRPO (Shao et al., 2024), we conduct a total of 9 training runs, applying rollout budgets of $G \in \{4, 8, 16\}$ for each difficulty subset. Our implementation is based on a modified version of the GRPO-Zero repository[2], which adopts a token-level policy gradient without KL regularization or clipping. We use a binary verifiable reward, assigning 1 to correct final answers and 0 otherwise. During training, all samples are generated with temperature 1.0 and a maximum generation length of 1024 tokens. We fix all optimization hyperparameters, including a batch size of 256, while adjusting the number of problems per batch (64, 32, 16) to realize group sizes of 4, 8, and 16, respectively. All experiments are conducted on a single A100 GPU. The wall-clock times for each training configuration are reported in Table 5.

We evaluate the resulting 9 models on three benchmarks: MATH500, AIME25, and AIME24. For each problem, we generate 256 reasoning paths in parallel. We compute pass@k for $k \in \{1, 2, 4, 8, 16, 32, 64, 128, 256\}$, and report results for each model trained under different difficulty subsets and rollout budgets (Figure 7). In addition, we aggregate results across the three benchmarks and report the averaged avg@256 and pass@256, as shown in Figure 1.

[2] https://github.com/policy-gradient/GRPO-Zero

Algorithm 1 Tree Rollout

Input: Policy π_θ, Prompt x, Base rollouts N, Forking points K, Branch rollouts B, Selection `method`.
Output: Trajectory set $\mathcal{T}$
Initialize $\mathcal{R} \leftarrow \emptyset$, $\mathcal{T} \leftarrow \emptyset$
Phase 1: Base Rollouts
for $i = 1$ to N **do**
  Sample base trajectory $\tau^{(i)} \sim \pi_\theta(\cdot \mid x)$
  $\mathcal{R} \leftarrow \mathcal{R} \cup \{\tau^{(i)}\}$
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{\tau^{(i)}\}$
end for
Phase 2: Branch Rollouts
for each $\tau \in \mathcal{R}$ **do**
  Select K forking points $\mathcal{F} \subset \{1, \ldots, |\tau|\}$ via Algorithm 2
  for each $f \in \mathcal{F}$ **do**
    Extract trajectory prefix $x_f \leftarrow [x; \tau_{1:f}]$
    for $j = 1$ to B **do**
      Sample continuation $\tau_f'^{(j)} \sim \pi_\theta(\cdot \mid x_f)$
      $\tilde{\tau}_f^{(j)} \leftarrow [\tau_{1:f}; \tau_f'^{(j)}]$
      $\mathcal{T} \leftarrow \mathcal{T} \cup \{\tilde{\tau}_f^{(j)}\}$
    end for
  end for
end for

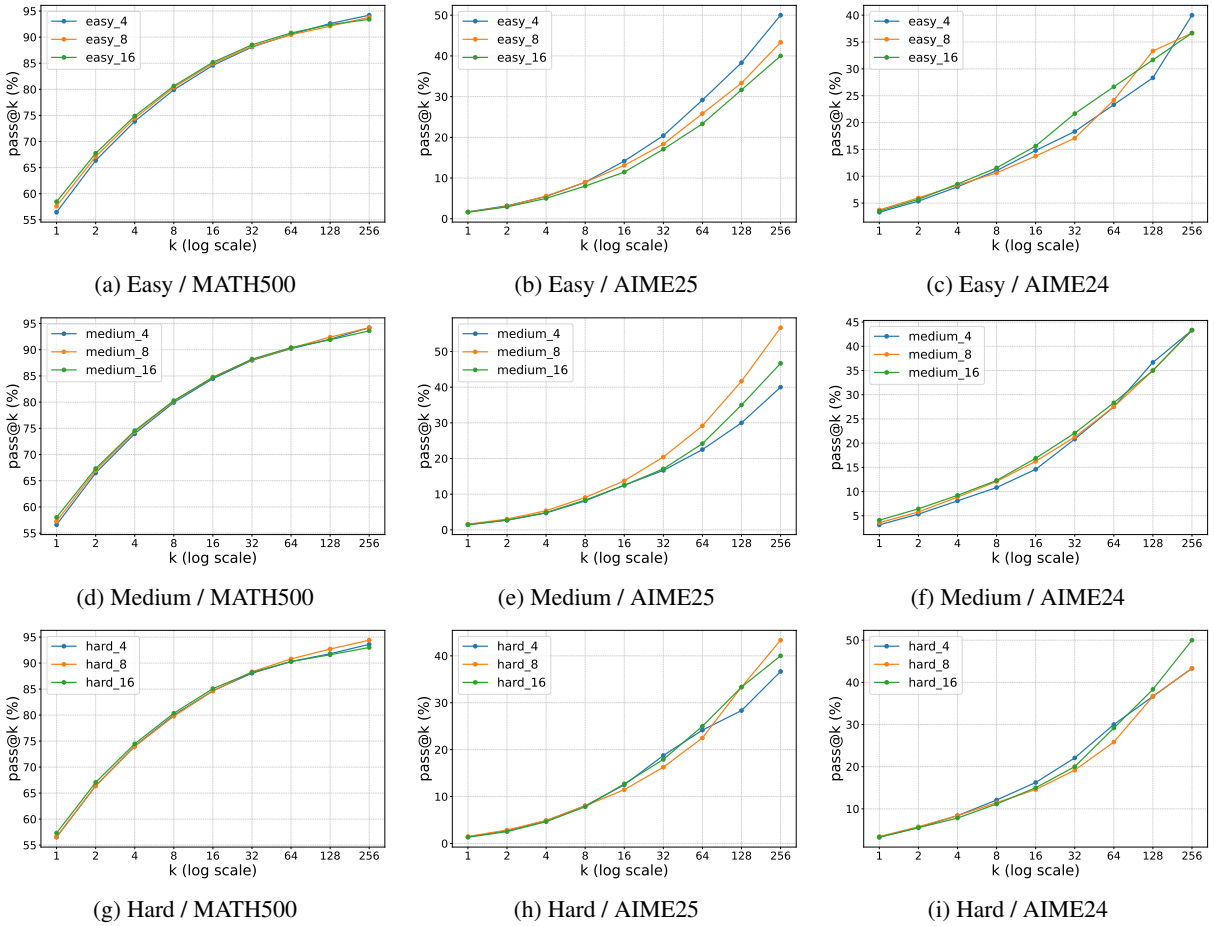
C Details on Tree-Structured Rollout (Section 2.2)

C.1 Tree Algorithm Details

Algorithm 1 provides the detailed procedure of the two-phase tree rollout strategy.

C.2 Experimental Details of Section 2.2.2

To evaluate cost-efficiency during inference, we measure the PassRate—the probability of discovering at least one correct solution—against token consumption on the AIME25 benchmark using Qwen2.5-3B-Instruct. We compare standard parallel sampling with two tree-structured variants: `fixed-seg` (equidistant branching) and `random` (uniform random branching). Detailed procedures for these forking strategies are provided in Appendix D.1.



(a) Easy / MATH500 (b) Easy / AIME25 (c) Easy / AIME24

(d) Medium / MATH500 (e) Medium / AIME25 (f) Medium / AIME24

(g) Hard / MATH500 (h) Hard / AIME25 (i) Hard / AIME24

Figure 7: pass@k performance on evaluation benchmarks for models trained on different difficulty subsets. Labels such as `easy_8` denote models trained on the Easy subset with group size $G = 8$.

Formal Definition of PassRate For parallel sampling, the model generates M independent trajectories, $\mathcal{T}_{\text{parallel}} = \{\tau^{(1)}, \ldots, \tau^{(M)}\}$. The PassRate is defined as:

$$\text{PassRate}_{\text{parallel}} = \mathbb{I}\left(\sum_{i=1}^{M} r(\tau^{(i)}) \geq 1\right)$$

where $r(\cdot) \in \{0, 1\}$ is the verifiable binary reward. Token consumption scales linearly with M.

For tree-structured rollouts, generating N base rollouts, K forking points per base, and B branches per forking point yields a set of leaf trajectories $\mathcal{T}_{\text{tree}}$. The total number of terminal paths is $N(1 + KB)$. The PassRate is evaluated over these leaves:

$$\text{PassRate}_{\text{tree}} = \mathbb{I}\left(\sum_{\tau \in \mathcal{T}_{\text{tree}}} r(\tau) \geq 1\right)$$

Unlike parallel sampling, tree rollouts consume tokens sub-linearly relative to the number of terminal paths, as context prefixes up to each forking point are computed only once and shared across branches.

Configurations To analyze PassRate against token consumption (Figure 2), we varied the computational budget for both methods. For parallel sampling, we evaluated independent rollout counts of $M \in \{4, 8, 16, 32\}$. For tree rollouts, we fixed the branching parameters at $K = 2$ and $B = 2$, and scaled the base rollouts $N \in \{2, 4, 8\}$. These configuration tuples (N, K, B) yield 10, 20, and 40 terminal paths, respectively, enabling a direct comparison of exploration efficiency relative to token expenditure. Detailed forking point selection algorithms are provided in Appendix D.1.

D Details on Forking Point Selection Strategies (Section 2.3)

D.1 Forking Point Selection Algorithms

We detail the five forking point selection algorithms in Algorithm 2, which is utilized in Phase 2 Branch Rollouts. Let a trajectory be denoted as a sequence of tokens $\tau = (y_1, y_2, \ldots, y_{|\tau|})$. For entropy-based methods, let H_t denote the token-level entropy

Algorithm 2 Forking Points Selection Algorithms

Input: Trajectory τ, Number of forking points K, Selection `method`, Token entropies $\{H_t\}_{t=1}^{|\tau|}$, Parsed sentences $\mathcal{S} = \{S_1, \ldots, S_M\}$, Sentence entropies $\{H_m^S\}_{m=1}^{M}$, Parsed steps $\mathcal{Z} = \{Z_1, \ldots, Z_L\}$, Step-level FCI scores $\{y_l\}_{l=1}^{L}$.
Output: Forking points $\mathcal{F} \subset \{1, \ldots, |\tau|\}$, where $|\mathcal{F}| = K$

if `method` is `random` **then**
  Sample $\mathcal{F} \subset \{1, \ldots, |\tau|\}$ uniformly at random
else if `method` is `fixed-seg` **then**
  $\mathcal{F} \leftarrow \left\{ \left\lfloor k \cdot \frac{|\tau|}{K+1} \right\rfloor \;\middle|\; k = 1, \ldots, K \right\}$
else if `method` is `ATB` **then**
  $\mathcal{C} \leftarrow \{l \in \{1, \ldots, L\} \mid y_l \geq \text{Quantile}(y_1, \ldots, y_L, 0.8)\}$
  $\mathcal{L}^* \leftarrow$ the K smallest indices (earliest steps) from $\mathcal{C}$
  $\mathcal{F} \leftarrow \{\nu_l \mid l \in \mathcal{L}^*\}$
else if `method` is `tok-entropy` **then**
  $\mathcal{F} \leftarrow \underset{t \in \{1,\ldots,|\tau|\}}{\arg\text{top}-K}\, H_t$
else if `method` is `sent-entropy` **then**
  $\mathcal{M}^* \leftarrow \underset{m \in \{1,\ldots,M\}}{\arg\text{top}-K}\, H_m^S$
  $\mathcal{F} \leftarrow \{\mu_m \mid m \in \mathcal{M}^*\}$
end if
return $\mathcal{F}$

at step t, which is computed using the language model's probability distribution over the vocabulary V:

$$H_t = -\sum_{v \in V} P(v \mid y_{<t}) \log P(v \mid y_{<t})$$

For the sentence-level method, we parse the trajectory into M sentences, $\mathcal{S} = \{S_1, S_2, \ldots, S_M\}$, by aligning character-level sentence boundaries detected via PySBD (Sadvilkar and Neumann, 2020) with the tokenizer's offset mapping. Each sentence S_m represents a contiguous span of token indices, and μ_m denotes the start token index of S_m. The sentence entropy H_m^S is defined as the average token entropy within that sentence to mitigate length bias:

$$H_m^S = \frac{1}{|S_m|} \sum_{t \in S_m} H_t$$

Attention-based Tree Branching (`ATB`) operates on the premise that attention scores serve as meaningful metrics to identify important reasoning behaviors. To leverage this insight, the trajectory is first segmented into L discrete steps (e.g., delimited by line breaks), denoted as $\mathcal{Z} = \{Z_1, \ldots, Z_L\}$, where ν_l represents the start token index of step Z_l. ATB then computes a Forward Context Influence (FCI) score y_l for each step, quantifying its influence on subsequent tokens by aggregating attention weights. When selecting forking points, ATB first isolates a candidate set consisting of the top 20% of steps with the highest FCI scores. From this set, it selects the K earliest steps as the final forking points.

Lastly, the operator $\arg\text{top}-K$ denotes the extraction of the subset of indices that correspond to the K highest values of a given metric.

Metric	`random`	`sent-entropy`	`tok-entropy`
NND ↓	0.157	0.130	**0.106**
MPD ↓	0.337	0.296	**0.251**
WCR@5 ↑	8.9	12.9	**23.6**
WCR@10 ↑	16.4	25.3	**35.3**

Table 6: Quantitative comparison of forking-point localization across different selection strategies. Arrows indicate stronger localization.

D.2 Quantitative and Qualitative Analysis of Forking-Point Localization

As illustrated in Figure 8, high-entropy tokens tend to densely cluster within narrow, highly uncertain segments of the reasoning trajectory. Consequently, the search budget is monopolized by repeated resampling at a single localized point, which severely limits the structural reach of the search tree.

To quantitatively characterize the localization phenomenon, we measure the distribution of selected forking points using three metrics: Nearest Neighbor Distance (NND), the average distance to the closest forking point; Mean Pairwise Distance (MPD), the average pairwise distance among all selected points; and Window-based Clustering Ratio (WCR@P), the percentage of forking-point pairs whose distance falls within $P\%$ of the trajectory length. Lower NND and MPD and higher WCR indicate stronger localization.

As shown in Table 6, `tok-entropy` exhibits the strongest localization, with the lowest NND (0.106) and MPD (0.251) and the highest WCR@5 (23.6%) and WCR@10 (35.3%). In comparison, `sent-entropy` increases NND and MPD to 0.130 and 0.296, respectively, while reducing WCR@5 and WCR@10 to 12.9% and 25.3%, quantitatively demonstrating that sentence-level entropy mitigates

the localization of forking points.

To further observe how the selected tokens differ across forking strategies, we conducted a case study comparing `tok-entropy`, `sent-entropy`, and `ATB`. Figure 9 visualizes the forking points ($K = 4$) selected by each method along a single, fixed reasoning path. The `tok-entropy` method suffers from localization, trapping the selected points within a narrow segment. In contrast, `sent-entropy` still targets the model's uncertain regions but distributes the forking points much more evenly, operating at a broader semantic level. Meanwhile, `ATB` selects steps receiving the highest attention weights; while this can be meaningful for dividing logical reasoning steps, it operates independently of the model's uncertainty. As shown in Figure 9, `ATB` often selects highly deterministic points to branch from. Performing additional sampling at these points yields low SibDiv and, consequently, a limited PassRate, as reflected in Table 1. Based on these observations, `sent-entropy` proves to be the most effective forking strategy by successfully leveraging uncertainty signals while resolving the localization issue.

D.3 Formal Definitions of SibDiv

Because recent studies emphasize semantic diversity for effective exploration (Jiang et al., 2025; Yao et al., 2025), it is critical to rigorously quantify the divergence of the explored reasoning paths discussed in Section 2.3.2. To this end, we model the generated reasoning tree as a collection of text segments, referred to as *blocks*. Formally, a *block* is defined as a contiguous sequence of reasoning spanning from the root to the first forking point, between two consecutive forking points, or from a final forking point to a leaf.

Let $\mathcal{F}$ denote the set of all forking points and $\mathcal{B}$ denote the set of all blocks within a single generated reasoning tree. We extract a dense semantic representation $\mathbf{e}_b$ for each block $b \in \mathcal{B}$ utilizing the `gte-large-en-v1.5` model (Zhang et al., 2024). Sibling Diversity (SibDiv) is formulated as the complement of the average pairwise cosine similarity among sibling blocks:

$$\textit{SibDiv} = 1 - \frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} \frac{\sum_{b_i, b_j \in B(f),\, i<j} \cos(\mathbf{e}_{b_i}, \mathbf{e}_{b_j})}{\binom{|B(f)|}{2}} \tag{6}$$

where $B(f) \subset \mathcal{B}$ represents the specific subset of child blocks branching directly from a given forking point f, and $\cos(\cdot, \cdot)$ denotes the cosine similarity function.

A higher SibDiv score indicates that the model successfully generates semantically distinct continuations from its forking points. By aggregating this localized diversity, SibDiv effectively captures the degree of semantic exploration and reasoning diversity across the entire tree.

Method	PassRate	SibDiv
`random`	10.0	0.0796
`fixed-seg`	13.3	0.0853
`ATB`	14.7	0.0874
`tok-entropy`	12.0	**0.1065**
`tok-entropy` w/ 5% dist.	14.0	0.1041
`tok-entropy` w/ 10% dist.	<u>15.3</u>	0.1007
`sent-entropy`	**17.3**	<u>0.1049</u>

Table 7: Comparison of forking-point selection strategies, including distance-forced token-entropy baselines.

D.4 Experimental Details of Section 2.3.3

We evaluated each forking strategy by generating reasoning trees under a fixed inference budget, measuring SibDiv and PassRate. Specifically, we conducted inference on the AIME26 benchmark using the Qwen2.5-3B-Instruct model. To isolate the effect of the forking point selection strategy from the inherent randomness of language model generation, we generated and fixed a single base rollout for each problem. We then applied each selection method to identify $K = 4$ distinct forking points along this fixed base trajectory. From each identified forking point, we generated $B = 4$ additional branch rollouts. This procedure ensures a strictly controlled environment, yielding exactly 17 terminal leaves (1 base + 16 new branches) per problem across all methods. Finally, we calculated the PassRate and SibDiv across these 17 leaves and averaged the metrics over the entire dataset. To ensure the robustness of our findings, we repeated this entire experimental procedure five independent times and report the final averaged results in Table 1.

D.5 Analyzing the Benefits of Sentence-Level Entropy

While `sent-entropy` substantially improves PassRate over `tok-entropy`, it remains unclear whether this gain arises simply from mitigating localization or from identifying more semantically meaningful decision points. To disentangle these effects, we introduce distance-forced `tok-entropy`



Figure 8: Visualization of token-entropy *localization*. Dark red boxed tokens indicate the selected top-4 high-entropy forking points. These points densely cluster within a narrow segment, monopolizing the search budget and limiting structural diversity.



Figure 9: Case studies comparing forking point selections across `tok-entropy`, `sent-entropy`, and ATB. Colored boxes indicate the tokens selected by each method, with overlapping boxes denoting agreement between methods. Notably, while `tok-entropy` suffers from localization, `sent-entropy` distributes points more evenly across broader semantic levels, whereas ATB often targets deterministic points regardless of uncertainty.

baselines that preserve the original token-level entropy signal while explicitly preventing nearby forking points. Starting from candidates ranked by token entropy, we greedily select forking points and skip any candidate whose distance from an already selected point is within a predefined fraction of the total trajectory length. We consider minimum-distance thresholds of 5% and 10%.

As shown in Table 7, explicitly enforcing a minimum distance between token-level forking points improves PassRate from 12.0 for standard `tok-entropy` to 14.0 and 15.3 with the 5% and 10% constraints, respectively. This confirms that localization itself is a limiting factor for token-level entropy-based branching. However, both distance-forced variants still underperform `sent-entropy`, which achieves a PassRate of 17.3. Since these variants mitigate localization while retaining the same token-level entropy signal, the remaining gap suggests that the benefit of `sent-entropy` cannot be attributed solely to distributing forking points more broadly. Rather, sentence-level entropy provides a more effective signal for identifying semantically meaningful decision points for branching.

E Algorithmic and Formal Details of DATPO

This section provides the formal definitions of the advantage components and the complete training procedure for Difficulty-Adaptive Sentence-entropy Guided Tree-Structured Policy Optimization (DATPO), expanding upon the methodology outlined in Section 3.

E.1 Formal Definitions of Value Estimation and Sibling Diversity term

Monte Carlo State Value ($\hat{V}_{MC}$) For a state s corresponding to a specific forking point, let $\mathcal{T}(s)$ denote the set of all terminal blocks descending from s. The Monte Carlo state value is formally defined as the expected verifiable reward over these terminal paths:

$$\hat{V}_{MC}(s) = \frac{1}{|\mathcal{T}(s)|} \sum_{\tau \in \mathcal{T}(s)} r(\tau). \tag{7}$$

For a terminal state s_{end}, the set of descending paths is empty, strictly enforcing $\hat{V}_{MC}(s_{\text{end}}) = 0$.

Block-Level Sibling Diversity ($\text{Div}_{\text{sib}}(b)$) Let $P(b)$ denote the forking point from which block b originates, and let $\mathcal{B}_{\text{sib}}(b)$ denote the set of all sibling blocks branching from $P(b)$. The sibling-diversity term for a specific block b is computed as its average semantic distance to its siblings:

$$\text{Div}_{\text{sib}}(b) = 1 - \frac{\sum_{b' \in \mathcal{B}_{\text{sib}}(b) \setminus \{b\}} \cos(\mathbf{e}_b, \mathbf{e}_{b'})}{|\mathcal{B}_{\text{sib}}(b)| - 1}, \tag{8}$$

where $\mathbf{e}_b$ is the embedding of block b. We define $\text{Div}_{\text{sib}}(b) = 0$ when b has no valid sibling block.

While the evaluation metric *SibDiv* (defined in Appendix D.3) aggregates pairwise similarities across *all* forking points to report a single tree-level scalar, the reward term $\text{Div}_{\text{sib}}(b)$ is evaluated at the individual block level to provide the advantage augmentation.

E.2 DATPO Objective Function

DATPO employs a token-level policy gradient objective mapped across the tree topology. Given a set of generated blocks $\mathcal{B}$ for a prompt q, the objective is defined as:

$$\mathcal{J}_{\text{DATPO}}(\theta) = \mathbb{E}_{q \sim \mathcal{Q}, \mathcal{B} \sim \pi_{\theta_{\text{old}}}} \Bigg[\frac{1}{\sum_{b \in \mathcal{B}} |b|} \sum_{b \in \mathcal{B}} \sum_{t=1}^{|b|} \min \Big(\rho_{b,t}(\theta) \hat{A}(b), \text{clip}\big(\rho_{b,t}(\theta), 1-\epsilon, 1+\epsilon\big) \hat{A}(b) \Big) \Bigg], \tag{9}$$

where $|b|$ is the sequence length of block b, $\rho_{b,t}(\theta) = \frac{\pi_\theta(y_t|x_{<t})}{\pi_{\theta_{\text{old}}}(y_t|x_{<t})}$ is the importance ratio, and $\hat{A}(b)$ is the block-level augmented advantage.

E.3 Complete DATPO Algorithm

Algorithm 3 details the end-to-end training procedure. By extending the two-phase rollout strategy into the training framework, the process is structured into four distinct phases: base rollouts, adaptive branch rollouts, block-level advantage estimation, and policy update.

E.4 Comparison with Other Tree-based Methods

To contextualize DATPO within the landscape of recent tree-based RLVR methodologies, we highlight two fundamental distinctions between our approach and existing frameworks such as TreeRL (Hou et al., 2025) and AttnRL (Liu et al., 2025a).

Algorithm 3 DATPO

Input: Initial policy π_θ, Prompts dataset $\mathcal{Q}$, Maximum number of forking points $K_{\max}$, Maximum number of branch rollouts per forking point $B_{\max}$, Base rollouts N, Diversity coefficient α.
Output: Optimized policy π_θ^*
Sample prompt $q \sim \mathcal{Q}$
Phase 1: Base Rollouts
Generate N independent base trajectories $\mathcal{R} = \{\tau^{(1)}, \ldots, \tau^{(N)}\}$ via $\pi_\theta(\cdot \mid q)$
Compute $V(\text{root}) = \frac{1}{N}\sum_{i=1}^{N} r(\tau^{(i)})$
Compute adaptive budgets: $\hat{K} = \lceil K_{\max}(1 - V(\text{root}))\rceil$, $\hat{B} = \lceil B_{\max}(1 - V(\text{root}))\rceil$
Phase 2: Adaptive Branch Rollouts
Initialize tree path set $\mathcal{T} \leftarrow \mathcal{R}$
if $\hat{K} > 0$ **and** $\hat{B} > 0$ **then**
  for each $\tau \in \mathcal{R}$ **do**
    Select $\hat{K}$ forking points $\mathcal{F}$ using `sent-entropy` forking strategy (Algorithm 2)
    for each $f \in \mathcal{F}$ **do**
      Extract trajectory prefix $x_f \leftarrow [q; \tau_{1:f}]$
      Generate $\hat{B}$ branch continuations from prefix x_f
      Add generated branches to $\mathcal{T}$
    end for
  end for
end if
Phase 3: Block-level Advantage Estimation
Partition $\mathcal{T}$ into a set of contiguous blocks $\mathcal{B}$
for each block $b \in \mathcal{B}$ **do**
  Compute $\hat{V}_{MC}(s_{\text{start}}^{(b)})$ and $\hat{V}_{MC}(s_{\text{end}}^{(b)})$
  $\hat{A}_{\text{base}}(b) = r(b) + \hat{V}_{MC}(s_{\text{end}}^{(b)}) - \hat{V}_{MC}(s_{\text{start}}^{(b)})$
  Compute $\text{Div}_{\text{sib}}(b)$ using block embeddings (Equation 8)
  $\hat{A}(b) = \hat{A}_{\text{base}}(b) + \mathbb{I}(\hat{A}_{\text{base}}(b) > 0) \cdot \alpha \cdot \text{Div}_{\text{sib}}(b)$
end for
Phase 4: Policy Update
Update θ by maximizing $\mathcal{J}_{\text{DATPO}}(\theta)$ (Equation 9) using advantages $\{\hat{A}(b)\}_{b\in\mathcal{B}}$
Anneal α according to schedule
return Optimized policy π_θ^*

Structural Robustness in Optimization Both TreeRL and AttnRL generate tree-structured rollouts but ultimately flatten these trees into a set of $N(1 + K \times B)$ independent sequences to perform sequence-level policy updates (Hou et al., 2025). A critical flaw in this unfolding process is that common trajectory prefixes are duplicated across multiple sequences, causing the policy to redundantly update on the same early tokens. To mitigate the resulting overfitting, these methods rely on a heuristic penalty, artificially downweighting the advantage of non-leaf steps by dividing it by the square root of the descending leaf count ($\sqrt{|L(s_n)|}$) (Hou et al., 2025; Liu et al., 2025a).

In contrast, DATPO is structurally immune to this redundancy. By strictly partitioning the generated tree into non-overlapping contiguous blocks, every token within the tree belongs to exactly one block. Because DATPO performs explicit block-level updates directly over the tree topology, it naturally prevents duplicate gradient updates on shared prefixes, providing a highly robust credit assignment mechanism without the need for ad-hoc advantage penalization.

Motivation for Difficulty-Adaptive Exploration While AttnRL (Liu et al., 2025a) similarly incorporates a difficulty-adaptive exploration mechanism, its approach is fundamentally driven by computational efficiency. Specifically, AttnRL reduces the sampling budget for easy problems to prevent inefficient exploration and filter out responses with zero advantages.

Conversely, DATPO's difficulty-adaptive rollout originates from a completely different motivation: maximizing intrinsic reasoning coverage (pass@k). As demonstrated in Section 2.1, adaptive budget allocation is not merely a resource-saving heuristic, but a key algorithmic factor for successfully expanding a model's pass@k without inducing mode collapse. Built entirely around this objective, DATPO further maximizes coverage through the structural superiority of trees (Section 2.2) and optimal semantic forking (Section 2.3). Thus, while DATPO and AttnRL share superficial similarities in their adaptive nature, they are built upon fundamentally distinct motivations and optimization goals.

F Details on Main Experiments (Section 4)

F.1 Experimental Setups

Models and Datasets We use Qwen2.5-3B-Base (Yang et al., 2024) and Qwen3-4B-Base (Yang et al., 2025) as base models. For training, we use the MATH dataset (Hendrycks et al., 2021). From the full set of 12,500 problems, we exclude the

500 problems used as the MATH500 and use the remaining 12,000 problems as the training set.

Evaluation We evaluate the trained models on mathematical reasoning benchmarks, specifically MATH500 (Hendrycks et al., 2021), AIME26, AIME25, AIME24, and AMC23. We report avg@k as the primary evaluation metric across all benchmarks. All evaluations were performed with a temperature of 1.0 and a maximum generation length of 2048. To assess the reasoning coverage and test-time scaling performance of the trained models, we additionally report pass@k and maj@k (majority voting), respectively. We use $k = 8$ for MATH500, and $k = 64$ for the remaining benchmarks. For robustness, both pass@k and maj@k results are computed by averaging over three independent evaluation runs for each dataset. The final maj@k in Figure 6 is then reported as the average of these results across all mathematical reasoning benchmarks.

Baselines We compare DATPO with several baselines: (1) **Base**, the base model without any additional training; (2) **GRPO** (Shao et al., 2024), standard GRPO enhanced with the token-level loss and clip-higher strategy from DAPO (Yu et al., 2025b). Following prior work (Liao et al., 2025) showing that dynamic sampling can be detrimental for relatively weak reasoning models, we intentionally adopt only this specific subset of DAPO components; (3) **Dr.GRPO** (Liu et al., 2025b), a refined variant of GRPO that corrects structural biases in advantage estimation; (4) **TreeRL** (Hou et al., 2025), a tree-based RL method that selects forking points from high-entropy tokens and estimates process-level advantages by combining local and global advantages; and (5) **AttnRL** (Liu et al., 2025a), another tree-based RL method built upon TreeRL with a focus on improving efficiency by selecting forking points based on attention scores and adopting adaptive sampling and one-step off-policy. For TreeRL and AttnRL, we adopt the same tree-expansion hyperparameters as reported in original papers. To ensure a fair comparison, we control the total number of generated tokens per problem to be comparable across all methods (detailed in Appendix F.3).

F.2 Implementation Details

Sampling and Algorithmic Hyperparameters Across all methods, we set the sampling temperature to 1.0 and the maximum generation length to 1024 tokens. For **GRPO** and **Dr.GRPO**, we sample 16 responses per problem. For the tree-based baselines, we strictly follow the hyperparameters reported in their respective original papers. Specifically, for **TreeRL**, we set the tree expansion parameters to $(N, K, B) = (6, 2, 2)$. For **AttnRL**, we use maximum budgets of $(N, K_{\max}, B_{\max}) = (6, 2, 2)$, select the top 20% of FCI steps as forking candidates, and apply $\Delta = 4$ and $\lambda = 0.9$ for adaptive batching. For our proposed **DATPO**, we set the adaptive tree parameters to $(N, K_{\max}, B_{\max}) = (4, 3, 4)$. We intentionally employ a larger maximum branch count ($B_{\max} = 4$) compared to the baselines, as calculating the sibling diversity term (Div_{sib}) across merely two branches ($B = 2$) diminishes its significance. To accommodate this wider branching while keeping the total number of generated tokens per problem comparable to the baseline methods, we adjusted the base and forking budgets accordingly. The diversity coefficient α is initialized to 0.2 and linearly annealed to 0 over the course of training. To compute the semantic diversity representations, we utilize the `gte-large-en-v1.5` embedding model (Zhang et al., 2024). Furthermore, to prevent Out-Of-Memory (OOM) errors caused by the expansion of adaptive methods (AttnRL and DATPO), we apply rollout chunking during the branch rollout phase. We generate branch rollouts in chunk sizes of 384 for Qwen2.5-3B-Base and 64 for Qwen3-4B-Base.

Method	Avg. Generated Tokens
GRPO	8,755.7
Dr.GRPO	8,608.3
TreeRL	11,029.9
AttnRL	8,807.3
DATPO (Ours)	8,952.4

Table 8: Average number of generated tokens per question during training across different methods.

Training Hyperparameters We formulate the task with a binary verifiable reward, assigning a reward of 1 if the final answer is correct and 0 otherwise. Models are optimized using the AdamW optimizer with $\beta = (0.9, 0.999)$ and a constant learning rate of 5×10^{-6}. We omit the KL divergence penalty and the clipping ratio is set to 0.2. For the GRPO baseline enhanced with the clip-higher strategy, we set the upper clip ratio to

Method	Wall Clock (Hours)
Parallel Rollout	
GRPO	26.3
Dr.GRPO	27.1
Tree-based Rollout	
TreeRL	56.2
AttnRL	55.5
DATPO w/o diversity	56.0
DATPO	60.3
DATPO w/ one-step off-policy	60.0

Table 9: Wall-clock comparison across different methods.

0.28 and the lower to 0.2. During training, we update the policy using 16 prompts per step, maintaining a global mini-batch size of 64 and a micro-batch size of 2 for gradient accumulation across all methods. The total effective batch size varies by method: GRPO and Dr.GRPO utilize a fixed batch size of 256; TreeRL employs a batch size of 480 (comprising 96 base rollouts and 384 branch rollouts); whereas the batch sizes for AttnRL and DATPO dynamically adjust per step due to their adaptive nature. Notably, because DATPO performs block-level policy updates, its mini-batch and micro-batch sizes are defined with respect to the number of contiguous blocks rather than full sequences. All experiments were executed on a computing cluster equipped with 8 NVIDIA A100 GPUs, where each individual training run was conducted on a single A100 GPU. Our codebase is developed based on the GRPO-Zero framework[3].

F.3 Computational Cost and Wall-Clock Time

To ensure a fair comparison, we control the total number of generated tokens per problem to be comparable across all methods. The actual average number of generated tokens per question during training for each method is detailed in Table 8. However, despite this token-level equivalence, tree-based methods generally incur a higher wall-clock time compared to standard parallel rollouts, as reported in Table 9. This discrepancy arises from the inherent sequential dependency in tree generation: branch rollouts cannot commence until the base rollouts are fully generated and forking points are explicitly selected, creating a computational bottleneck. Furthermore, our proposed DATPO requires additional overhead to compute block-level embeddings for the diversity term, slightly increasing its wall-clock time relative to other tree-based baselines. We note that adopting the one-step off-policy approach proposed in AttnRL (Liu et al., 2025a) can marginally mitigate this latency. Table 10 provides a detailed breakdown of the per-step runtime of DATPO by computational component.

[3] https://github.com/policy-gradient/GRPO-Zero

Component	Time (s)	Total runtime (%)
Policy	150.44	50.14
Diversity embedding	13.65	4.55
Update	102.17	34.05
Others	33.76	11.25
Total	**300.02**	**100.00**

Table 10: Per-step runtime breakdown of DATPO across different computational components.

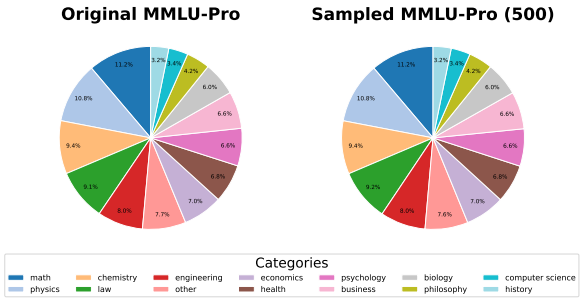


Figure 10: Comparison of category-wise distributions between the original MMLU-Pro dataset and our sampled subset (500 instances). Stratified sampling ensures the original proportions are preserved.

F.4 Generalization to Out-Of-Domain Tasks

To investigate whether the models trained on mathematical reasoning datasets can lead to improvements in reasoning performance across other domains, we additionally conduct out-of-domain evaluations using GPQA-Diamond (Rein et al., 2023) and MMLU-Pro (Wang et al., 2024). For MMLU-Pro, we applied stratified sampling to evaluate on a subset of 500 instances, preserving the original proportional distribution across categories such as math, physics, chemistry, engineering, law, and economics. A comparison of the category-wise distributions between the original and sampled datasets is provided in Figure 10. We evaluate the models trained with each method using two base models, Qwen2.5-3B-Base and Qwen3-4B-Base, and report the avg@8 metric for both benchmarks.

Table 11 presents the results of the out-of-domain evaluation. Overall, DATPO demonstrates consistent performance across both the GPQA-Diamond and MMLU-Pro datasets. Specifically, compared to the strongest baselines, DATPO achieves a +2.4 percentage point improvement on

Method	GPQA-Diamond	MMLU-Pro
Qwen2.5-3B-Base		
Base	21.1	18.4
GRPO	25.3	32.6
Dr.GRPO	26.1	31.2
TreeRL	25.3	33.1
AttnRL	25.9	**33.2**
DATPO (Ours)	**28.5**	33.1
Qwen3-4B-Base		
Base	24.2	34.8
GRPO	35.9	55.9
Dr.GRPO	38.2	57.3
TreeRL	36.7	55.6
AttnRL	38.4	56.6
DATPO (Ours)	**39.6**	**57.7**

Table 11: Out-of-domain evaluation results. We report avg@8 accuracy for each dataset.

GPQA-Diamond and a marginal -0.1 point difference on MMLU-Pro using the Qwen2.5-3B-Base model. When applied to the Qwen3-4B-Base model, it yields consistent improvements of +1.2 and +0.4 percentage points on the respective benchmarks. These results demonstrate that the enhancements in reasoning capacity observed on in-domain mathematical benchmarks successfully generalize to improved reasoning performance on out-of-domain tasks.

F.5 Further Ablation Studies

Effect of Difficulty-Adaptive Rollout While Section 4.2 indirectly demonstrates the benefits of our approach against non-adaptive baselines (e.g., TreeRL), we conduct this ablation to explicitly isolate the impact of the difficulty-adaptive mechanism by evaluating a static, non-adaptive variant of DATPO. Applying our default tree-expansion parameters of $(N, K_{\max}, B_{\max}) = (4, 3, 4)$ to a static rollout would significantly increase the overall training budget. Therefore, to maintain a token consumption comparable to our adaptive framework, we restrict the tree-expansion parameters of this non-adaptive variant to a narrower configuration of $(N, K, B) = (6, 2, 2)$.

As reported in Table 12, the full DATPO framework outperforms the non-adaptive variant in both avg@k and pass@k. The lower pass@k of the non-adaptive model aligns with our empirical analysis in Section 2.1: uniformly expanding the rollout budget across all problems, regardless of their difficulty, can lead to a decrease in pass@k. Furthermore, the absence of difficulty adaptation also degrades avg@k. We attribute this to a compounding effect during training: the restricted reasoning coverage artificially limits the variety of valid reasoning paths discovered, which consequently undermines the effectiveness of the annealed sibling-diversity term that relies on a rich, diverse set of trajectories to properly optimize the policy.

Benchmark	DATPO w/o difficulty-adaptive	DATPO
MATH500	62.9 / 81.0	63.5 / 81.7
AIME26	2.6 / 21.1	2.4 / 33.3
AIME25	1.7 / 25.6	1.6 / 33.3
AIME24	5.4 / 31.1	4.8 / 35.6
AMC23	37.0 / 90.0	39.8 / 90.8
Average	21.9 / 49.8	**22.4 / 54.9**

Table 12: Ablation study on difficulty-adaptive rollout. Results are reported as avg@8 / pass@8 for MATH500 and avg@64 / pass@64 for all other benchmarks.

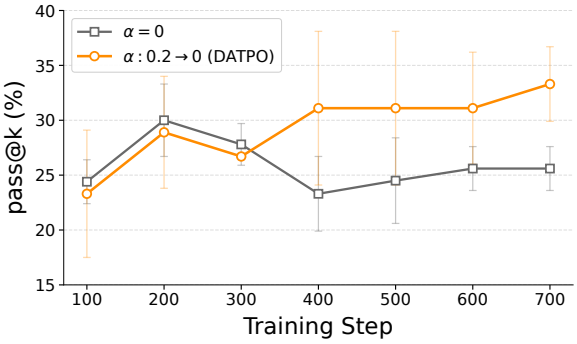


Figure 11: pass@64 on AIME26 across training checkpoints under different diversity-coefficient schedules.

Training Dynamics of the Diversity-Augmented Advantage. Table 4 reports the effect of the diversity coefficient only at the final checkpoint. To examine how the diversity-augmented advantage affects reasoning coverage throughout training, we additionally evaluate pass@64 on AIME26 at 100-step intervals. We compare training without the diversity term ($\alpha = 0$) against DATPO with the default linearly annealed schedule ($\alpha : 0.2 \rightarrow 0$). Each result is averaged over three independent evaluation runs and reported as the mean $\pm$ standard deviation.

As shown in Figure 11, without the diversity term, pass@64 peaks at 30.0% at step 200 and decreases to 25.6% by step 700. In contrast, DATPO exhibits an overall upward trend after the initial training stage and reaches 33.3% at step 700. These results suggest that the diversity-augmented advantage mitigates the early saturation of reasoning cov-

Embedding	Size	MATH500	
		avg@8	pass@8
`all-mpnet-base-v2`	110M	62.9	81.9
`bge-m3`	560M	61.6	81.7
`gte-base-en-v1.5`	137M	63.4	81.9
`gte-large-en-v1.5`	434M	63.5	81.7

Table 13: Ablation study on different embedding models.

Method	Avg. Tokens	MATH500	
		avg@8	pass@8
TreeRL (4,3,4)	17,766.0	62.5	81.3
AttnRL (4,3,4)	12,478.7	62.1	80.0
DATPO	8,952.4	63.5	81.7

Table 14: Performance of baselines under wider tree topology. Avg. Tokens denotes the average number of generated tokens per question during training. The values in parentheses specify the tree expansion hyperparameters (N, K, B).

erage and promotes its continued expansion during training.

Effect of Embedding Models As noted in Appendix F.2, DATPO defaults to using `gte-large-en-v1.5` to compute the block-level sibling-diversity term (Div_{sib}). To assess the sensitivity of our framework to the choice of the embedding model, we conduct an additional ablation study comparing it against three alternative models of varying scales: `all-mpnet-base-v2` (Reimers and Gurevych, 2019), `bge-m3` (Chen et al., 2024), and `gte-base-en-v1.5` (Zhang et al., 2024).

As reported in Table 13, while the 434M-parameter `gte-large-en-v1.5` model achieves the highest avg@8 accuracy (63.5%), smaller architectures such as `all-mpnet-base-v2` (110M) and `gte-base-en-v1.5` (137M) also yield highly competitive performance, even marginally outperforming the default configuration in pass@8 (81.9%). These results demonstrate that DATPO is robust to the scale and specific choice of the underlying embedding model. Consequently, the effectiveness of our diversity-augmented advantage appears to stem primarily from capturing stable, relative semantic distance signals among sibling blocks, rather than relying strictly on the absolute capacity or size of the embedding model itself.

F.6 Performance of Baselines under Wider Tree Topology

As detailed in Appendix F.2, the tree-based baseline methods (TreeRL and AttnRL) were evaluated using the default tree expansion hyperparameters reported in their original papers, specifically $(N, K, B) = (6, 2, 2)$ for TreeRL and $(N, K_{\max}, B_{\max}) = (6, 2, 2)$ for AttnRL. In contrast, our proposed DATPO utilized a wider branching configuration of $(N, K_{\max}, B_{\max}) = (4, 3, 4)$ to ensure a sufficient number of branch rollouts for computing the sibling-diversity term. Although we rigorously constrained the total number of generated tokens to be comparable across all methods in our main experiments (Appendix F.3), one might argue that DATPO's performance gains simply stem from this wider tree topology ($B = 4$) rather than its algorithmic design.

To isolate the impact of our algorithmic design, we conduct an additional experiment where both TreeRL and AttnRL are trained using the identical wider configuration of $(4, 3, 4)$ (or $(4, 3, 4)$ maximum limits for AttnRL).

As reported in Table 14, forcing the baselines into this wider topology does not improve their overall performance. We observe three key findings: First, our proposed DATPO achieves the highest performance in both avg@8 and pass@8 while generating the fewest average tokens per problem during training. Second, compared to their original $(6, 2, 2)$ configurations (as reported in Table 2), both TreeRL and AttnRL experience performance degradation in some metrics when scaled to the $(4, 3, 4)$ topology. Taken together, these results confirm that the superiority of our framework is driven by the algorithmic improvements of the difficulty-adaptive rollout and diversity-guided exploration, rather than merely relying on wider tree expansion hyperparameters.

G Further Related Works

G.1 Tree-based Search in Reinforcement Learning

Tree-based search, famously successful in reinforcement learning milestones like AlphaGo (Silver et al., 2016), is now being adapted to the generative landscape of LLMs to unlock more systematic exploration. TreeRL (Hou et al., 2025) pioneers this with an on-policy framework using entropy-guided branching. To address computational bottlenecks, AttnRL (Liu et al., 2025a) improves effi-

ciency by leveraging difficulty-aware adaptive sampling within a one-step off-policy pipeline. Recent research optimizes structural efficiency: TEMPO (Tran et al., 2025) leverages prefix-tree structure for branch-aware credit assignment, while TreePO (Li et al., 2025a) maximizes KV-cache reuse. To prevent paths from collapsing into homogeneous reasoning, LATR (Xing et al., 2025) explicitly maximizes trajectory-level diversity by dynamically branching and pruning semantically redundant paths via lookahead simulation. Furthermore, TreeAdv (Cao et al., 2026) refines credit assignment by combining an entropy-based branching method with a tree-structured advantage redistribution. While these methods primarily utilize tree structures for credit assignment or computational efficiency, our work leverages tree structures with the primary goal of expanding the model's reasoning coverage.

H Use of AI assistants

AI assistants were utilized for code implementation and for refining the linguistic presentation of the manuscript. All AI-generated outputs were carefully reviewed, modified, and validated by the authors.