TREERPO: TREE RELATIVE POLICY OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards trains reasoning models from the correctness of their final answers, but a single outcome provides limited information about which intermediate choices helped or hindered a solution. We introduce TREERPO, which obtains intermediate supervision by sampling a tree of continuations and propagating verified terminal rewards toward the root. Children of the same prefix form local comparison groups, allowing a GRPO-style update to distinguish alternative reasoning segments without a separate process reward model. Group filtering removes nearly uniform reward groups, and a mean-dependent advantage scale preserves differences in estimated continuation quality. On four mathematical reasoning benchmarks, TREERPO raises the macro accuracy of Qwen2.5-Math-1.5B from the GRPO baseline's 32.7% to 35.6%, a 2.9-percentage-point improvement. For Qwen2.5-Math-7B, macro accuracy rises from 40.7% to 41.4%. The 1.5B model also produces shorter evaluation responses, with a reported aggregate length reduction of 18.1%. These results support tree-based, outcome-derived supervision as a practical alternative to assigning one relative advantage to an entire reasoning trajectory.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) improves reasoning by training language models on answers that can be checked automatically. A mathematical solution can be compared with a reference answer, and a program can be evaluated using tests. This form of feedback underlies influential reasoning systems and training recipes (OpenAI, 2024; Guo et al., 2025; Qwen, 2024; Shao et al., 2024). It avoids learning a separate evaluator for every intermediate state, but usually attaches the reward to a completed response. The same final outcome must then guide decisions made at many different points in the solution.

This creates a credit-assignment problem. Two solutions may share a useful prefix and diverge at a later step, yet trajectory-level normalization compares their complete outputs. Conversely, two continuations of the same prefix may have different prospects of reaching the correct answer, even before either has finished. A terminal verifier cannot directly label the local choice. Process reward models address this gap by scoring intermediate steps (Lightman et al., 2023; Wang et al., 2024; Lu et al., 2024a), but introduce an additional model and a process-supervision pipeline. Constructing that supervision is itself difficult when a plausible-looking step leads to an unsuccessful continuation or an unconventional step still permits a correct solution.

Our approach asks a narrower, operational question: after a candidate reasoning segment, how often do sampled continuations obtain a correct final answer? The verifier still judges only completed paths. Multiple continuations turn those terminal outcomes into an estimate of the segment's downstream reward. Repeating this construction at several depths produces a hierarchy of local comparisons. The resulting supervision measures continuation success under the sampling policy, not the intrinsic logical correctness of a sentence.

We introduce TREERPO, a tree-relative variant of group-based policy optimization. A question forms the root of a sampled tree, and each edge contains a bounded-length token segment. Terminal outcomes are propagated upward by averaging child rewards. At each parent, its children form a group of alternatives with exactly the same preceding context. Their relative rewards are used to train the corresponding segments. This changes the unit of comparison from complete answers to alternative continuations of a shared prefix, while retaining the final-answer verifier and clipped policy-update structure of GRPO (Shao et al., 2024).

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

Two design choices accompany this construction. First, groups with almost identical estimated rewards are excluded from the update, because they provide little local ranking information. Second, we scale centered rewards using the group's mean-dependent Bernoulli variance rather than the empirical standard deviation of its fractional rewards. The latter distinction matters because two groups can have the same normalized ranks despite very different reward gaps. These choices govern which local comparisons enter training and how strongly they influence the update; they do not require a learned value or process reward model.

We evaluate the method using Qwen2.5-Math-1.5B and 7B on four mathematical benchmarks. The main table shows an advantage over GRPO on all four benchmarks for the 1.5B model and on three, with one tie, for the 7B model. The 1.5B evaluation responses are also shorter across the four benchmarks. We distinguish this inference-time token saving from training efficiency: tree expansion creates additional continuations, and shorter test responses alone do not demonstrate lower end-to-end training cost.

Our contributions are:

- We formulate outcome-derived intermediate supervision using sampled continuation trees. Bottom-up reward estimates and sibling groups give local learning signals without a separate process reward model.
- We combine group filtering and a mean-dependent advantage scale with a clipped relative-policy objective, and explain how these components interact with fixed-token segmentation and terminal verification.
- We report results for two model sizes on four mathematical benchmarks, together with training curves, evaluation-length comparisons, and a smaller-batch setting. The strongest improvement is observed on the 1.5B model, with a 2.9-point macro-accuracy gain over GRPO.

2 Related Work

Reasoning through prompting and training. Chain-of-thought prompting exposes intermediate reasoning without changing model parameters (Wei et al., 2022). Search and structured prompting extend this idea by considering multiple intermediate states or programs (Yao et al., 2023; Zhang et al., 2023; Bi et al., 2023; Yang et al., 2024b). Supervised approaches instead build question–solution corpora and update the model, as in MetaMath, MathGenie, and DART-Math (Yu et al., 2024; Lu et al., 2024b; Tong et al., 2024). TreeRPO uses a tree during training to construct relative advantages; it does not require tree search to generate each evaluation answer. The tree is therefore a supervision mechanism rather than an inference-time solver.

Outcome and process supervision. Outcome reward models score completed answers (Cobbe et al., 2021; Yu et al., 2023), while process reward models provide intermediate feedback (Lightman et al., 2023; Wang et al., 2024; Lu et al., 2024a). Sampling continuations can inform such process supervision, but training an explicit step evaluator remains a separate design decision. TreeRPO directly uses the sampled terminal rewards in each training batch. It does not learn an evaluator that generalizes these scores to unseen prefixes. Its intermediate estimates consequently depend on the continuations available in the current tree.

Relative policy optimization. PPO uses a clipped policy surrogate (Schulman et al., 2017); GRPO replaces a learned value baseline with comparisons among sampled outputs for the same prompt (Shao et al., 2024). Rule-based reasoning recipes and variants such as DAPO and VAPO further develop this approach (Zeng et al., 2025; Luo et al., 2025b; Yu et al., 2025; Yue et al., 2025; Hu et al., 2025). DPO and Step-DPO instead optimize preference comparisons from constructed data (Rafailov et al., 2023; Lai et al., 2024). TreeRPO retains group-relative learning but forms groups at internal prefixes. The distinction is not the use of a verifier alone, but the way multiple outcomes are reused to compare sibling segments at different depths.

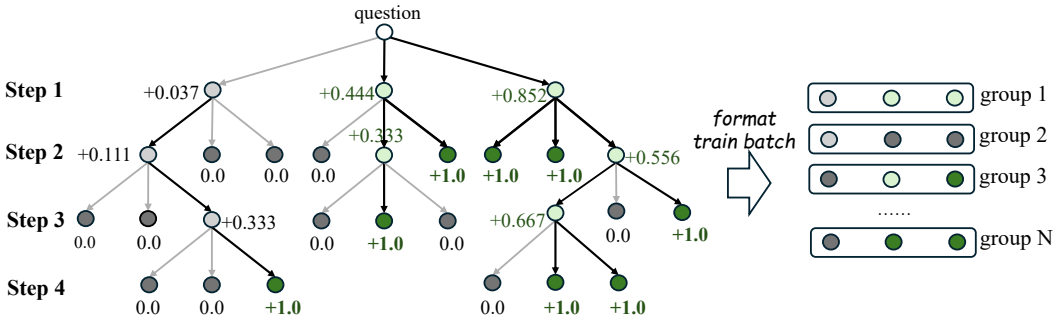

Figure 1: Tree sampling and local group construction. Terminal paths are verified, their rewards are propagated upward, and children sharing a parent form a comparison group. The original illustration uses a schematic branching pattern and four drawn levels; the experimental configuration uses branching factor $N = 8$ and maximum depth $D = 3$.

3 Tree Relative Policy Optimization

For a question q, let π_{old} denote the policy used to collect a training tree. A node v represents a prefix s_v consisting of the question and all segments on its path from the root. A child c extends that prefix with a segment z_c. The verifier ϕ assigns a binary reward to the completed path. We use the same final-answer verification mechanism for GRPO and TreeRPO; intermediate scores are obtained from sampled outcomes, not extra labels.

3.1 Sampling Alternative Continuations

At each expandable node, we sample N continuations from π_{old}. Each continuation contains at most L_{step} tokens. A branch stops when a stopping token is generated or its path reaches depth D; other active branches can continue until they meet their own stopping conditions. This produces a tree with variable realized branch lengths, even though the maximum depth and segment length are fixed. In the experiments, $N = 8$, $D = 3$, and $L_{\text{step}} = 384$, giving a maximum of 1,152 generated tokens along a path.

Segments are token blocks, not annotated mathematical steps. A block may contain several short deductions or end inside a longer explanation. This simple boundary rule permits the same sampling procedure across questions, but the term "step-level" refers to the algorithm's segmentation. It does not imply that the boundaries coincide with complete logical propositions. The state shared by a sibling group nevertheless remains well defined: all its alternatives start from the same token prefix.

Tree sampling changes the allocation of continuations. In flat GRPO sampling, complete responses branch at the question and are compared at that level. Here, a sampled prefix can be extended in several ways, and those alternatives can branch again. The outcome of a terminal path contributes to the estimates of the segments preceding it. Shared prefixes thus support several local comparisons, although the resulting estimates are correlated because they reuse descendant outcomes.

3.2 Propagating Verifiable Outcomes

For a terminal node v, let P_v be the full question–response path. Its reward is

$$R(v) = \phi(P_v), \qquad \phi(P_v) \in \{0, 1\}. \tag{1}$$

For an internal node, we average its children's rewards:

$$R(v) = \frac{1}{|\text{Ch}(v)|} \sum_{c \in \text{Ch}(v)} R(c). \tag{2}$$

Applying Eq. (2) from the leaves to the root yields fractional rewards in $[0, 1]$. They estimate downstream verification success for sampled continuations. A segment with reward 0.75 is therefore not

"75% logically correct"; its descendants have a high average terminal reward under this sampling construction.

This recursion averages locally, with equal weight for each sampled child. In a balanced tree, it equals an average over the descendant leaves. When branches terminate at different depths, local child averaging need not equal an unweighted average over every leaf: a terminal child still occupies one child position at its parent. Stating the recursion explicitly avoids conflating these two aggregations. The method uses the recursively defined reward throughout group formation and advantage computation.

For example, suppose two sibling segments have descendant outcomes $(1, 1, 1, 0)$ and $(1, 0, 0, 0)$. Their rewards are 0.75 and 0.25. Both may lie on at least one successful path, but they receive different estimates of how reliably the current continuation policy finishes the solution. Conversely, a sampled subtree with only failures provides no evidence that its starting segment is necessarily wrong: later decisions may also be responsible. The signal is an empirical continuation value, which is useful for relative training without being a proof of local correctness.

3.3 Filtering Local Groups

Each parent defines a group $\mathcal{G}_v = \mathrm{Ch}(v)$. Its reward range is

$$\Delta R_{\mathcal{G}_v} = \max_{c \in \mathcal{G}_v} R(c) - \min_{c \in \mathcal{G}_v} R(c). \tag{3}$$

We retain the group for training when $\Delta R_{\mathcal{G}_v} > \tau$, using $\tau = 0.1$ in the reported experiments. This is a reward-range threshold, not a variance threshold. Groups whose children all receive the same reward have zero centered advantages; nearly uniform groups offer only a weak estimated distinction. Filtering focuses the update on groups with a nontrivial reward gap, following the motivation of informative-group sampling in DAPO (Yu et al., 2025).

The filtering operation takes place after reward propagation. All descendants needed to compute a group's rewards must already have been sampled and verified. Consequently, this form of pruning reduces the training data passed to the optimizer but should not be interpreted as automatically saving the generation cost of the discarded groups. It also selects informative groups rather than guaranteeing that every retained gradient will improve test accuracy. A nonzero reward range is evidence of a local empirical comparison, not a guarantee about the quality of the update.

3.4 Scaling Fractional-Reward Advantages

GRPO commonly centers each reward and divides by the group's empirical standard deviation. This can remove information about the magnitude of reward differences. With population-standard-deviation normalization, both $(0, 0, 1, 1)$ and $(0.49, 0.49, 0.51, 0.51)$ produce $(-1, -1, 1, 1)$. The two groups have the same ordering, but their estimated success gaps differ substantially.

For a retained group with rewards $(R_1, \ldots, R_n)$, TreeRPO uses

$$\mu = \frac{1}{n} \sum_{i=1}^{n} R_i, \qquad b(\mu) = \mu(1 - \mu), \qquad A_i = \frac{R_i - \mu}{b(\mu)}. \tag{4}$$

We preserve the denominator specified in the original method: $b(\mu)$ is a Bernoulli *variance*, not its square root. For the two example groups, Eq. (4) gives $(-2, -2, 2, 2)$ and $(-0.04, -0.04, 0.04, 0.04)$ before filtering. Thus small reward gaps remain small when the group mean is the same. The second example would also be removed by the default range threshold, illustrating that filtering and scaling address related but distinct aspects of weak reward contrasts.

This rule is not algebraically identical to standard binary GRPO normalization. For binary rewards, dividing by $\mu(1-\mu)$ rather than $\sqrt{\mu(1-\mu)}$ changes the update scale. Filtering ensures that retained groups are nonconstant, so $0 < \mu < 1$ in exact arithmetic. It does not prevent a small denominator near the endpoints. The intended benefit is preserving reward-gap information; the equation alone does not establish an unbiased estimator or a general stability guarantee.

3.5 Clipped Segment-Level Policy Update

For child segment z_i in a retained group with parent prefix s_v, define the importance ratio at token t as

$$u_{i,t}(\theta) = \frac{\pi_\theta(z_{i,t} \mid s_v, z_{i,<t})}{\pi_{\text{old}}(z_{i,t} \mid s_v, z_{i,<t})}. \tag{5}$$

The segment's advantage A_i is shared by its tokens. The clipped relative-policy term is

$$\mathcal{L}_i(\theta) = \frac{1}{|z_i|} \sum_{t=1}^{|z_i|} \min(u_{i,t}(\theta)A_i, \text{clip}(u_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon)A_i)\,. \tag{6}$$

Training averages these terms over sampled retained groups and adds a KL penalty against a reference policy:

$$\mathcal{J}(\theta) = \mathbb{E}_{q,\,\mathcal{G}_v}\left[\frac{1}{|\mathcal{G}_v|}\sum_{i\in\mathcal{G}_v}(\mathcal{L}_i(\theta) - \beta\mathcal{K}_i(\theta))\right], \tag{7}$$

where $\mathcal{K}_i$ denotes the token-averaged KL term on the segment's states. The preceding path provides context; local alternatives are compared only within their common-parent group. This formulation expresses the intended group-relative objective without treating correlated segments as independent full answers. The archived settings additionally specify an entropy-loss coefficient, reported in Section 4.1.

4 Experiments

4.1 Experimental Setup

Models and data. We compare GRPO and TreeRPO using Qwen2.5-Math-1.5B and Qwen2.5-Math-7B (Yang et al., 2024a). Training uses the MATH training split, comprising 7,500 problems (Hendrycks et al., 2021). Evaluation covers AIME24, MATH-500, OlympiadBench, and Minerva Math (Lightman et al., 2023; He et al., 2024; Lewkowycz et al., 2022). All four tasks evaluate mathematical reasoning, so the experiment tests transfer across mathematical benchmarks rather than across unrelated task domains.

Training configuration. Both methods use temperature 0.6, learning rate 10^{-6}, maximum prompt length 512, and the default prompt batch size 128. GRPO samples eight responses per question, uses a response cap of 1,152 tokens, and a minibatch size of 64. TreeRPO uses branching factor eight, depth three, and 384 tokens per segment, matching the maximum tokens along one path but not the total number of generated paths. Its minibatch size is half of the constructed training-data size, which varies with the tree and filtering. The original settings specify KL coefficient $\beta = 0.001$ and entropy-loss coefficient $\alpha = -0.001$. The reward-range threshold is $\tau = 0.1$. A batch-size-16 setting is evaluated separately.

Implementation and evaluation. The experiments use A800 GPUs and the rllm framework (Luo et al., 2025a;b), built on verl (Sheng et al., 2024) with vLLM generation (Kwon et al., 2023). Answer correctness is determined by the rllm verification function. We report sampled Pass@1 accuracy, with an equal-weight macro average across the four benchmarks, rather than best-of-multiple-answer success. Evaluation temperature is 0.6. The archived prose specifies eight evaluation samples per question while its table caption specifies Avg@16; that sample-count discrepancy requires confirmation from the evaluation configuration. The reported benchmark values are retained without reinterpreting them as evidence for either sample count.

4.2 Accuracy Across Model Sizes

Table 1 shows higher macro accuracy for TreeRPO at both model sizes. For the 1.5B model, the gains are 3.0 points on AIME24, 2.8 on MATH-500, 2.4 on OlympiadBench, and 3.5 on Minerva Math. The macro gain is 2.9 points after rounding the mean difference. The improvement is therefore distributed across all four tasks rather than driven by only the largest evaluation score. These are absolute percentage-point changes, not relative percentage increases.

Table 1: Reported sampled Pass@1 accuracy (%) on four mathematical benchmarks. Macro is their equal-weight average. Benchmark scores and macro values are preserved from the source table; the original evaluation-sample-count discrepancy is described in Section 4.1.

Method	AIME24	MATH-500	Olympiad	Minerva	Macro
Qwen2.5-Math-1.5B					
GRPO	13.8	67.9	28.5	20.5	32.7
TREERPO	16.8	70.7	30.9	24.0	35.6
Qwen2.5-Math-7B					
GRPO	26.7	74.3	34.7	27.1	40.7
TREERPO	26.7	75.5	35.4	28.1	41.4

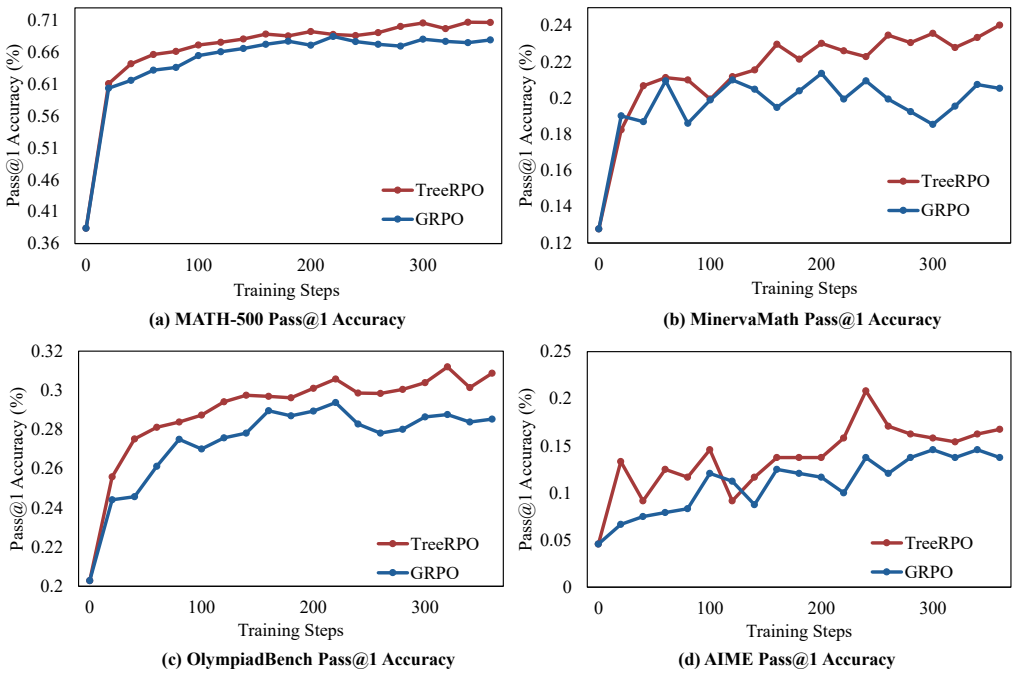

Figure 2: Qwen2.5-Math-1.5B accuracy over training steps, using the original four-panel figure. Panels cover MATH-500, Minerva Math, OlympiadBench, and AIME24. The plotted decimal accuracy values are fractions (for example, 0.7 means 70%), despite the percent symbol in the original axis label.

For the 7B model, MATH-500 improves by 1.2 points, OlympiadBench by 0.7, and Minerva Math by 1.0; AIME24 is unchanged at the displayed precision. Macro accuracy increases from 40.7% to 41.4%. The larger model already has a stronger GRPO baseline, but the present comparison does not isolate why its gain is smaller. Training-data difficulty, sampling variation, and the interaction between branching and model capability are possible explanations, not conclusions established by this experiment.

Progress during training. Figure 2 compares the 1.5B learning curves through 300 updates. Looking at the trajectory as well as the final table helps distinguish an isolated endpoint difference from a separation visible over multiple checkpoints. The panels show benchmark-dependent fluctuations, especially for AIME24. We interpret the curves as measurements along the reported runs, not independent replications. They support the observed training behavior but do not provide a confidence interval over random training seeds.

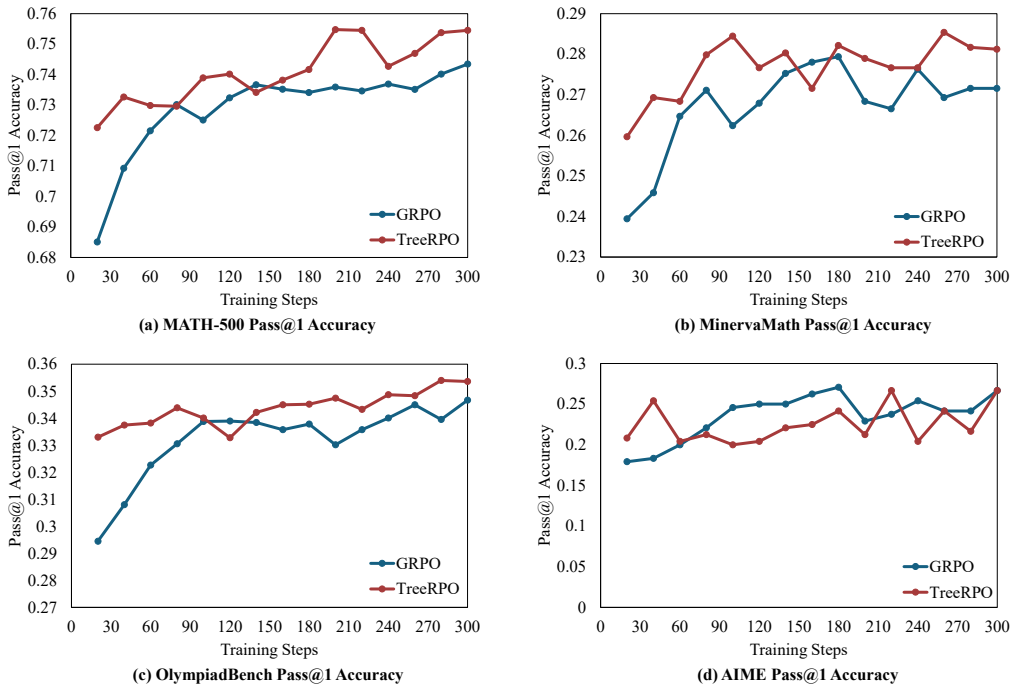

Figure 3: Qwen2.5-Math-7B accuracy over training steps on the same four benchmarks. Accuracy is plotted as a fraction. The final table shows improvements on three benchmarks and a tie on AIME24.

Figure 3 presents the corresponding 7B curves. The smaller endpoint improvements are consistent with a narrower separation in several panels. In particular, matching AIME24 in the final table is not inconsistent with differences at earlier checkpoints. The final comparison and the learning trajectory answer different questions: the table summarizes reported endpoint performance, while the curves reveal how performance develops during optimization.

4.3 Evaluation Response Length

For Qwen2.5-Math-1.5B, Figure 4 shows shorter average evaluation responses for TreeRPO across the four tasks. The source reports reductions of 17.1% on MATH-500, 22.3% on Minerva Math, 18.0% on OlympiadBench, and 15.3% on AIME24, together with an aggregate reduction of 18.1%. The arithmetic mean of the four displayed reductions rounds to 18.2%; the reported 18.1% aggregate is therefore retained as a separate source statistic rather than described as their exact arithmetic mean. Confirming its weighting requires the underlying length measurements.

The accuracy and length results must be considered together. A shorter answer is not necessarily a better answer, and optimizing brevity alone could omit necessary deductions. Here the 1.5B model's four accuracy scores increase while the reported average lengths decrease. This joint observation supports a more economical output behavior in the evaluated setting. It does not show that every answer becomes shorter, or that the policy has learned to identify all redundant reasoning steps.

Training cost follows a different accounting. A depth-three, eight-way tree can contain many more terminal paths than eight flat rollouts, even when both use the same 1,152-token maximum path length. Prefix reuse may reduce repeated generation work, and group filtering reduces update data, but neither makes the two procedures automatically compute-matched. The reported length reduction should therefore be read as an evaluation-time token-efficiency result. No new training-time speedup is inferred from it.

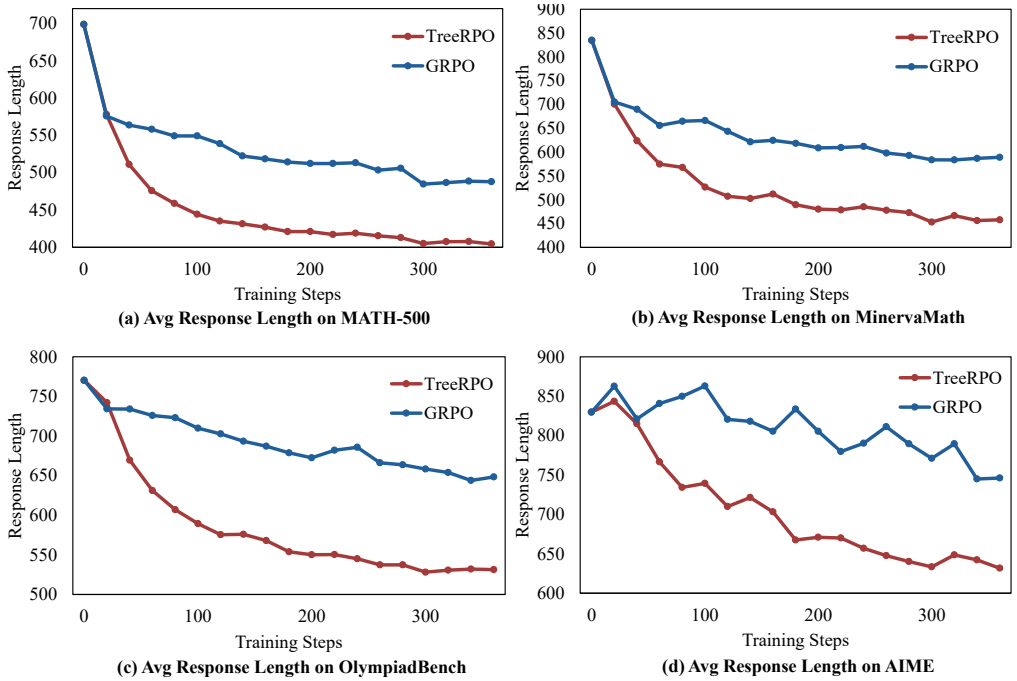

Figure 4: Average evaluation response length for Qwen2.5-Math-1.5B across training, in generated tokens. The original curves are preserved. These are lengths of evaluation answers, not total training-rollout token counts or wall-clock measurements.

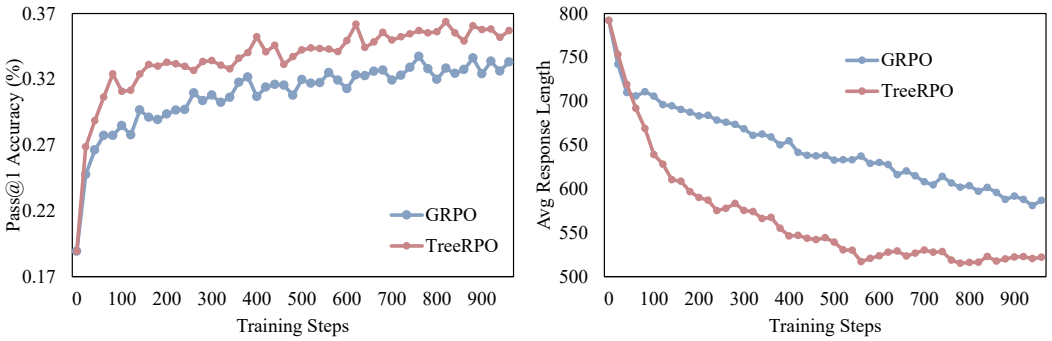

Figure 5: Batch-size-16 experiment: macro accuracy (left) and average response length (right) over the four benchmarks. Accuracy values are fractions despite the original percent axis label. Training-step counts alone do not equate compute across batch settings or sampling methods.

4.4 Smaller Batches and Qualitative Behavior

The batch-size-16 comparison in Figure 5 provides an additional setting beyond the default batch size of 128. It reports macro accuracy and average response length over the four tasks. The plotted trajectories retain an advantage for TreeRPO in accuracy and show shorter outputs over much of training. This is evidence from two tested batch settings, not a comprehensive hyperparameter robustness study. Because the smaller-batch plot spans a different number of updates, its horizontal axis should not be treated as a matched prompt count or matched compute budget across settings.

A qualitative example appears in Appendix A.3. Both responses correctly solve $1/(x-5) > 0$ as $(5, +\infty)$, while the TreeRPO response uses less repeated exposition. The example illustrates a possible manifestation of the aggregate length difference, rather than establishing a general reasoning mechanism. It is deliberately interpreted alongside the quantitative accuracy and length results, not used as an additional benchmark or an independent success-rate estimate.

4.5 What the Experiments Establish

The results support the combined tree-relative training procedure in the reported mathematical setting. They do not separately isolate tree branching, group filtering, and advantage scaling. Each changes a different part of the training process: branching allocates outcome samples to intermediate prefixes, filtering changes the groups retained for optimization, and scaling changes the magnitude of their updates. The present comparison evaluates that combination against GRPO. A component's mechanistic benefit should not be inferred solely from the full method's endpoint advantage.

Likewise, the two model sizes provide a useful scope check rather than a scaling law. The 1.5B model benefits on every reported benchmark, whereas the 7B model has smaller gains and one tie. This difference motivates examining how many informative sibling groups are available as model capability changes. If most children of a prefix succeed, their rewards can become nearly uniform and the group is filtered. If they all fail, the same issue occurs. Tree supervision is most informative where sampled alternatives differ in downstream outcomes; the experiment does not directly measure how often these cases arise.

4.6 Interpreting the Local Learning Signal

Why compare siblings rather than unrelated prefixes? The tree construction holds the preceding context fixed within each group. This matters because a terminal success rate can differ for two reasons: the quality of the current segment and the difficulty of the state from which it starts. Comparing siblings removes the second difference at the point where the alternatives are sampled. The common parent is already part of the conditioning context, so the centered reward asks whether one extension has better sampled prospects than its local alternatives. This does not eliminate uncertainty about future decisions, but it makes the comparison more specific than assigning a single relative score to an entire answer. A trajectory can consequently contain segments with different local advantages, instead of every segment inheriting the same trajectory-level comparison.

How terminal evidence is reused. One verified leaf supplies information at several depths. Its outcome contributes to the reward of its parent, then to the parent's parent, and so on. This reuse gives the method intermediate supervision without asking the verifier to inspect unfinished mathematics. Ancestor estimates share descendant outcomes, so the training segments are not independent reward observations. We therefore interpret the tree as a structured allocation of a finite outcome-sampling budget, not as an unlimited source of independent process labels. Wider or deeper sampling can change that allocation, but the current results evaluate one default branching and depth configuration rather than an optimized allocation rule.

Relating local supervision to shorter outputs. The verifier rewards the final answer, and the specified tree objective does not directly reward a shorter evaluation response. The observed length reduction therefore describes a learned output behavior, not a mechanical shortening of the evaluation cap. A possible explanation is that local comparisons help favor continuations that reach a successful answer with less unnecessary exposition. The qualitative example is compatible with that interpretation, but does not establish it causally. A long continuation may be necessary, and a short one can still be incorrect. The joint accuracy and length measurements are more informative than either quantity alone: for the 1.5B model, the reported improvement in correctness accompanies, rather than trades off against, shorter outputs on these benchmarks.

5 Conclusion

TreeRPO constructs intermediate learning signals from terminal verification by sampling continuation trees and comparing sibling segments. It combines bottom-up rewards, informative-group filtering, and relative policy updates without a separately trained process reward model. The reported experiments show higher macro accuracy than GRPO for Qwen2.5-Math-1.5B and 7B, with shorter evaluation responses for the 1.5B model. The main contribution is the use of sampled downstream outcomes for local training comparisons. Improving segmentation, controlling tree-sampling cost, and isolating the contributions of individual components remain useful directions. Training-cost measurements remain distinct from evaluation token savings.

AI USE STATEMENT

The original manuscript reports using DeepSeek-V3.1 and Gemini 2.5 for language polishing after the authors developed the research ideas, methods, experiments, and analyses. A generative AI assistant was also used to help revise the organization, explanatory text, and formatting of this manuscript for the present template. No new experimental results were generated by the assistant. The authors are responsible for reviewing AI-assisted edits and verifying the final manuscript.

ETHICS STATEMENT

This work studies mathematical reasoning using language-model training and benchmark evaluation; it does not involve human-subject experiments. Correctness on mathematical benchmarks does not guarantee that the resulting models are reliable or safe in other applications. Deployment should include task-appropriate evaluation and safeguards.

REPRODUCIBILITY STATEMENT

Section 3 defines tree sampling, reward propagation, group filtering, and the policy objective. Section 4.1 reports the available training and evaluation settings, and Appendix A.1 consolidates the configuration and remaining reporting ambiguities. All numerical results and figures are retained from the source manuscript; no additional experiments are claimed.

REFERENCES

Zhen Bi, Ningyu Zhang, Yinuo Jiang, Shumin Deng, Guozhou Zheng, and Huajun Chen. When do program-of-thoughts work for reasoning? *arXiv preprint arXiv:2308.15452*, 2023.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

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

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

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

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model, 2025. URL https://arxiv.org/abs/2503.24290.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms, 2024. URL https://arxiv.org/abs/2406.18629.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 3843–3857. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/18abbeef8cfe9203fdf9053c9c4fe191-Paper-Conference.pdf.

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

Jianqiao Lu, Zhiyang Dou, Hongru Wang, Zeyu Cao, Jianbo Dai, Yunlong Feng, and Zhijiang Guo. Autopsv: Automated process-supervised verifier. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024a. URL http://papers.nips.cc/paper_files/paper/2024/hash/9246aa822579d9b29a140ecdac36ad60-Abstract-Conference.html.

Zimu Lu, Aojun Zhou, Houxing Ren, Ke Wang, Weikang Shi, Junting Pan, Mingjie Zhan, and Hongsheng Li. MathGenie: Generating synthetic data with question back-translation for enhancing mathematical reasoning of LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2732–2747, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.151. URL https://aclanthology.org/2024.acl-long.151/.

Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, Ce Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepcoder: A fully open-source 14b coder at o3-mini level, 2025a. Notion Blog.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl, 2025b. Notion Blog.

OpenAI. Learning to reason with llms, 2024. URL https://openai.com/index/learning-to-reason-with-llms/.

Qwen. Qwq-32b: Embracing the power of reinforcement learning, 2024. URL https://qwenlm.github.io/blog/qwq-32b/.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

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

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. DART-math: Difficulty-aware rejection tuning for mathematical problem-solving. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=zLU21oQjD5.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning, 2025. URL https://arxiv.org/abs/2506.01939.

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

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

Zhicheng Yang, Yinya Huang, Jing Xiong, Liang Feng, Xiaodan Liang, Yiwei Wang, and Jing Tang. AlignedCoT: Prompting large language models via native-speaking demonstrations. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 2857–2896, Miami, Florida, USA, November 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.163. URL https://aclanthology.org/2024.findings-emnlp.163/.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf.

Fei Yu, Anningzhe Gao, and Benyou Wang. Outcome-supervised verifiers for planning in mathematical reasoning. *arXiv preprint arXiv:2311.09724*, 2023.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=N8N0hgNDRt.

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

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models, 2023.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, Xiangpeng Wei, Xiangyu Yu, Gaohong Liu, Juncai Liu, Lingjun Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Ru Zhang, Xin Liu, Mingxuan Wang, Yonghui Wu, and Lin Yan. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks, 2025. URL https://arxiv.org/abs/2504.05118.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild, 2025. URL https://arxiv.org/abs/2503.18892.

Yifan Zhang, Jingqin Yang, Yang Yuan, and Andrew Chi-Chih Yao. Cumulative reasoning with large language models. *arXiv preprint arXiv:2308.04371*, 2023.

A Additional Details

A.1 Configuration and Reporting Notes

Table 2 consolidates the configuration given in the original manuscript. The available archive contains manuscript sources and figures, not training code or resolved run configurations. Entries therefore document the reported setup rather than an independently reproduced execution.

Table 2: Reported training configuration. The tree path cap equals depth times segment cap; it is not a cap on total tokens generated for a question.

Setting	Value
Models	Qwen2.5-Math-1.5B and 7B
Training data	MATH training split, 7,500 problems
Training / evaluation temperature	0.6 / 0.6
Learning rate	10^{-6}
Default prompt batch	128
Additional batch setting	16
GRPO responses per question	8
GRPO minibatch	64
TreeRPO minibatch	Half of constructed training-data size
Maximum prompt length	512
GRPO maximum response length	1,152
Tree branching factor N	8
Maximum depth D	3
Segment token cap L_{step}	384
Reward-range threshold τ	0.1
KL coefficient β	0.001
Entropy-loss coefficient α	-0.001
Accelerator	A800

Evaluation sample count. The original parameter paragraph states that eight candidate answers are sampled per question, while the main-table caption states Avg@16. Both describe an estimate of single-answer correctness, not a best-of-n score, but they imply different numbers of samples. The discrepancy cannot be resolved from the manuscript archive alone. The revised table avoids asserting a sample count and retains all reported benchmark values.

Numerical consistency. The source table reports 35.6% macro accuracy for the 1.5B method, whereas its abstract states 35.5%. The revised abstract follows the table. The OlympiadBench gain is 2.4 points from 30.9 minus 28.5; the original inline delta of 2.6 was inconsistent with those scores. MATH-500 increases by 2.8 points from 67.9 to 70.7. The four displayed evaluation-length reductions average to 18.175%, while the original aggregate is 18.1%; the aggregate's weighting should be checked against the original measurements. These observations do not replace any benchmark score or curve value.

A.2 Tree Size and Shared Prefixes

A fully expanded tree with branching factor N and depth D has at most N^D leaves and $\sum_{d=1}^{D} N^d$ generated edges, excluding the root. At the reported values, these bounds are 512 leaves and 584 edges. With at most 384 tokens per edge, the corresponding upper bound on edge-token generation is 224,256 tokens per question, before early stopping. This is a structural bound, not a measured training token count. Eight independent GRPO responses at a 1,152-token cap have an upper bound of 9,216 response tokens per question. Matching path length therefore does not match the total sampling budget.

Shared prefixes can be reused in storage and generation, avoiding a complete restart for every leaf. However, prefix sharing does not remove the need to generate distinct child segments, score terminal outcomes, or process retained groups. Realized costs depend on stopping, tree occupancy, batching, and cache management. The manuscript does not supply measurements that separate these costs, so the response-length experiment is not presented as a training-throughput benchmark.

A.3 Illustrative Response Pair

Figure 6 preserves the original example. Both methods obtain the correct interval. Its role is to illustrate differences in exposition, not to establish that the shorter response is always more correct or that the example is representative of every benchmark.

Question: Solve $\frac{1}{x-5} > 0$. Enter your answer using interval notation.

TreeRPO:

To solve the inequality $\frac{1}{x-5} > 0$, we need to determine when the expression $\frac{1}{x-5}$ is positive. The fraction $\frac{1}{x-5}$ is positive when the denominator $x-5$ is positive. Therefore, we need to solve the inequality $x-5>0$.

Adding 5 to both sides of the inequality, we get:

$$x > 5$$

So, the solution to the inequality $\frac{1}{x-5} > 0$ is $(5, +\infty)$.

The final answer in interval notation is: \boxed{$(5, +\infty)$}.

GRPO:

To solve the inequality $\frac{1}{x-5} > 0$, we need to determine the values of x for which the expression $\frac{1}{x-5}$ is positive. The expression $\frac{1}{x-5}$ is positive when the denominator $x-5$ is positive. Therefore, we need to solve the inequality $x-5>0$.

Let's solve this step-by-step:

1. Start with the inequality:

$$x-5>0$$

2. Add 5 to both sides of the inequality:

$$x>5$$

So, the solution to the inequality $\frac{1}{x-5} > 0$ is $x>5$. In interval notation, this is written as $(5, +\infty)$.

The final answer is: \boxed{$(5, +\infty)$}.

Figure 6: Original response example for $1/(x-5) > 0$. Both outputs give $(5, +\infty)$; the TreeRPO output contains less repeated exposition.

A.4 Limitations and Future Directions

The experiments are restricted to mathematical reasoning with two Qwen2.5-Math model sizes. Fixed-token segments do not necessarily align with logical steps, and recursively averaged outcomes measure success under sampled continuations rather than local validity. Tree expansion can be expensive, particularly without optimized KV-cache sharing. More precise segmentation, redundancy-aware data construction, and better scheduling of branches are natural extensions. Work on rejection-sampled reasoning data suggests that redundant responses deserve attention (Yuan et al., 2023); branching at uncertain positions is another possible direction (Wang et al., 2025). These extensions have not been evaluated in the reported experiments.