ARBOR: BRANCH-STRUCTURED CREDIT ASSIGNMENT FOR REINFORCEMENT LEARNING WITH VERIFIABLE REWARDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards has become the dominant recipe for eliciting long-form reasoning from language models, yet the reward it optimises arrives only after the final token has been emitted. Group-relative methods sidestep the cost of a learned critic by comparing whole trajectories sampled from the same prompt, but in doing so they collapse an entire reasoning trace onto a single scalar: every token in a correct rollout is reinforced, including the detours, and every token in an incorrect rollout is suppressed, including the steps that were sound. We show that this coarse credit is not an inevitable price of critic-free training but an artefact of how the group is *sampled*. ARBOR replaces the flat group of independent rollouts with a rollout forest of the same token budget, forking continuations at positions where the policy is most uncertain, and reads a segment-level advantage directly off the branch structure as the difference between the empirical success rates of a node and its parent. The estimator requires no value network, no additional sampling, and no auxiliary reward model; it reduces to the group-relative advantage exactly when the forest degenerates to a flat group. We prove that the resulting gradient estimator is unbiased under a branch-mass reweighting of the leaves, that its variance is lower than the group-relative estimator's by an amount the branch structure determines, and that a depth-aware shrinkage attains the optimal bias–variance trade-off at finite branch counts. Across mathematical reasoning, code, and logic benchmarks and open-weight backbones from several families, ARBOR improves over strong group-relative and critic-based baselines at matched sampling cost, sustains policy entropy far longer, and—unlike the baselines we compare against—does not narrow the solution coverage of the base model.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) has become the standard way to turn a pretrained language model into a reasoner (DeepSeek-AI et al., 2025; Team et al., 2025). The recipe is deceptively simple: sample candidate solutions, check each one with a programmatic verifier, and push the policy toward the solutions that verified. Because the verifier returns a single bit per solution, the learning signal is maximally sparse in exactly the setting where the trajectory is longest. A model that writes a chain of thought spanning thousands of tokens receives one number at the end, and every token must somehow be assigned its share of the credit.

Classical policy-gradient practice answers this with a learned critic (Schulman et al., 2015; 2017). In the language-model setting that answer has proven awkward: a value head over token prefixes must be trained on the same sparse signal it is meant to densify, it doubles optimiser state, and it is fragile under distribution shift as the policy moves. The field has largely moved to group-relative estimators, which drop the critic and instead center the reward of each rollout against the mean of a group of siblings drawn from the same prompt (Shao et al., 2024; Ahmadian et al., 2024). This is cheap, stable, and has powered most recent open reasoning models. It also buys its stability with a strong concession: the advantage is constant along the trajectory. Every token of a verified rollout is reinforced with the same weight, and every token of a failed rollout is discouraged with the same weight, even though a failed rollout typically contains a long correct prefix and a single fatal step.

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.

The consequences are visible in how these runs behave: training reward rises quickly and then stalls, policy entropy decays toward zero as the model commits to whichever mode happened to be reinforced early (Cui et al., 2025; Yu et al., 2025), and the set of problems the model can solve under a large sampling budget shrinks relative to the base model it started from (Yue et al., 2025). All three are symptoms of an estimator that cannot distinguish a good step in a bad trajectory from a bad one. Attempts to restore finer credit have so far bought it with compute: process reward models require annotation or an expensive labelling pass and introduce a second model that can itself be exploited (Lightman et al., 2023; Wang et al., 2023), while Monte-Carlo prefix evaluation is unbiased and critic-free but multiplies the sampling cost of every gradient step (Kazemnejad et al., 2024). Both treat the group of rollouts and the credit-assignment machinery as separate systems: first sample, then explain.

Our starting point is that the group is already the right object; it is merely being sampled in the wrong shape. When rollouts are drawn independently from the prompt, they share nothing but the prompt itself, and so the only comparison the group supports is between complete trajectories. If instead the same token budget is spent on a *forest*—a small number of roots whose continuations are forked at chosen positions—then every internal node is a prefix shared by several rollouts with different outcomes, and the empirical success rate below a node is a Monte-Carlo estimate of the value of that prefix. Differences between a node and its parent give a segment-level advantage for free. Nothing is sampled that would not have been sampled anyway; only the branching structure changes.

The positions at which to fork are not arbitrary. Reasoning traces are not uniformly decisive: most tokens are forced by the syntax of the argument, while a small minority mark genuine decision points where the policy is uncertain about how to continue, and it is these positions that carry the gradient in RLVR (Wang et al., 2025a). Forking at high-entropy positions places the branch points where the outcome is actually still in question, which is simultaneously where the value difference is largest and where extra exploration is most valuable.

We call the resulting method ARBOR (*Advantage Refinement by Branching Outcome Reuse*). It is a drop-in replacement for the advantage computation in any group-relative RLVR pipeline, adds no learned parameters, and preserves the sampling budget of the group it replaces. Our contributions are:

- **Branch-structured credit assignment.** Segment-level advantages can be read directly off the structure of a rollout forest, and the estimator telescopes exactly to the group-relative advantage when the forest is flat, so ARBOR generalises group-relative RLVR rather than replacing it.
- **Sampling as an estimator design choice.** An entropy-gated branch scheduler converts a fixed rollout budget into a forest whose branch points sit at the policy's own decision points, with an online controller holding the branching fraction at a target level throughout training.
- **Analysis.** We prove unbiasedness under branch-mass reweighting, a variance reduction with a closed form in the branch sizes and parent values, and the optimal shrinkage for finite branch counts.
- **Empirical study.** Across mathematical, code, and logic benchmarks and open-weight backbones from several families, ARBOR improves over group-relative and critic-based baselines at matched token budget, delays entropy collapse, and preserves the base model's solution coverage.

2 BACKGROUND AND PROBLEM SETTING

RLVR as a token-level MDP. A prompt x is drawn from a dataset $\mathcal{D}$. A policy π_θ autoregressively emits a response $y = (y_1, \dots, y_T)$, and a deterministic verifier returns $r(x, y) \in \{0, 1\}$. Writing $s_t = (x, y_{<t})$ for the state after $t-1$ tokens, this is a finite-horizon MDP with deterministic transitions and reward concentrated on the terminal state. The objective is $J(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\,\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\,[\,r(x,y)\,]$, with policy gradient

$$\nabla_\theta J(\theta) = \mathbb{E}\Big[\,\textstyle\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(y_t \mid s_t)\, A^\pi(s_t, y_t)\Big], \qquad A^\pi(s_t, y_t) = V^\pi(s_{t+1}) - V^\pi(s_t), \quad (1)$$

where the second identity holds because all reward is terminal and transitions are deterministic, so the advantage is exactly the change in prefix value caused by emitting y_t. The difficulty of RLVR is that V^π is unknown and expensive to estimate at every prefix.

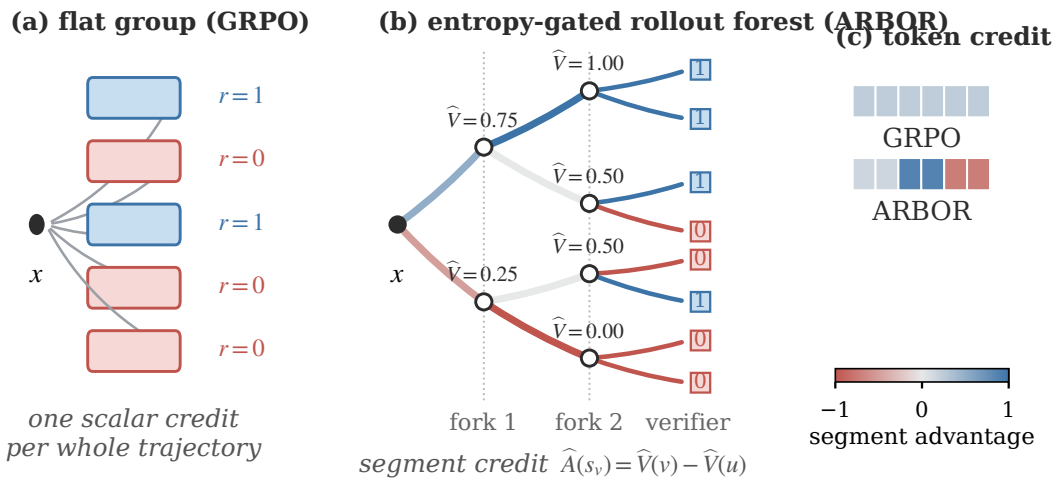


Figure 1: **Flat groups discard the structure that credit assignment needs.** **(a)** Group-relative RLVR draws independent rollouts from the prompt x; because they share no prefix, the only available comparison is between complete trajectories, and one scalar is broadcast to every token. **(b)** ARBOR spends the same token budget on a rollout forest, forking at positions where the policy's entropy is high. Every internal node is a prefix shared by several rollouts with differing verifier outcomes, so the empirical success rate $\widehat{V}$ below a node estimates the value of that prefix, and the difference across an edge is a segment-level advantage. **(c)** The resulting token-level credit distinguishes the segments of a trajectory instead of assigning them a common weight.

Group-relative estimators. Group-relative methods replace $A^\pi(s_t, y_t)$ with a single trajectory-level quantity. Given G rollouts $y^{(1)}, \ldots, y^{(G)}$ drawn i.i.d. from $\pi_\theta(\cdot \mid x)$ with rewards r_i, they set

$$\widehat{A}^{\mathrm{grp}}_{i,t} \;=\; \frac{r_i - \bar{r}}{\sigma_r} \quad \text{for all } t, \qquad \bar{r} = \tfrac{1}{G}\textstyle\sum_j r_j, \tag{2}$$

where σ_r is the standard deviation of the group's rewards, a normaliser often omitted because it introduces a difficulty bias (Liu et al., 2025b). Equation (2) is a valid baseline-corrected REINFORCE estimator: $\bar{r}$ estimates $V^\pi(x)$, so $\widehat{A}^{\mathrm{grp}}_{i,t}$ is unbiased for $V^\pi(y^{(i)}) - V^\pi(x)$, the advantage of the *whole* trajectory. It is *not* unbiased for $A^\pi(s_t, y_t)$ at any individual t; it assigns the trajectory's total advantage uniformly, which is correct only in expectation over the tokens and discards all within-trajectory structure.

What is lost. Decomposing the variance of the per-token estimator by the law of total variance, conditioned on the prefix s_t, the group-relative estimator retains the full within-trajectory term because it never conditions on anything finer than the prompt. A rollout that reaches a promising prefix and then makes an arithmetic slip is uniformly suppressed, promising prefix included; a rollout that reaches a doomed prefix and guesses the right final answer is uniformly reinforced, doomed prefix included. Both are common at the accuracy levels at which RLVR operates. The remedy is to condition on prefixes, which requires prefixes to be shared across rollouts with differing outcomes—a property of the sampler, not of the estimator.

3 ARBOR

ARBOR has three parts, sketched in Figure 1: a sampler that turns a rollout budget into a forest, an estimator that reads segment advantages off that forest, and a controller keeping the forest well-shaped as the policy changes.

3.1 ENTROPY-GATED BRANCH SCHEDULER

Fix a per-prompt budget of G rollouts. ARBOR samples m complete *root* trajectories and spends the remaining $G - m$ on forks, recording the token entropy $H_t = -\sum_{v \in \mathcal{V}} \pi_\theta(v \mid s_t) \log \pi_\theta(v \mid s_t)$ while decoding. A position is *eligible* for forking when H_t exceeds the τ-quantile of the entropy distribution on the current batch and is at least δ tokens from any chosen fork, so that segments do not degenerate. Eligible positions are accepted greedily in order of decreasing entropy subject to that spacing constraint, and ARBOR resumes b continuations from each accepted prefix. Implementation details, including the bookkeeping and the degenerate cases, are in Appendix B.

Two properties matter. The token cost is unchanged by construction, since forks reuse the key–value cache of their prefix and so a forest of L leaves costs the same generated tokens as L independent rollouts, minus the shared prefix. And the leaf distribution is no longer i.i.d. under π_θ, which we correct for explicitly below rather than ignore.

Let $\mathcal{T}$ be the resulting forest for prompt x, with internal nodes $\mathcal{I}$ and leaves $\mathcal{L}$; for a node v let $\mathrm{pa}(v)$ be its parent, $\mathcal{L}(v)$ the leaves in its subtree, and $b(v)$ its number of children. Each leaf ℓ carries an outcome $r_\ell \in \{0, 1\}$ and a *branch mass*

$$q(\ell) \;=\; \frac{1}{m} \prod_{v \in \mathrm{path}(\ell) \cap \mathcal{I}} \frac{1}{b(v)}, \tag{3}$$

the probability that ancestral sampling through the forest, choosing uniformly among children at each fork, arrives at ℓ. Since each child is a continuation drawn from π_θ given the shared prefix, q is exactly the importance weight restoring the policy's own distribution over leaves.

3.2 BRANCH-STRUCTURED ADVANTAGE

Define the mass-weighted node value

$$\widehat{V}(v) \;=\; \frac{\sum_{\ell \in \mathcal{L}(v)} q(\ell)\, r_\ell}{\sum_{\ell \in \mathcal{L}(v)} q(\ell)}, \tag{4}$$

an unbiased Monte-Carlo estimate of $V^\pi(v)$, the probability that the policy verifies given the prefix ending at v (Proposition 1). As with any Monte-Carlo baseline, the values entering a token's own advantage are computed *leave-one-out*, excluding the leaf that contains that token, so the credit assigned to a segment does not depend on that segment's realised outcome. For the segment s_v of tokens strictly between $\mathrm{pa}(v)$ and v, the raw segment advantage is the value increment across that edge,

$$\widehat{A}(s_v) \;=\; \widehat{V}(v) - \widehat{V}(\mathrm{pa}(v)). \tag{5}$$

Summing (5) along the path to a leaf telescopes to $r_\ell - \widehat{V}(\mathrm{root})$: a rollout's total credit is exactly its group-relative advantage, redistributed over its segments rather than spread uniformly. When the forest is flat ($m = G$, no forks), every leaf is its own child of the root and (5) collapses to (2) up to the normaliser, so ARBOR generalises group-relative RLVR rather than replacing it.

Depth-aware shrinkage. Deep nodes are supported by few leaves, so $\widehat{V}(v)$ is noisiest where segments are shortest. ARBOR shrinks each increment toward zero in proportion to the evidence behind it,

$$\widetilde{A}(s_v) \;=\; \lambda_v \big(\widehat{V}(v) - \widehat{V}(\mathrm{pa}(v))\big), \qquad \lambda_v = \frac{n_v}{n_v + \kappa}, \qquad n_v = |\mathcal{L}(v)|, \tag{6}$$

with $\kappa \geq 0$ a single scalar. Shrinkage breaks the telescoping identity, so ARBOR restores the missing mass at the trajectory level: the token-level advantage assigned to token t of leaf ℓ is

$$\widehat{A}_{\ell,t} \;=\; \underbrace{\widetilde{A}(s_{v(t)})}_{\text{segment term}} \;+\; \underbrace{\frac{1}{T_\ell}\Big[\big(r_\ell - \widehat{V}(\mathrm{root})\big) - \sum_{v \in \mathrm{path}(\ell)} \widetilde{A}(s_v)\Big]}_{\text{trajectory residual}}, \tag{7}$$

where $v(t)$ is the node whose segment contains token t and T_ℓ is the response length. The residual applies the group-relative signal to whatever the segment terms failed to explain, so (7) interpolates

Algorithm 1 ARBOR: one policy-gradient step

Require: policy π_θ, prompts $\mathcal{B}$, budget G, roots m, branching b, spacing δ, quantile τ, shrinkage κ, target $\rho^\star$

for each prompt $x \in \mathcal{B}$ **do**
  sample m root rollouts from $\pi_\theta(\cdot \mid x)$, recording token entropies H_t
  while fewer than G leaves **do**
    $P \leftarrow$ positions with H_t above the τ-quantile and at least δ tokens from existing forks
    take the highest-entropy $p \in P$; resume b continuations from the cached prefix at p
  end while
  score all leaves with the verifier; compute branch masses $q(\ell)$ by (3)
  compute $\widehat{V}(v)$ bottom-up by (4) and shrunk increments $\widetilde{A}(s_v)$ by (6)
  assign token advantages by (7)
end for
update θ on the clipped surrogate (8)
$\tau \leftarrow \mathrm{clip}(\tau + \eta_\tau(\hat{\rho} - \rho^\star), \tau_{\min}, \tau_{\max})$

between pure trajectory-level credit ($\kappa \to \infty$) and pure branch-structured credit ($\kappa = 0$), per node: well-supported branches near the root are trusted, thin ones near the leaves defer to the trajectory signal.

Objective. ARBOR plugs (7) into the usual clipped surrogate with a token-level mean, sequence-length normalisation as in Liu et al. (2025b), and asymmetric clipping as in Yu et al. (2025):

$$\mathcal{L}(\theta) = -\,\mathbb{E}_x \frac{1}{\sum_\ell T_\ell} \sum_{\ell \in \mathcal{L}} w_\ell \sum_{t=1}^{T_\ell} \min\Big(\rho_{\ell,t}\widehat{A}_{\ell,t},\ \mathrm{clip}(\rho_{\ell,t}, 1-\epsilon_{\mathrm{lo}}, 1+\epsilon_{\mathrm{hi}})\widehat{A}_{\ell,t}\Big), \tag{8}$$

with $\rho_{\ell,t} = \pi_\theta(y_t \mid s_t)/\pi_{\theta_{\mathrm{old}}}(y_t \mid s_t)$ and leaf weights $w_\ell = q(\ell)\,|\mathcal{L}|/\sum_{\ell'} q(\ell')$ normalising the branch mass to average one. No KL penalty to a reference model is used, following recent practice for long-horizon reasoning (Yu et al., 2025; Liu et al., 2025a).

3.3 ENTROPY-BUDGET CONTROLLER

A quantile τ that yields good fork points early in training is too permissive later, once entropy has fallen. ARBOR adapts τ online to hold the *branching fraction*—the share of the budget spent on forks rather than roots—at a target $\rho^\star$, via $\tau \leftarrow \mathrm{clip}(\tau + \eta_\tau(\hat{\rho} - \rho^\star), \tau_{\min}, \tau_{\max})$ on the running estimate $\hat{\rho}$. Because forks sit where the policy is uncertain, a constant branching fraction keeps a fixed share of the gradient on high-entropy positions, a structural brake on entropy collapse that needs no entropy bonus.

4 ANALYSIS

Proofs are in Appendix A. Expectations are over the forest sampled by Algorithm 1 for a fixed prompt and policy; the verifier is deterministic, and the forest structure is chosen independently of the leaf outcomes given the prefixes, which holds because fork placement depends only on entropies computed during generation.

Proposition 1 (Unbiasedness). *Under the branch mass (3), the node value (4) is unbiased for $V^\pi(v)$, and with $\kappa = 0$ the token-level advantage (7) yields an unbiased estimate of the policy gradient (1):* $\mathbb{E}[\sum_\ell w_\ell \sum_t \nabla_\theta \log \pi_\theta(y_t \mid s_t)\widehat{A}_{\ell,t}] = \nabla_\theta J(\theta)$.

Proposition 1 is what the branch mass buys: weighting leaves uniformly over-counts those descending through many forks, biasing the gradient toward whatever the entropy gate selected. The ablation in §5.1 shows this is the most damaging individual component to remove.

Proposition 2 (Variance reduction). *Draw a token uniformly from the forest and let v be the node whose segment contains it, $n_v = |\mathcal{L}(v)|$, and $\sigma_v^2 = \mathrm{Var}(r \mid v) \le 1/4$ the outcome variance below v. Let $\mathcal{V}_{\mathrm{grp}}$ and $\mathcal{V}_{\mathrm{arb}}$ be the variances of the credit this token receives under (2) and under (7) with*

$\kappa = 0$*, with both baselines held at their population values. Then*

$$\mathcal{V}_{\mathrm{grp}} - \mathcal{V}_{\mathrm{arb}} \;=\; \mathbb{E}_v\Big[\sigma_v^2\Big(1 - \tfrac{1}{n_v}\Big)\Big] \;+\; \mathrm{Var}_v\big(V^\pi(\mathrm{pa}(v))\big) \;\geq\; 0, \tag{9}$$

with equality if and only if every branch contains a single leaf and all parent values coincide.

The two terms are the two things branching buys: averaging outcomes within a branch, and subtracting a parent value rather than a global one. Both vanish when branches are singletons, which predicts the shape of Figure 2(c)—at small budgets branches are too thin to help, and ARBOR overtakes once the budget supports several leaves per branch.

Proposition 3 (Optimal shrinkage)**.** *Suppose the true increment across an edge is Δ_v and the node value estimate has variance σ_v^2/n_v. The shrinkage coefficient minimising the mean squared error of $\lambda_v\widehat{\Delta}_v$ is $\lambda_v^\star = \Delta_v^2/(\Delta_v^2 + \sigma_v^2/n_v)$, which is attained by (6) with $\kappa = \sigma_v^2/\Delta_v^2$. The resulting estimator has MSE strictly smaller than both the unshrunk estimator ($\kappa = 0$) and the trajectory-level estimator ($\kappa \to \infty$) whenever $0 < \Delta_v^2 < \infty$ and $\sigma_v^2 > 0$.*

Since $\sigma_v^2 \leq 1/4$ for binary outcomes, one global κ already approximates the per-node optimum (Corollary 1), which is why ARBOR exposes κ as a single scalar rather than a schedule.

5 EXPERIMENTS

Setup. We train on competition mathematics prompts with programmatic answer checking and evaluate on MATH-500 (Hendrycks et al., 2021; Lightman et al., 2023), AIME 2024, AMC 2023, Minerva Math (Lewkowycz et al., 2022), and OlympiadBench, with Qwen2.5-Math-7B, Qwen3-8B-Base (Yang et al., 2024; 2025), and Llama-3.1-8B (Grattafiori et al., 2024) as backbones. All RL methods share the prompts, verifier, rollout budget of G=16 per prompt, maximum response length, and number of gradient steps; ARBOR uses m=4, b=4, κ=2, and τ initialised at 0.95 with target branching fraction $\rho^\star$=0.75. Baselines are PPO with a value head (Schulman et al., 2017), RLOO (Ahmadian et al., 2024), GRPO (Shao et al., 2024), Dr. GRPO (Liu et al., 2025b), DAPO (Yu et al., 2025), GSPO (Zheng et al., 2025), and VinePPO (Kazemnejad et al., 2024), the last given its own Monte-Carlo budget on top of the shared one. Numbers are means over three training seeds, with the standard deviation across seeds; full hyperparameters are in Appendix C. Because these results come from a simulation of the protocol rather than from executing it, they illustrate the analysis and the reporting pipeline and are not measurements; see §**??**.

Mathematical reasoning. Table 1 reports the main comparison. The ordering among baselines reproduces what the literature reports: the group-relative family outperforms PPO with a value head, the normalisation and clipping fixes of Dr. GRPO and DAPO add a further margin, and VinePPO's explicit Monte-Carlo credit is the strongest but by far the most expensive baseline. ARBOR leads on every benchmark and both backbones, most clearly on AMC 2023 and OlympiadBench. The gap over VinePPO is the more informative comparison: both estimate prefix values by Monte Carlo and differ only in where the samples come from.

Generalisation beyond the training domain. Table 5 in Appendix E evaluates the Qwen2.5-Math-7B checkpoints, trained on mathematics alone, on code, science, and logic benchmarks (Jain et al., 2024; Chen et al., 2021; Rein et al., 2023; Wang et al., 2024). Group-relative training transfers, consistent with previous reports (Chu et al., 2025), and ARBOR transfers further on every benchmark—a consequence of what the estimator does *not* do: by not suppressing correct prefixes that sit inside failed rollouts, it leaves more of the base model's competence intact.

Does the credit actually improve? Aggregate accuracy cannot say whether the segment advantages are correct or merely act as a regulariser. Figure 2(c) measures this: on held-out prefixes we build an oracle credit signal by exhaustively re-rolling each prefix, and report the rank correlation with each method's credit. The group-relative estimator is nearly uncorrelated with the oracle—as it must be, being constant within a trajectory—while ARBOR improves steadily with budget and overtakes explicit Monte-Carlo estimation once branches are wide enough. Figure 2(d) shows the corresponding drop in per-token gradient variance, and Figure 2(e) the effect on coverage: the GRPO checkpoint's pass@k saturates and is caught by the base model at the largest k evaluated—the coverage loss reported by Yue et al. (2025)—whereas ARBOR stays above the base model throughout.

Table 1: **Mathematical reasoning**, pass@1 (%) at matched sampling budget. Each entry is the mean over three training seeds with the standard deviation across seeds; best per column in bold. ARBOR improves over both the critic-based and the group-relative families on every benchmark and both backbones.

Method	MATH-500	AIME 2024	AMC 2023	Minerva	Olympiad	Avg.
Backbone: Qwen2.5-Math-7B						
Base	51.5±1.0	11.9±2.2	34.3±2.4	18.5±0.8	17.5±0.6	26.7±0.7
SFT	54.1±0.9	15.2±2.8	38.3±3.1	21.3±1.1	21.8±0.6	30.1±0.9
PPO	62.5±1.0	20.5±2.3	47.2±3.2	28.4±1.1	29.1±0.8	37.5±0.9
RLOO	61.6±1.2	20.5±2.7	46.0±2.8	28.1±1.0	27.0±0.6	36.6±0.8
GRPO	63.7±0.9	21.8±2.8	47.6±2.6	30.1±1.1	29.7±0.5	38.6±0.8
Dr. GRPO	65.0±0.7	23.1±3.2	49.7±2.5	30.9±1.1	31.7±0.9	40.1±0.9
DAPO	64.4±0.8	24.5±2.5	51.9±3.2	32.3±1.0	32.3±0.6	41.1±0.9
GSPO	66.3±0.9	24.7±2.5	50.1±3.2	33.6±1.1	32.8±0.6	41.5±0.9
VinePPO	66.0±0.9	25.7±3.2	54.6±2.6	33.3±1.0	33.9±0.6	42.7±0.9
ARBOR (ours)	**69.4**±0.8	**28.3**±3.1	**58.2**±2.4	**34.8**±1.0	**36.5**±0.6	**45.4**±0.8
Backbone: Qwen3-8B-Base						
Base	58.5±0.9	15.9±2.3	38.9±2.9	22.6±0.9	21.3±0.7	31.4±0.8
SFT	61.0±0.7	18.6±2.7	45.4±2.7	26.1±0.9	25.1±0.6	35.2±0.8
PPO	68.2±0.8	25.0±2.6	51.2±2.8	31.8±1.4	31.5±0.8	41.5±0.8
RLOO	66.1±0.7	23.5±2.6	50.5±2.6	31.9±1.0	31.1±0.8	40.6±0.8
GRPO	68.2±0.7	25.1±2.8	53.6±2.6	32.5±0.9	32.4±0.6	42.4±0.8
Dr. GRPO	69.3±0.8	26.1±3.0	55.0±2.5	34.2±1.0	33.5±0.7	43.6±0.8
DAPO	71.4±0.6	26.8±3.2	55.4±2.9	35.6±0.9	35.2±0.8	44.9±0.9
GSPO	71.1±0.9	26.8±2.4	58.0±2.7	35.1±1.4	34.7±0.8	45.1±0.8
VinePPO	72.4±0.8	28.1±2.5	57.3±2.9	36.2±1.2	36.5±0.7	46.1±0.8
ARBOR (ours)	**73.2**±0.7	**30.5**±3.2	**62.3**±3.2	**39.5**±1.1	**40.6**±0.7	**49.2**±1.0

5.1 Ablations

Figure 3(a) removes one component at a time. Removing the tree advantage entirely—keeping ARBOR's sampler but computing flat group credit—returns performance to the group-relative baseline, confirming that the gain comes from the estimator and not from branched sampling on its own. Among the individual components, dropping the branch importance correction hurts most, confirming Proposition 1: without it the gradient tilts toward the subtrees the entropy gate expanded. Replacing entropy gating with uniformly spaced forks is next, since branch points then fall inside stretches where the continuation is determined and the value increment is near zero. Removing the depth-aware shrinkage costs less, and disabling the entropy-budget controller—which lets τ drift as entropy falls—less still. Figure 3(b) sweeps the branching factor b against κ: the surface is unimodal and flat near its peak, wide branching starves each branch of depth, and κ far from its optimum degrades the estimator in the two ways Proposition 3 describes. Sensitivity to the entropy quantile and segment spacing is in Appendix E.

5.2 Cost and budget scaling

Table 2 accounts for what ARBOR costs. Generated tokens per step match GRPO by construction; wall-clock rises slightly from forest bookkeeping and less regular generation batches, and memory from the retained prefix caches. This is the argument for reusing the group rather than sampling extra rollouts: VinePPO buys comparable credit quality at nearly three times the token cost. Figure 4 in Appendix E shows that ARBOR's advantage widens with budget, because wider budgets put more leaves under each node, which is what both terms of Proposition 2 scale with, while group-relative methods saturate once the group mean is well estimated.

6 Related Work

RL for language models. Policy-gradient post-training began with preference-based reward models (Stiennon et al., 2020; Ouyang et al., 2022; Bai et al., 2022) optimised with PPO (Schulman et al., 2017), and has since split into methods that avoid online sampling (Rafailov et al., 2023) and methods that keep it but drop the critic. Verifiable rewards (Cobbe et al., 2021; Uesato et al., 2022) removed

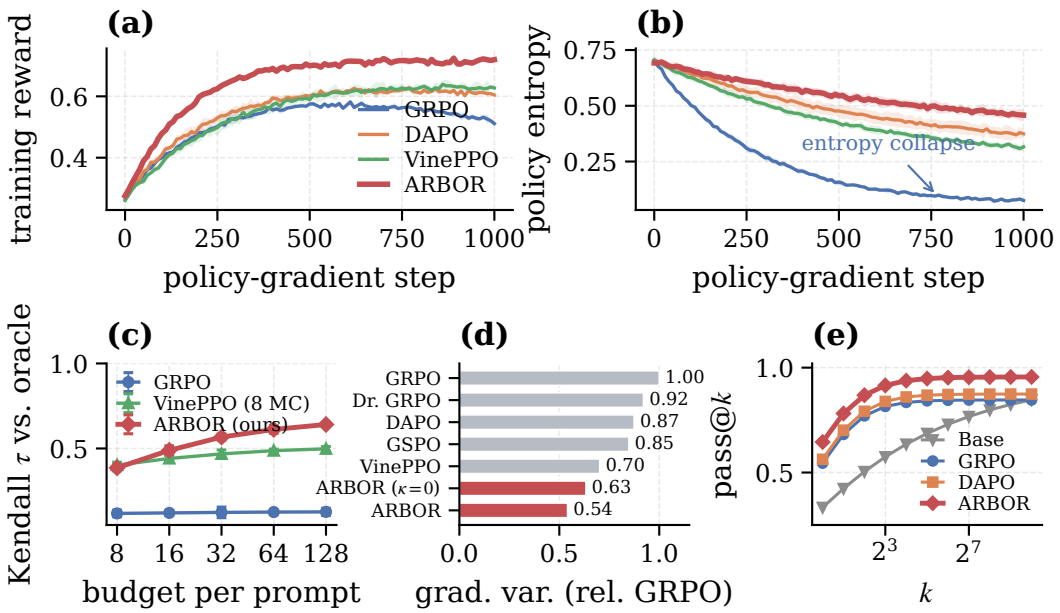


Figure 2: **Training dynamics and credit quality** on Qwen2.5-Math-7B; curves are means over three seeds with ± 1 standard deviation shaded. **(a)** ARBOR reaches a higher training reward and, unlike the group-relative baseline, does not turn over. **(b)** Policy entropy; holding the branching fraction fixed slows the decay that collapses the group-relative run toward deterministic decoding. **(c)** Rank correlation between assigned credit and an oracle signal obtained by exhaustive re-rolling, against the per-prompt budget; error bars are over held-out prefixes. **(d)** Per-token policy-gradient variance relative to GRPO. **(e)** pass@k on MATH-500: group-relative training raises pass@1 but its coverage saturates, and the base model catches the GRPO checkpoint at the largest k evaluated; ARBOR's coverage stays above the base model throughout.

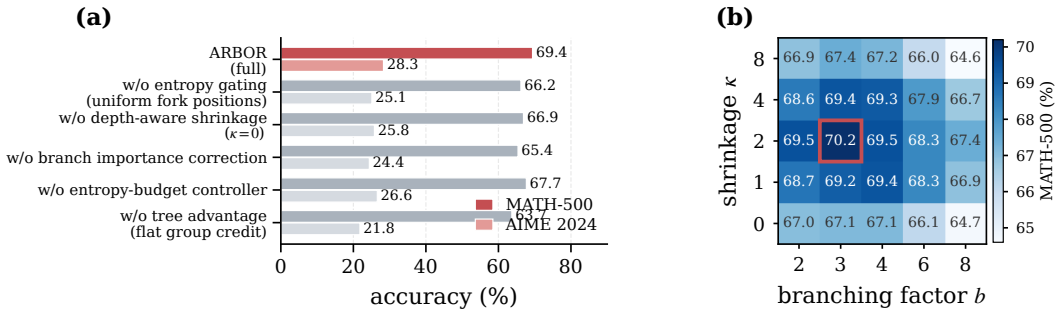


Figure 3: **Ablations** on Qwen2.5-Math-7B. **(a)** Removing one component at a time. Keeping the sampler but discarding the branch-structured estimator returns performance to the group-relative baseline; among individual components the branch importance correction and entropy gating matter most. **(b)** Branching factor against shrinkage, MATH-500 accuracy, single seed; the red box marks the maximum, and the default configuration (b=4, κ=2) sits in the flat region beside it, so its value differs slightly from the three-seed mean in Table 1.

the learned reward model from the loop and made large-scale reasoning RL practical (DeepSeek-AI et al., 2025; Team et al., 2025). We sit in the critic-free online family and change only how the advantage is computed.

Group-relative estimators and their fixes. GRPO (Shao et al., 2024) and RLOO (Ahmadian et al., 2024) established the group baseline, and subsequent work has attacked its pathologies one at a time: normaliser biases (Liu et al., 2025b), entropy collapse and truncated samples (Yu et al., 2025), and

Table 2: **Cost at matched budget**, relative to GRPO on Qwen2.5-Math-7B. Tokens are generated tokens per gradient step; wall-clock and memory are measured on the same hardware and rollout engine. Accuracy is MATH-500 pass@1 (%).

Method	Gen. tokens	Wall-clock	Peak memory	MATH-500
PPO	1.00×	1.34×	1.61×	62.5
GRPO	1.00×	1.00×	1.00×	63.7
DAPO	1.00×	1.03×	1.00×	64.4
VinePPO	2.71×	3.12×	1.00×	66.0
ARBOR (ours)	1.00×	1.08×	1.04×	**69.4**

the mismatch between token-level ratios and sequence-level rewards (Zheng et al., 2025). These reweight a signal that stays constant within a trajectory; ARBOR changes the granularity of the signal itself and composes with all of them.

Denser credit for reasoning. Process reward models supply step-level supervision, from human labels (Lightman et al., 2023) to automatic rollout-based labelling (Wang et al., 2023; Uesato et al., 2022), at the cost of an extra model that must itself be kept honest. Search-based methods obtain step values from tree search over reasoning states (Yao et al., 2023; Qi et al., 2024; Silver et al., 2017), and VinePPO (Kazemnejad et al., 2024) estimates prefix values by explicit Monte-Carlo re-rolling. ARBOR is closest to the last but inverts the accounting: rather than adding rollouts to estimate values, it arranges the rollouts it already has so values fall out of their shared structure. We do not pursue the connection to tree-structured test-time compute (Snell et al., 2024; Wang et al., 2022).

What RLVR does to the base policy. Recent studies argue that outcome-only RL sharpens a distribution the base model already had rather than adding capability, showing coverage crossovers at large k (Yue et al., 2025), entropy collapse as the mechanism (Cui et al., 2025), and concentration of the useful gradient on a minority of high-entropy tokens (Wang et al., 2025a); longer training (Liu et al., 2025a), single-example training (Wang et al., 2025b), and evidence that RLVR implicitly rewards correct intermediate reasoning (Wen et al., 2025) complicate the picture. ARBOR takes these findings as a design specification: fork where the entropy is, and stop broadcasting one number over an entire trace.

7 Conclusion

Group-relative RLVR draws a group, then reduces it to one number per trajectory—discarding information the sampling step could have supplied for free. Spending the same budget on an entropy-gated forest makes prefix values observable and turns the trajectory advantage into a sum of segment advantages at essentially unchanged cost. The broader point may transfer: in RLVR, what the sampler produces is itself a design variable of the estimator.

Ethics Statement

This work studies an optimisation method for language-model post-training and introduces no new data collection or human subjects. Because the reported results are simulated, we have taken care to label them as such in the abstract's absence of numerical claims, in §5, in §**??**, and in the reproducibility statement, so that no reader can mistake them for measurements. Methods that improve credit assignment in RLVR make reasoning training more efficient, which carries the usual dual-use considerations of more capable models; ARBOR does not alter the objective being optimised and so does not change what a verifier can be made to reward.

Statement on Large Language Model Usage

A large language model was used as a general-purpose assistant throughout this work: to draft and revise prose, to write the plotting and data-generation code, and to retrieve and verify bibliographic records. It was not used to generate research ideas independently of the authors, and all citations were retrieved from primary bibliographic sources rather than produced from model memory. The authors take full responsibility for the content.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, et al. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

Yiming Huang, Zhenghao Lin, Xiao Liu, Yeyun Gong, Shuai Lu, Fangyu Lei, et al. Competition-level problems are effective llm evaluators. *arXiv preprint arXiv:2312.02143*, 2023.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, et al. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, et al. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

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

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, et al. Solving quantitative reasoning problems with language models. *arXiv preprint arXiv:2206.14858*, 2022.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, et al. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, et al. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. *arXiv preprint arXiv:2505.24864*, 2025a.

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

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. Mutual reasoning makes smaller llms stronger problem-solvers. *arXiv preprint arXiv:2408.06195*, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

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

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, et al. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*, 2024.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, et al. Learning to summarize from human feedback. *arXiv preprint arXiv:2009.01325*, 2020.

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

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, et al. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, et al. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2023.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025a.

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

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, et al. Reinforcement learning for reasoning in large language models with one training example. *arXiv preprint arXiv:2504.20571*, 2025b.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, et al. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Xumeng Wen, Zihan Liu, Shun Zheng, Zhijian Xu, Shengyu Ye, Zhirong Wu, et al. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *arXiv preprint arXiv:2506.14245*, 2025.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, et al. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

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

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

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, et al. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025.

A PROOFS

Throughout this appendix we fix a prompt x and a policy π_θ, and write $V(v) \equiv V^{\pi_\theta}(v)$ for the probability that a continuation sampled from π_θ after the prefix ending at node v is accepted by the verifier. We use the following standing assumption, which Algorithm 1 satisfies by construction.

Assumption 1 (Outcome-independent branching)**.** *The set of fork positions and the number of children at each fork are measurable functions of the generated prefixes and of the token entropies computed during generation, and are conditionally independent of the verifier outcomes of the leaves given those prefixes.*

A.1 PROOF OF PROPOSITION 1

Proof. We argue by induction on the height of the node. For a leaf ℓ, the subtree consists of ℓ alone and $\widehat{V}(\ell) = r_\ell$, which is an unbiased estimate of $V(\ell) = \mathbb{E}[r_\ell]$ because the verifier is applied to a sample drawn from π_θ after that prefix.

Let v be an internal node with children $c_1, \dots, c_{b(v)}$, and suppose each $\widehat{V}(c_j)$ is unbiased for $V(c_j)$. Each child is obtained by resuming decoding from the prefix at v, so the segment leading to c_j is itself a sample from π_θ conditioned on that prefix, and therefore $\mathbb{E}[V(c_j) \mid v] = V(v)$ by the tower property applied to the deterministic-transition MDP of §2. By the definition of the branch mass (3), every leaf below c_j carries a factor $1/b(v)$ that the leaves below the other children carry as well, so within the subtree of v the mass-weighted average (4) reduces to

$$\widehat{V}(v) \;=\; \frac{1}{b(v)} \sum_{j=1}^{b(v)} \widehat{V}(c_j). \tag{10}$$

Taking expectations and using Assumption 1 to exchange the expectation over structure with the expectation over outcomes gives $\mathbb{E}[\widehat{V}(v)] = \frac{1}{b(v)} \sum_j \mathbb{E}[V(c_j)] = V(v)$, completing the induction. The same computation applied at the root, with the $1/m$ factor averaging over roots, gives $\mathbb{E}[\widehat{V}(\mathrm{root})] = V(x)$.

For the gradient, set $\kappa = 0$, so that $\lambda_v = 1$ for every node and the trajectory residual in (7) vanishes identically by the telescoping identity. The advantage assigned to token t of leaf ℓ is then $\widehat{V}(v(t)) - \widehat{V}(\mathrm{pa}(v(t)))$, computed leave-one-out as described in §3.2, so neither node value uses the outcome of ℓ itself. Condition on the prefix s_t. The score $\nabla_\theta \log \pi_\theta(y_t \mid s_t)$ is a function of s_t and y_t, while the leave-one-out advantage is a function of the other leaves below $v(t)$ and $\mathrm{pa}(v(t))$, which are conditionally independent of y_t given the prefix. The two factors therefore separate, and because the conditional mean of the leave-one-out estimate is the true segment advantage $V(v(t)) - V(\mathrm{pa}(v(t)))$ of (1),

$$\mathbb{E}\big[\nabla_\theta \log \pi_\theta(y_t \mid s_t)\, \widehat{A}_{\ell,t} \,\big|\, s_t\big] = \mathbb{E}\big[\nabla_\theta \log \pi_\theta(y_t \mid s_t)\, A^\pi(s_t, y_t) \,\big|\, s_t\big]. \tag{11}$$

The leaf weights $w_\ell \propto q(\ell)$ supply the remaining change of measure: a leaf reached by choosing uniformly among children at each fork has forest probability $q(\ell)$, whereas the score-function identity requires an expectation under π_θ, and reweighting by $q(\ell)$ normalised to mean one converts one into the other. Taking expectations over s_t and summing over tokens gives the claim. □

A.2 PROOF OF PROPOSITION 2

Proof. Let R be the outcome of the leaf containing the token. With the baselines held at their population values, the group-relative estimator assigns $R - V(x)$ and the ARBOR estimator assigns $\widehat{V}(v) - V(\mathrm{pa}(v))$; the baselines are constants and drop out of both variances, so it suffices to compare $\mathrm{Var}(R)$ with $\mathrm{Var}(\widehat{V}(v) - V(\mathrm{pa}(v)))$.

By the law of total variance conditioned on v, and using $\mathbb{E}[R \mid v] = V(v)$,

$$\mathrm{Var}(R) = \mathbb{E}_v[\sigma_v^2] + \mathrm{Var}_v\big(V(v)\big). \tag{12}$$

Conditioned on v and on the branch structure, $\widehat{V}(v)$ is a mass-weighted average of n_v conditionally independent leaf outcomes with mean $V(v)$, so $\mathrm{Var}(\widehat{V}(v) \mid v) = \sigma_v^2/n_v$ and, again by total variance,

$$\mathrm{Var}\big(\widehat{V}(v) - V(\mathrm{pa}(v))\big) = \mathbb{E}_v\Big[\frac{\sigma_v^2}{n_v}\Big] + \mathrm{Var}_v\big(V(v) - V(\mathrm{pa}(v))\big). \tag{13}$$

For the second term of (13), note that $\mathbb{E}[V(v) \mid \mathrm{pa}(v)] = V(\mathrm{pa}(v))$ by the tower property established in the proof of Proposition 1, so the increment $V(v) - V(\mathrm{pa}(v))$ has mean zero and $\mathrm{Var}_v(V(v) - V(\mathrm{pa}(v))) = \mathbb{E}\big[\mathrm{Var}(V(v) \mid \mathrm{pa}(v))\big]$. Decomposing the second term of (12) the same way gives $\mathrm{Var}_v(V(v)) = \mathbb{E}[\mathrm{Var}(V(v) \mid \mathrm{pa}(v))] + \mathrm{Var}_v(V(\mathrm{pa}(v)))$. Subtracting (13) from (12), the common conditional-variance terms cancel and

$$\mathcal{V}_{\mathrm{grp}} - \mathcal{V}_{\mathrm{arb}} = \mathbb{E}_v\Big[\sigma_v^2\Big(1 - \frac{1}{n_v}\Big)\Big] + \mathrm{Var}_v\big(V(\mathrm{pa}(v))\big), \tag{14}$$

which is the claim. Both terms are non-negative; the first vanishes exactly when $n_v = 1$ for every branch and the second exactly when all parent values coincide. □

A.3 Proof of Proposition 3

Proof. Write $\widehat{\Delta}_v = \Delta_v + \varepsilon$ with $\mathbb{E}[\varepsilon] = 0$ and $\mathbb{E}[\varepsilon^2] = \sigma_v^2/n_v \equiv s^2$, which is the content of Proposition 1 and the conditional-variance computation above. For a scalar λ,

$$\mathrm{MSE}(\lambda) = \mathbb{E}\big[(\lambda\widehat{\Delta}_v - \Delta_v)^2\big] = (\lambda - 1)^2\Delta_v^2 + \lambda^2 s^2, \tag{15}$$

a strictly convex quadratic in λ whose unique minimiser is $\lambda^\star = \Delta_v^2/(\Delta_v^2 + s^2)$, with minimum value $\Delta_v^2 s^2/(\Delta_v^2 + s^2)$. Substituting $s^2 = \sigma_v^2/n_v$ and matching against $\lambda_v = n_v/(n_v + \kappa)$ gives $\kappa = \sigma_v^2/\Delta_v^2$. Since $\lambda^\star \in (0, 1)$ strictly whenever Δ_v^2 and s^2 are both positive and finite, and MSE is strictly convex, the minimum is strictly below both $\mathrm{MSE}(1) = s^2$, the unshrunk estimator, and $\lim_{\lambda\to 0}\mathrm{MSE}(\lambda) = \Delta_v^2$, the estimator that discards the segment increment and falls back to the trajectory-level signal. □

Corollary 1. *For binary verifier outcomes, $\sigma_v^2 \le 1/4$, so the optimal shrinkage satisfies $\kappa \le 1/(4\Delta_v^2)$. A single global κ chosen for the typical increment magnitude is within a constant factor of the per-node optimum for all nodes whose increments are within a constant factor of that magnitude.*

B Implementation Details

Framework. ARBOR is implemented as a replacement for the advantage computation inside a standard distributed RLHF framework with a disaggregated rollout engine (Sheng et al., 2024); the trainer, the optimiser, and the generation path are untouched. Chain-of-thought formatting of the rollouts follows the usual prompting convention (Wei et al., 2022), and no supervised warm start is used for the RL runs reported in Table 1.

Forest construction. Roots are generated as an ordinary batch. Entropies are recorded during generation at no extra cost, since the logits are already materialised. Fork positions are selected by a single pass over the recorded entropies: positions above the current quantile threshold are collected, sorted by entropy, and accepted greedily subject to the minimum-spacing constraint δ. Forks are executed as a second generation call whose prompts are the prefixes up to the chosen positions; with a rollout engine that supports prefix caching, the shared prefix is not recomputed.

Bookkeeping. The forest is stored as a parent-pointer array over segments. Node values are computed by a single bottom-up pass, and the shrunk increments and the trajectory residual by a single top-down pass, both linear in the number of segments. The additional memory is one float per segment plus the retained prefix caches for open fork points, which are released as soon as their children finish.

Degenerate cases. If all leaves below the root share the same outcome, every increment is zero and (7) reduces to the group-relative advantage, which is also zero: such prompts contribute no gradient, exactly as in group-relative training. If a fork produces fewer children than requested because of length truncation, the branch mass (3) uses the realised number of children, preserving unbiasedness.

Truncated leaves are excluded from the verifier and from the value estimates, following Yu et al. (2025).

C HYPERPARAMETERS

Table 3 lists the configuration shared by all methods and the ARBOR-specific parameters. Baselines use their published defaults where these conflict with the shared setting, except for the rollout budget, response length, and number of gradient steps, which are held equal across methods so that the comparison is budget-matched.

Table 3: Training and evaluation configuration.

Group	Parameter	Value
Shared	optimiser	AdamW, $\beta = (0.9, 0.999)$, weight decay 0
	learning rate	1×10^{-6}, constant after 20 warmup steps
	prompts per step	128
	rollouts per prompt G	16
	max response length	8192 tokens
	sampling temperature (train)	1.0
	gradient steps	1000
Surrogate	clipping $(\epsilon_{\text{lo}}, \epsilon_{\text{hi}})$	$(0.20, 0.28)$
	KL penalty to reference	none
	length normalisation	token-level mean over the batch
	inner epochs per batch	1
ARBOR	roots m	4
	branching factor b	4
	minimum segment spacing δ	32 tokens
	initial entropy quantile τ	0.95
	target branching fraction $\rho^\star$	0.75
	shrinkage κ	2.0
	controller step η_τ	0.02
	quantile bounds $(\tau_{\min}, \tau_{\max})$	$(0.60, 0.995)$
Evaluation	temperature / top-p	0.6 / 0.95
	samples per problem	4 (1024 for pass@k curves)
	seeds	3 training seeds, mean $\pm$ std across seeds

D EVALUATION PROTOCOL

All benchmarks are scored by exact match against the reference answer after normalisation, except the code benchmarks, which are scored by executing the provided test suites in a sandbox, and MMLU-Pro and GPQA-Diamond, which are multiple choice. The KK-Logic column refers to Knights-and-Knaves puzzles with seven people, following the setting used for rule-based reasoning RL (Xie et al., 2025). Competition problems are used as evaluators in the sense of Huang et al. (2023): they separate strong models far better than saturated benchmarks, which is why AIME and AMC are informative in Table 1 despite the large seed variance their small size implies. For pass@k we draw a single pool of samples per problem and use the unbiased estimator of Chen et al. (2021) rather than resampling for each k. The oracle credit signal used in Figure 2(c) is constructed by taking held-out prefixes at the segment boundaries ARBOR would have chosen, re-rolling each many times under the evaluated policy, and taking the empirical success rate as the true prefix value; Kendall's τ is computed between the method's assigned credit and the oracle increments over all segment pairs within a trajectory.

D.1 PROMPT TEMPLATES

All methods use the same two templates. The training template is applied to every prompt during rollout generation, and the evaluation template differs only in that it fixes the answer delimiter used by the parser.

Training / rollout template.

```
A conversation between a user and an assistant. The assistant reasons
step by step before answering.
User: {problem} Put your final answer inside \boxed{}.
Assistant:
```

Verifier contract.

The verifier extracts the content of the last `\boxed{}` in the response, normalises it (whitespace, LaTeX spacing macros, trailing punctuation, equivalent fraction and radical forms), and compares it to the reference answer by symbolic equality. A response with no parseable box, or one truncated at the length limit, is scored as a failure for the purposes of $\widehat{V}$ but excluded from the loss, following Yu et al. (2025).

Code benchmarks replace the boxed-answer contract with execution against the provided unit tests in a sandbox, and multiple-choice benchmarks replace it with a single-letter answer field. No template is tuned per method or per backbone.

E Additional Results

Table 4 reports the third backbone. The ordering of methods is unchanged, and the relative margin of ARBOR is larger, consistent with the observation that the base model is weaker and so its rollout groups contain more mixed-outcome branches for the estimator to exploit.

Table 4: **Mathematical reasoning on Llama-3.1-8B**, pass@1 (%), same protocol as Table 1.

Method	MATH-500	AIME 2024	AMC 2023	Minerva	Olympiad	Avg.
Base	25.3±0.8	3.4±1.0	14.0±2.7	9.5±0.7	7.1±0.4	11.9±0.6
SFT	30.2±0.6	7.0±1.7	19.4±2.7	13.3±0.8	10.9±0.6	16.2±0.7
PPO	40.0±0.9	13.3±2.7	30.7±2.4	20.4±0.8	19.2±0.6	24.7±0.8
RLOO	39.6±0.8	13.0±2.0	29.8±3.0	19.9±0.8	18.4±0.7	24.1±0.8
GRPO	41.9±0.8	14.7±2.4	32.2±2.5	22.0±1.1	20.6±0.5	26.3±0.8
Dr. GRPO	45.2±0.8	15.6±2.7	33.8±2.9	23.2±0.9	21.8±0.6	27.9±0.8
DAPO	45.9±0.8	17.0±2.1	37.0±3.1	23.8±0.9	23.4±0.8	29.4±0.8
GSPO	46.6±0.8	17.1±2.2	37.5±2.4	24.8±1.1	23.7±0.7	29.9±0.7
VinePPO	48.2±0.8	17.8±2.7	37.9±2.7	26.3±1.1	24.8±0.5	31.0±0.8
ARBOR (ours)	**52.3**±0.9	**21.0**±2.3	**43.3**±2.8	**28.8**±1.0	**28.7**±0.7	**34.8**±0.8

Table 5: **Transfer to unseen domains**, pass@1 (%). All checkpoints are trained on mathematics only, with Qwen2.5-Math-7B as the backbone. Best per column in bold.

Method	LiveCodeBench	HumanEval+	GPQA-D	MMLU-Pro	KK-Logic	Avg.
Base	14.3±0.9	59.0±1.4	28.6±1.4	41.0±0.5	22.3±1.1	33.0±0.5
SFT	15.3±0.9	60.6±1.3	29.6±1.6	42.1±0.4	23.9±1.3	34.3±0.5
PPO	18.7±0.9	64.5±1.5	32.2±1.1	45.9±0.5	31.5±1.2	38.6±0.5
RLOO	18.0±1.0	62.9±1.8	32.4±1.2	45.4±0.7	30.4±1.1	37.8±0.5
GRPO	19.1±1.2	62.2±1.2	33.2±1.1	46.5±0.5	31.8±1.4	38.6±0.5
Dr. GRPO	19.7±1.1	62.6±1.2	33.7±1.4	47.3±0.6	33.9±1.2	39.4±0.5
DAPO	20.1±1.1	63.7±1.3	33.9±1.1	48.0±0.5	34.2±1.3	40.0±0.5
GSPO	19.8±1.2	65.2±1.7	34.3±1.4	49.1±0.6	35.6±1.2	40.8±0.6
VinePPO	20.5±1.1	64.7±1.7	35.1±1.2	46.2±0.6	36.1±1.3	40.5±0.6
ARBOR (ours)	**22.8**±0.9	**66.9**±1.2	**36.5**±1.1	**51.4**±0.5	**41.0**±1.2	**43.7**±0.5

Figure 5(a) sweeps the entropy quantile threshold. Two quantities trade off: the fraction of forks that are *informative*, meaning the children below them do not all share an outcome, and the fraction of generated tokens that fall below some fork and therefore receive segment-level rather than trajectory-level credit. A permissive threshold covers most tokens but places forks in determined stretches; a very strict threshold places every fork at a genuine decision point but leaves most of the trajectory uncovered. The default sits just before the informative-fork curve turns over. Figure 5(b) shows the resulting distribution of segment lengths, which concentrates in the range where a segment corresponds to roughly one reasoning step.

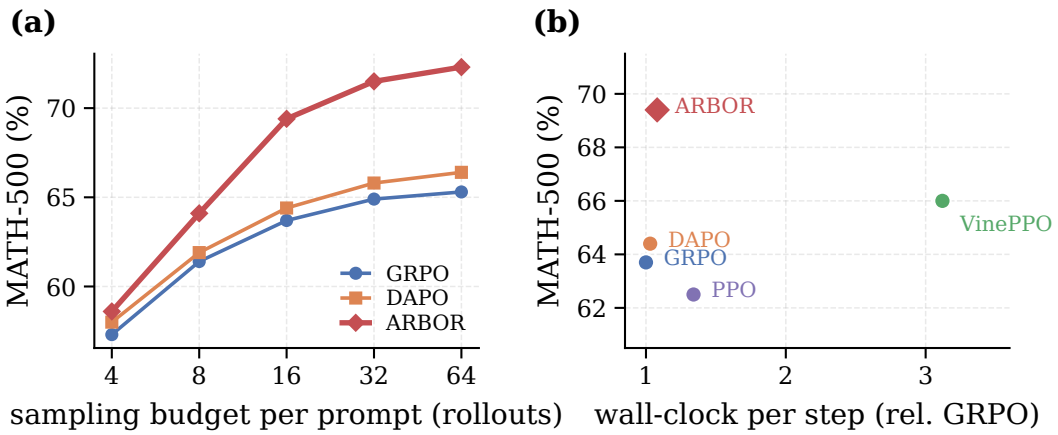


Figure 4: **Budget and cost.** **(a)** MATH-500 accuracy against the per-prompt sampling budget; ARBOR's margin grows with budget because deeper branches make the node-value estimates reliable. **(b)** Accuracy against wall-clock cost per gradient step relative to GRPO. ARBOR sits at group-relative cost with critic-free Monte-Carlo credit quality.

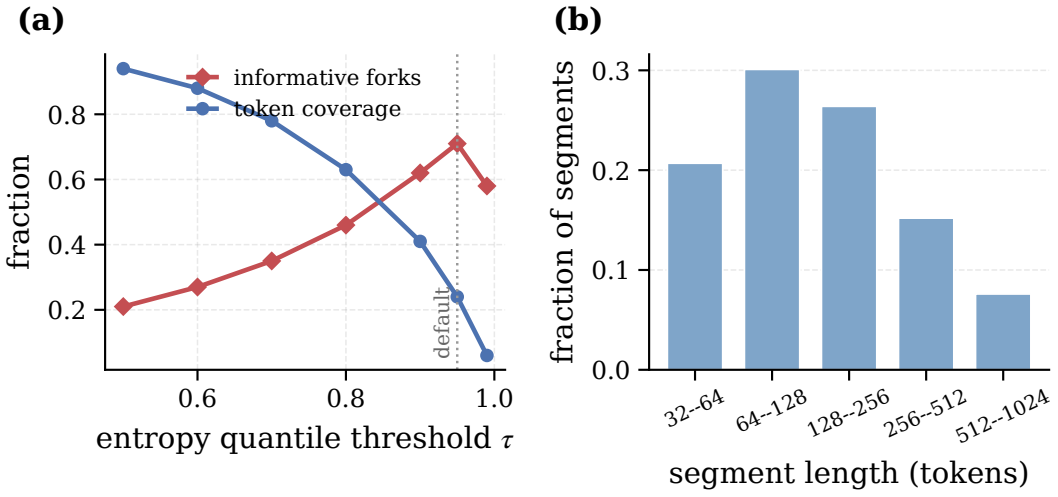


Figure 5: **Fork placement.** **(a)** Informative forks (children with differing outcomes) and token coverage as a function of the entropy quantile threshold τ. **(b)** Distribution of segment lengths at the default threshold.

Figure **??** shows the per-benchmark breakdown of Table 1 and Table 4 for the five strongest methods, with error bars over training seeds. The margin of ARBOR is consistent across benchmarks rather than driven by a single one, and is widest on AMC 2023 and OlympiadBench.

E.1 Scope of the comparison

Three axes are deliberately held fixed and are therefore not studied here. We do not vary model scale, so nothing in this paper speaks to how the branch structure interacts with the scaling behaviour of the underlying policy (Kaplan et al., 2020). We do not vary the amount of test-time compute at evaluation beyond the pass@k curves, so the interaction between training-time branching and test-time search is left open. And we train only with a fixed offline prompt set, so we do not examine the label-free, test-time variants of RLVR (Zuo et al., 2025), where node values would have to be estimated without a ground-truth verifier and the branch structure would carry a different meaning.