HAVER-GRPO: HAZARD-AWARE ROLLOUT ALLOCATION FOR EFFICIENT RLVR

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards can waste substantial rollout compute on prompt groups whose responses are all incorrect and yield no group-relative learning signal. We introduce HAVER-GRPO, a sequential allocation strategy that decides whether an all-wrong group merits another rollout. HAVER estimates verifier-success hazard and response-token cost, then allocates fresh rollouts by marginal recovery yield under a shared budget price. The first verified response is inserted into a fixed-size GRPO group, while every attempted token is charged to the compute budget. We analyze recovery probability, expected cost, and sensitivity to estimation error. Across five mathematical reasoning benchmarks and three training seeds, our main experiments achieve mean pass@1(avg@8) accuracies of 49.7% and 56.5% for the 1.5B and 7B models, respectively. Tokens-to-target decrease by 34.7% and 21.4% relative to uniform GRPO, demonstrating improved training efficiency at the same accuracy targets.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) has become a practical way to improve language-model reasoning without fitting a learned reward model (Cobbe et al., 2021; Hendrycks et al., 2021). In group relative policy optimization (GRPO), several responses are sampled for each prompt and their verifier outcomes are normalized within the group. This critic-free construction is attractive because it is simple and memory efficient, but it creates a sharp failure mode: when every response in a group is rejected, the centered group reward is zero and the group contributes no ordinary policy-gradient direction.

The usual response is to sample a larger fixed group or to oversample prompts and filter groups after generation. Both strategies pay for responses before knowing whether they will be useful. Fresh-response histories and response lengths vary, so counting calls is not equivalent to counting compute. This suggests a sequential allocation question inside the prompt group: after observing the current failures, is one more response worth its token cost?

We study this question with HAVER-GRPO (Hazard-Aware Verifier-Grounded Rollout Allocation for GRPO). The method starts with a fixed group of K on-policy responses. Groups containing a verifier-positive response follow the configured GRPO path unchanged. An all-wrong group enters a sequential controller. At each round, the controller updates a group-specific posterior for the next-response success hazard and a token-cost estimate. It continues sampling only when the expected marginal number of recovered groups per policy token exceeds a shared shadow price, and it stops at a predeclared fresh-rollout cap. If a fresh response is correct, it replaces one failed member; the optimizer still sees a group of size K. No hidden repair label or future outcome is used by the controller.

The central distinction from prior adaptive allocation is granularity. VIP predicts prompt success probabilities and allocates a batch-level rollout budget across prompts (Nguyen et al., 2026). SARA makes an early commit/abandon/continue decision while assembling effective groups (Nomand et al., 2026). HAVER-GRPO instead keeps a fixed GRPO group and asks whether an already observed all-wrong group merits within-prompt recovery sampling, using both a next-success hazard and the measured token cost. We evaluate this design through benchmark accuracy and generated-token accounting.

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.

This paper makes three scoped contributions:

- **A token-costed recovery formulation.** We isolate the all-wrong event in fixed-K GRPO and formulate recovery as a finite-horizon sequential allocation problem. The resulting controller uses only deployable observations: the initial verifier outcomes, the observed fresh-response history, and predicted non-padding token cost.
- **A hazard-aware fixed-interface mechanism.** We derive a marginal recovery-per-token index, couple group-level decisions through a shared budget price, and reconstruct the training group by replacing one failed slot with the first verifier-positive response. This design changes rollout allocation while preserving the optimizer-facing group size and the configured group-relative objective. We additionally characterize recovery probability, expected capped cost, and sensitivity to hazard and cost-estimation error.
- **An auditable 300-update evaluation.** We compare accuracy and generated policy-token costs against Uniform GRPO, DAPO, and Fixed-cap recovery under the same training horizon. We also report tokens-to-target, recovery accounting, and hazard and cost component ablations.

The 300-update results are averaged over three training seeds; we do not claim state-of-the-art performance.

2 Related Work

Verifiable rewards and group optimization. RLVR trains on automatically checkable outcomes in mathematics and code (Cobbe et al., 2021; Hendrycks et al., 2021; Shao et al., 2024; DeepSeek-AI, 2025). GRPO and related estimators form advantages from sampled rewards (Shao et al., 2024; Ahmadian et al., 2024; Liu et al., 2025). HAVER retains an exact outcome verifier and changes sampling before the configured group-relative update.

Filtering homogeneous groups. DAPO oversamples candidate groups and retains those with nonzero reward variance (Yu et al., 2025). Prompt-selection and advantage-collapse methods instead change batch composition or the optimization signal (Bae et al., 2025; Lin et al., 2025; He et al., 2026; Zheng et al., 2025). HAVER conditions on an observed all-wrong group and tests whether another on-policy draw is worth its token cost.

Adaptive allocation. VIP, HORA, and VIGOR allocate rollouts using prompt predictions, posterior hit utility, or group reward variance (Nguyen et al., 2026; Wang et al., 2026; Jiang et al., 2026). SARA uses sequential commit/abandon/continue decisions while assembling effective groups (Nomand et al., 2026). HAVER instead starts after an all-wrong initial group, ranks fresh responses by recovery yield per token, and returns exactly K responses to the optimizer.

Token cost and sequential decisions. Reasoning length varies across responses (Aggarwal & Welleck, 2025); HAVER therefore charges non-padding tokens for every attempted rollout. Its controller draws on hazard models and sequential allocation under a shared resource constraint (Cox, 1972; Wald, 1945; Gittins, 1979). The quality of its hazard and cost estimates affects which all-wrong groups receive another response.

3 Preliminaries and Problem Setup

3.1 Reinforcement learning with verifiable rewards

Let $\mathcal{D} = \{q_1, \ldots, q_Q\}$ be a training set of prompts. For each prompt q, a behavior policy π_{old} samples K responses $o_{1:K}$, and a rule-based verifier assigns outcome reward $R_j = v(q, o_j) \in \{0, 1\}$. For token $o_{j,t}$, define the importance ratio

$$\rho_{j,t}(\theta) = \frac{\pi_\theta(o_{j,t} \mid q, o_{j,<t})}{\pi_{\text{old}}(o_{j,t} \mid q, o_{j,<t})}. \tag{1}$$

Group-based RLVR inherits the likelihood-ratio policy gradient from REINFORCE and the trust-region motivation of TRPO and PPO (Williams, 1992; Schulman et al., 2015a; 2017). Unlike actor–critic methods that estimate advantages with a learned value function, for example through

generalized advantage estimation (Schulman et al., 2015b), GRPO constructs a response-group baseline directly from sampled verifier rewards:

$$J(\theta) = \mathbb{E}_{q\sim\mathcal{D},\, o_{1:K}\sim\pi_{\text{old}}}\left[\frac{1}{K}\sum_{j=1}^{K}\frac{1}{|o_j|}\sum_{t=1}^{|o_j|}\left(\ell_{j,t}^{\text{clip}}(\theta)\right)\right]. \tag{2}$$

Here the clipped token term is

$$\ell_{j,t}^{\text{clip}}(\theta) = \min\Big(\rho_{j,t}\widetilde{A}_j, \text{clip}(\rho_{j,t}, 1-\epsilon, 1+\epsilon)\widetilde{A}_j\Big) - \beta D_{\text{KL}}(\pi_\theta\|\pi_{\text{ref}}). \tag{3}$$

where $\widetilde{A}_j$ is shared by all tokens in response o_j. Exact outcome verifiers have been used for mathematical answers, executable code, and other tasks with deterministic checks (Cobbe et al., 2021; Chen et al., 2021; Shao et al., 2024). As in recent reasoning-oriented RLVR systems, β may be set to zero; our allocation rule does not depend on that choice. HAVER evaluates the same surrogate on Y'_g; only the sampled group changes.

3.2 Group-relative advantage estimation

GRPO standardizes each reward against its response group (Shao et al., 2024):

$$\widetilde{A}_j^{\text{GRPO}} = \frac{R_j - \bar{R}}{\sqrt{K^{-1}\sum_{k=1}^{K}(R_k-\bar{R})^2 + \delta}}, \qquad \bar{R} = K^{-1}\sum_{k=1}^{K} R_k, \tag{4}$$

where $\delta > 0$ prevents division by zero. Dr. GRPO removes the reward standard-deviation normalization (Liu et al., 2025), whereas RLOO uses a leave-one-out baseline (Ahmadian et al., 2024):

$$\widetilde{A}_j^{\text{Dr}} = R_j - \bar{R}, \qquad \widetilde{A}_j^{\text{RLOO}} = R_j - \frac{1}{K-1}\sum_{k\neq j} R_k. \tag{5}$$

All three estimators vanish when every reward in the group is identical. DAPO addresses the resulting inefficiency with dynamic sampling: it oversamples candidate prompts, retains groups satisfying $0 < \sum_j R_j < K$, and combines this filtering with asymmetric clipping and token-level loss aggregation (Yu et al., 2025). This improves the fraction of effective groups, but full candidate groups are generated before their saturation is known.

3.3 All-wrong recovery as a budgeted problem

For group g, define the all-wrong event

$$H_g = \left\{\sum_{j=1}^{K} R_{g,j} = 0\right\}. \tag{6}$$

On H_g, Equations 4–5 give $\widetilde{A}_{g,j} = 0$ for every response, hence the reward-dependent part of $\nabla_\theta J_g(\theta)$ is zero. Let $Z_{g,t} \in \{0,1\}$ be the verifier outcome of fresh attempt t, $c_{g,t}$ its non-padding response-token cost, and $\tau_g \le C_{\max}$ the stopping time. Under a shared fresh-token budget $T_{\max}$, the allocation target is

$$\begin{aligned} \max_{\{\tau_g\}} \quad & \mathbb{E}\left[\sum_{g:H_g}\mathbb{I}\Big\{\max_{1\le t\le\tau_g} Z_{g,t} = 1\Big\}\right] \\ \text{s.t.} \quad & \sum_{g:H_g}\sum_{t=1}^{\tau_g} c_{g,t} \le T_{\max}, \qquad 0 \le \tau_g \le C_{\max}. \end{aligned} \tag{7}$$

The objective counts recovered groups. The controller observes each verifier result before admitting another response (Wald, 1945; Gittins, 1979). The shared fresh-token budget applies to the second-stage decisions; reported policy tokens also include all initial responses and discarded failures:

$$T_{\text{policy}} = \sum_{r\in\mathcal{R}} \text{len}_{\text{nonpad}}(y_r), \tag{8}$$

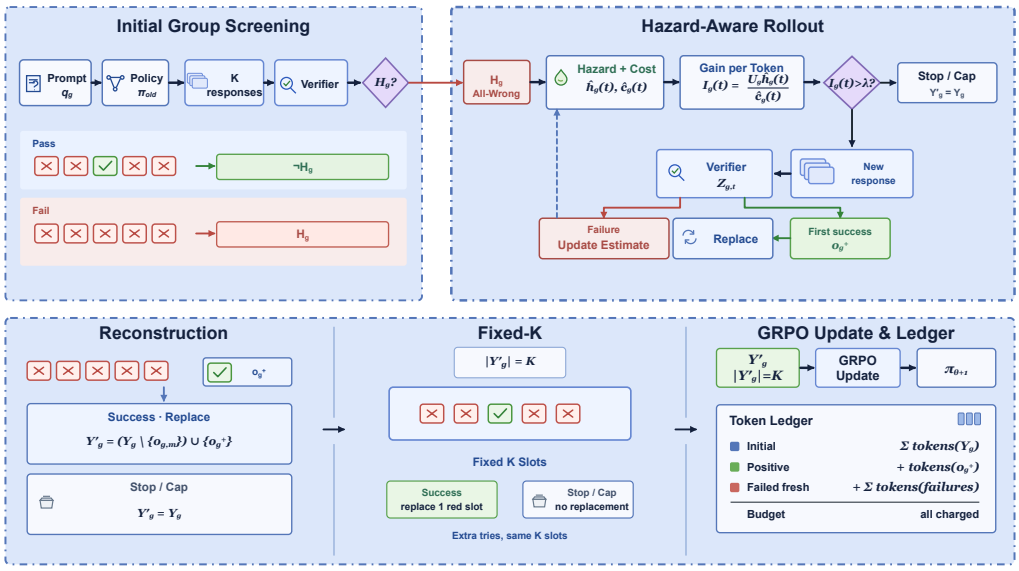


Figure 1: **HAVER-GRPO architecture.** Stage 1 screens a fixed-capacity group of K initial responses. Only an all-wrong group enters the hazard-aware allocation loop in Stage 2, where the next rollout is selected by estimated recovery yield per token. A verifier-positive fresh response replaces one failed slot; stopping leaves the slots unchanged. Stage 3 always constructs Y'_g with $|Y'_g| = K$ before the GRPO update, while every generated response remains charged in the policy-token ledger.

Prompt and padding tokens and verifier runtime are outside this metric.

Under the simplifying assumption that fresh responses for group g are conditionally independent with constant success probability p_g, a cap of C fresh attempts recovers the group with probability

$$P(\text{recover} \mid g, C) = 1 - (1 - p_g)^C. \tag{9}$$

For a policy that stops at its first success or the cap, the expected number of fresh calls is

$$\mathbb{E}[C_g \mid p_g, C] = \sum_{t=0}^{C-1} (1 - p_g)^t = \frac{1 - (1 - p_g)^C}{p_g}, \tag{10}$$

with $\mathbb{E}[C_g \mid 0, C] = C$. This is not the cost of the Fixed-cap baseline, which generates all C attempts; Appendix B gives the history-dependent token-cost accounting.

4 Hazard-Aware Verifier-Grounded Rollout Allocation

Figure 1 summarizes HAVER-GRPO. It leaves informative groups unchanged and applies a sequential recovery controller only on event H_g.

4.1 Sequential recovery state and prior

For an active group g, let $\mathcal{H}_{g,t} = (q_g, o_{g,1:K}, Z_{g,1:t}, c_{g,1:t})$ be the information available after t fresh attempts. The next-success hazard is

$$h_g(t) = \Pr(Z_{g,t+1} = 1 \mid \mathcal{H}_{g,t}, H_g). \tag{11}$$

We instantiate a Beta–Bernoulli model with a prior $p_g \sim \text{Beta}(a_g, b_g)$ at entry to the all-wrong route. The experiments initialize $a_g = b_g = 1$. If $s_{g,t}$ successes and $f_{g,t}$ failures have been observed in fresh attempts, then

$$p_g \mid \mathcal{H}_{g,t} \sim \text{Beta}(a_g + s_{g,t}, b_g + f_{g,t}), \qquad \widehat{h}_g(t) = \frac{a_g + s_{g,t}}{a_g + b_g + t}. \tag{12}$$

On the all-wrong route, $s_{g,t} = 0$ until recovery. All groups enter with the same Beta$(1, 1)$ prior, so their hazard estimates differ only with the number of observed fresh failures; their predicted token costs can also differ.

Each additional failure increases the posterior denominator without increasing its success count. For a fixed entry prior, the resulting hazard on the unrecovered route is $\widehat{h}_g(t) = a_g/(a_g + b_g + t)$ and decreases with t. Repeated failures therefore reduce the case for another draw even before the fresh-response cap is reached. A verified success ends recovery immediately; the controller does not generate further responses merely to increase the number of positive samples in that group.

4.2 Token cost and marginal recovery index

Let $\widehat{c}_g(t) > 0$ predict the non-padding tokens of the next response using only $\mathcal{H}_{g,t}$. If recovering group g has utility U_g, its one-step expected value and value per token are

$$\widehat{V}_g(t) = U_g \widehat{h}_g(t), \qquad I_g(t) = \frac{\widehat{V}_g(t)}{\widehat{c}_g(t)}. \tag{13}$$

The reported experiments use $U_g = 1$ to target recovered groups. Setting $\widehat{c}_g(t) = 1$ yields the call-aware ablation; the full method remains token-aware because reasoning lengths vary substantially.

The denominator makes the allocation sensitive to response length. With equal hazards and utilities, a group with twice the predicted token cost receives half the index. Conversely, equal costs reduce the ranking to the hazard ranking. Thus hazard-only and cost-only isolate different parts of the decision: the probability of repairing a group and the compute required for that repair. Generated lengths are charged after every draw, including unsuccessful draws, so the cost model affects allocation but never substitutes for measured accounting.

4.3 Shared-price allocation and stopping

A Lagrange multiplier $\lambda \geq 0$ represents the shadow price of the shared policy-token budget in Equation 7. At each allocation round, HAVER selects the active group with the largest marginal index and continues only if that index exceeds the price:

$$g^* = \arg\max_{g \in \mathcal{A}} I_g(t_g), \qquad \text{continue } g^* \iff I_{g^*}(t_g) > \lambda, \quad t_g < C_{\max}. \tag{14}$$

The development procedure chooses λ to meet the target budget; evaluation reuses it unchanged. A group exits the active set after its first verified success, when its cap is reached, or when its index falls below λ. This shared price couples otherwise independent stopping decisions and reallocates saved tokens toward groups with higher predicted recovery per token.

The rule follows by requiring positive estimated net value, $U_g \widehat{h}_g(t) - \lambda \widehat{c}_g(t) > 0$. Hazard or cost error can change the ranking; Appendix B bounds that error. The price stops low-index draws, while $C_{\max}$ limits draws even if an index stays high.

4.4 Verifier-grounded reconstruction and GRPO update

Every fresh response is scored by the same exact verifier as the base run. On success, the response replaces a predeclared failed member, producing

$$Y'_g = (Y_g \setminus \{o_{g,m}\}) \cup \{o_g^+\}, \qquad |Y'_g| = K, \tag{15}$$

where the replacement index m follows a fixed seeded rule independent of response content. The optimizer then applies the configured GRPO, Dr. GRPO, or RLOO estimator to Y'_g. If no attempt succeeds, $Y'_g = Y_g$ and the group retains zero reward advantage. HAVER therefore changes rollout allocation without introducing pseudo-rewards or modifying group normalization.

Restoring reward contrast. An initially all-wrong group repaired by one positive response has reward mean $1/K$. Under the reward centering used in our experiments, the retained positive response and the $K - 1$ retained failures have advantages

$$\widetilde{A}_g^+ = 1 - \frac{1}{K}, \qquad \widetilde{A}_g^- = -\frac{1}{K}, \qquad \frac{1}{K}\sum_{j=1}^{K}(R_{g,j} - \bar{R}_g)^2 = \frac{K-1}{K^2}. \tag{16}$$

For $K = 8$, these centered advantages are $7/8$ and $-1/8$, and the reward variance is $7/64$. Because the experiment disables group-standard-deviation normalization, these are also the configured group-relative advantages. Recovery restores a contrast through the exact verifier without changing the batch cardinality. This identifies the learning signal made available by reconstruction; downstream accuracy still depends on the policy update, which is evaluated separately in Section 5.

At each step, the controller updates the active groups' indices, samples the highest-index eligible group, and charges the generated tokens before observing the verifier result. It stops that group at first success, the price threshold, or the cap. Algorithm 1 summarizes one update; Appendix C specifies budget reservation and token accounting.

Algorithm 1 HAVER-GRPO for one policy update

Require: prompts $\mathcal{B}$, policy π_{old}, verifier v, group size K
Require: fresh-token budget $T_{\max}$, cap $C_{\max}$, price λ
Generate and verify K responses for each $q_g \in \mathcal{B}$; charge all non-padding tokens
Set $Y'_g \leftarrow Y_g$ and $\mathcal{A} \leftarrow \{g : \sum_{o \in Y_g} v(q_g, o) = 0\}$; set $t_g \leftarrow 0$
while $\mathcal{A} \neq \varnothing$ and the remaining fresh-token budget permits a rollout **do**
    Update $\widehat{h}_g(t_g)$, $\widehat{c}_g(t_g)$, and $I_g(t_g)$ for $g \in \mathcal{A}$
    $g^* \leftarrow \arg\max_{g \in \mathcal{A}} I_g(t_g)$
    if $I_{g^*}(t_{g^*}) \leq \lambda$ or $t_{g^*} = C_{\max}$ **then**
        Remove g^* from $\mathcal{A}$; **continue**
    end if
    Sample one fresh response for q_{g^*}; charge its non-padding tokens and set $t_{g^*} \leftarrow t_{g^*} + 1$
    if $v(q_{g^*}, o) = 1$ **then**
        Replace one predeclared failed slot in Y'_{g^*} with o; remove g^* from $\mathcal{A}$
    else if $t_{g^*} = C_{\max}$ **then**
        Remove g^* from $\mathcal{A}$
    end if
end while
Compute the configured group-relative advantages on all fixed-size Y'_g and update π_θ

5 Experiments

We evaluate reasoning accuracy, training efficiency, and the contribution of hazard and token-cost estimation under a common training horizon.

5.1 Setup

Models, data, and training. We use Qwen2.5-Math-1.5B-Instruct and Qwen2.5-Math-7B-Instruct (Yang et al., 2024). The formal matrix fixes $B = 64$, $K = 8$, and 300 optimizer updates. Training uses LoRA rank 16 and scaling 32 (Hu et al., 2021), bf16, learning rate 5×10^{-6}, and seeds 42, 43, and 44. Reported results are averaged across these runs. We evaluate AIME24, AIME25, AMC23, MATH-500, and OlympiadBench with pass@1(avg@8) (Hendrycks et al., 2021; He et al., 2024). Training and evaluation prompts are disjoint; controller parameters are selected on development data. Full decoding settings and implementation details are in Appendix C.

Baselines and accounting. We compare the pretrained Base, Uniform GRPO, DAPO's evaluate-then-filter dynamic sampling (Yu et al., 2025), Fixed-cap recovery, and HAVER-GRPO. Fixed-cap generates all $C_{\max}$ fresh responses for each routed group, even after a success; only the first success can replace an initial slot. All trained methods share the checkpoint, verifier, prompt order, group size, and optimizer recipe. Every generated response, including discarded attempts, is charged to the policy-token total. We report accuracy, total rollouts, generated tokens, and tokens-to-target at the same 300-update horizon.

Accuracy targets and aggregation. The five-suite mean weights benchmarks equally. For seed s, let $a_s(u)$ be this mean at checkpoint u and $T_s(u)$ its cumulative generated policy tokens. We locate a

common target $a^\star$ on the three-seed mean curve:

$$\bar{a}(u) = \frac{1}{3}\sum_s a_s(u), \quad u^\star = \min\{u : \bar{a}(u) \geq a^\star\}, \quad \overline{T}_{\text{target}} = \frac{1}{3}\sum_s T_s(u^\star), \tag{17}$$

The targets are 44.16% at 1.5B and 50.50% at 7B (Table 2). Total tokens measure the cost of completing all 300 updates; tokens-to-target measure the cost of reaching the same accuracy. The distinction matters because a method can spend more tokens over the full run while reaching the target earlier.

5.2 Main Results

Reasoning accuracy. Table 1 and Figure 2 summarize final accuracy and training progress. HAVER-GRPO achieves five-suite mean accuracies of 49.7% and 56.5% at 1.5B and 7B, respectively. On MATH-500, it reaches 80.5% and 89.4%, compared with Uniform GRPO's 76.8% and 85.8%. The training curves show how this accuracy gap develops over the common 300-update horizon.

Table 1: **Benchmark accuracy after 300 updates.** The comparison uses $B = 64$, $K = 8$, and 300 updates. Entries report pass@1(avg@8) accuracy in percent across five mathematical reasoning benchmarks. Trained-method entries average three seeds; Base denotes the checkpoint before training. Fixed-cap values are underlined; bold marks the best value within each model.

Model	Method	AIME24 (%)	AIME25 (%)	AMC23 (%)	MATH-500 (%)	OlympiadBench (%)
1.5B	Base	12.5	8.3	56.6	73.5	40.1
	Uniform GRPO	21.3	13.8	62.8	76.8	46.1
	DAPO	26.3	18.3	67.5	79.3	50.1
	Fixed-cap	26.7	17.9	66.9	79.6	49.6
	HAVER-GRPO (Ours)	**27.9**	**19.6**	**69.1**	**80.5**	**51.3**
7B	Base	19.2	15.0	57.8	81.3	43.6
	Uniform GRPO	28.3	22.5	66.3	85.8	49.6
	DAPO	33.8	27.5	71.3	88.3	53.6
	Fixed-cap	34.2	26.7	70.6	88.6	53.0
	HAVER-GRPO (Ours)	**35.4**	**29.2**	**73.1**	**89.4**	**55.3**

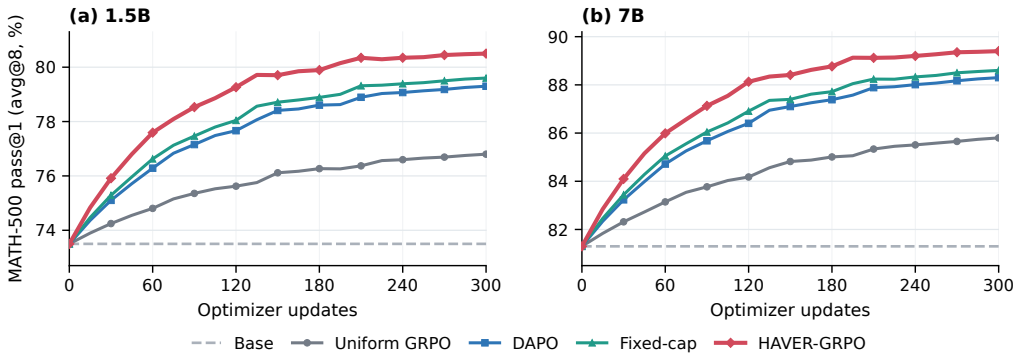


Figure 2: **MATH-500 accuracy over training.** Pass@1(avg@8) versus optimizer update for 1.5B and 7B under $B = 64$, $K = 8$, averaged across three seeds. Dashed lines show the Base accuracy; the final values correspond to Table 1.

The gains extend across all five benchmarks. Relative to Uniform GRPO, the five-suite mean increases by 5.52 and 5.98 points at 1.5B and 7B; the gains over DAPO are 1.38 and 1.58 points. On AIME24/AIME25, HAVER-GRPO exceeds DAPO by $1.6/1.3$ points at 1.5B and $1.6/1.7$ points at 7B.

Training efficiency. Table 2 reports compute costs and Figure 3 shows their evolution during training. HAVER-GRPO reduces tokens-to-target by 34.7% and 21.4% relative to Uniform GRPO at 1.5B and 7B. Compared with DAPO, it uses 48.0% and 57.8% fewer generated tokens while achieving higher mean accuracy.

Table 2: **Training efficiency over 300 updates.** $B = 64$, $K = 8$; entries average seeds 42, 43, and 44. Avg. pass@1 is the equal-weight mean of the five benchmark accuracies in Table 1. Rollout counts and policy-token totals include every generated response. The accuracy targets for Tok.-to-target are the Uniform-GRPO five-suite means, 44.16% at 1.5B and 50.50% at 7B. Token costs are reported in millions. Fixed-cap values are underlined; bold highlights the best accuracy and tokens-to-target within each model.

Model	Method	Avg. pass@1 ↑	Roll. (K) ↓	Tok. (M) ↓	Tok.-to-target (M) ↓
1.5B	Uniform GRPO	44.2	153.6	131.914	131.914
	DAPO	48.3	376.1	319.326	188.434
	Fixed-cap	48.1	219.4	191.257	114.677
	HAVER-GRPO (Ours)	**49.7**	191.8	165.921	**86.140**
7B	Uniform GRPO	50.5	153.6	275.154	275.154
	DAPO	54.9	515.3	943.791	583.315
	Fixed-cap	54.6	246.7	469.211	297.223
	HAVER-GRPO (Ours)	**56.5**	211.1	397.919	**216.274**

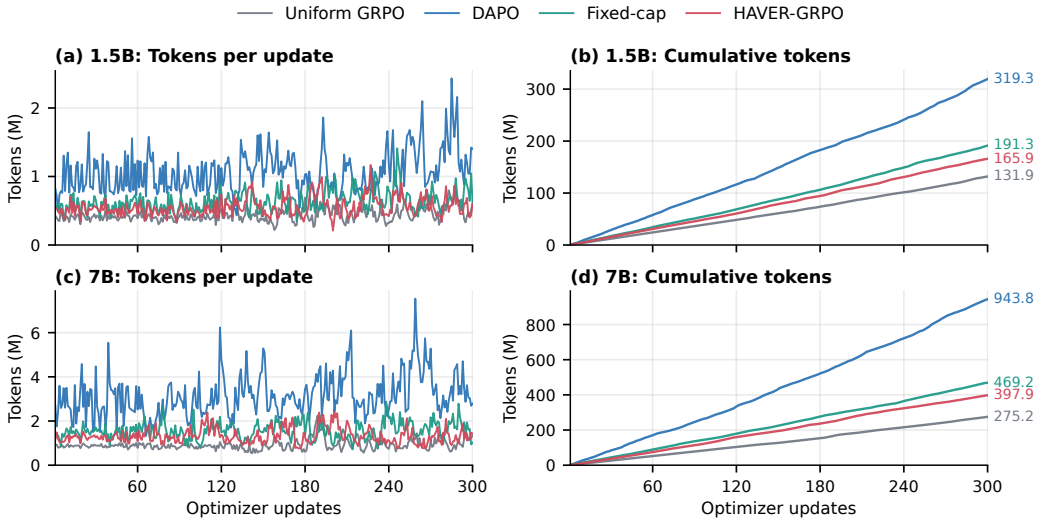


Figure 3: **Generated policy tokens across 300 updates.** Panels (a,c) show generated response tokens per update; panels (b,d) show their running totals for 1.5B and 7B, respectively. Curves average three seeds.

The total-token comparison and the target comparison answer complementary questions. HAVER-GRPO uses more generated tokens than Uniform GRPO over the full run, because it performs additional recovery sampling, but improves final accuracy and reaches the common targets with fewer tokens. Against Fixed-cap, it reduces total generated tokens by 13.2% at 1.5B and 15.2% at 7B, while improving five-suite mean accuracy by 1.54 and 1.86 points. These results show that allocating fresh draws selectively improves the accuracy–compute trade-off within the common training horizon.

5.3 Mechanism and Ablations

Table 3 separates the hazard and cost components. Hazard-only uses the failure-history posterior with unit call cost; Cost-only uses predicted token cost with a development-frozen mean hazard. Combining both components yields the highest accuracy on all five benchmarks at both model scales. Appendix C.7 reports recovery counts and extra-token costs to distinguish recovery behavior from downstream accuracy.

Table 3: **Controller ablations.** Fixed-cap uses a uniform limit; Hazard-only uses the failure-history posterior with unit call cost; Cost-only adds token-cost estimation to a development-frozen mean hazard; HAVER-GRPO uses group-specific hazard and cost estimates. Entries are percentages averaged across seeds 42, 43, and 44 under the same $B = 64$, $K = 8$, 300-update setting as Table 1. Fixed-cap values are underlined; bold marks the best value within each model.

Model	Method	AIME24 (%)	AIME25 (%)	AMC23 (%)	MATH-500 (%)	OlympiadBench (%)
1.5B	Fixed-cap	26.7	17.9	66.9	79.6	49.6
	Hazard-only	27.5	18.8	68.1	80.1	50.5
	Cost-only	27.1	18.3	67.5	79.9	50.1
	HAVER-GRPO (Ours)	**27.9**	**19.6**	**69.1**	**80.5**	**51.3**
7B	Fixed-cap	34.2	26.7	70.6	88.6	53.0
	Hazard-only	35.0	28.3	71.9	89.0	54.3
	Cost-only	34.6	27.5	71.3	88.8	53.7
	HAVER-GRPO (Ours)	**35.4**	**29.2**	**73.1**	**89.4**	**55.3**

6 Limitations and Conclusion

HAVER-GRPO is a narrow allocation mechanism. Its hazard posterior can be misspecified, response outcomes need not be independent, and a recovery-efficient sampler may fail to improve the final policy if recovered responses do not produce useful optimization directions. The method also depends on an exact verifier and is currently motivated by mathematical reasoning. These limitations make matched-token downstream evaluation essential.

We presented a verifier-grounded, token-aware sequential controller for all-wrong GRPO groups. The method keeps the policy objective and training group size fixed while deciding whether another on-policy response is worth its measured cost. In the 300-update experiments, HAVER-GRPO improves five-suite mean accuracy over DAPO by 1.4 and 1.6 percentage points at 1.5B and 7B, while reducing generated policy tokens by 48.0% and 57.8%, respectively.

Ethics statement

This work uses mathematical reasoning benchmarks and model-generated responses evaluated by an exact verifier. It does not involve human participants, personally identifiable information, clinical data, or human-subject experiments. The main ethical risks concern misuse of automated reasoning systems and overinterpretation of benchmark results. We state the limitations of the experiments and do not claim state-of-the-art performance.

Reproducibility statement

The paper specifies the model settings, verifier interface, fixed group size K, fresh-rollout cap, decoding configuration, optimizer configuration, 300-update comparison, and token-ledger accounting. All initial responses, successful fresh responses, and discarded fresh failures are charged to the policy-token budget. The experiments use $B = 64$, $K = 8$, and 300 optimizer updates. Table 1 reports benchmark accuracy, and Table 2 reports training efficiency.

LLM Usage

Large language models were used as general-purpose writing tools to assist with drafting and polishing the manuscript. Their assistance included refining the language, improving readability, and enhancing clarity through sentence rephrasing, grammar checking, and formatting.

The authors retained control over the research ideas, methodology, and scientific interpretation. AI-assisted material was reviewed and revised by the authors to ensure consistency with the intended content of the paper.

The authors take full responsibility for the manuscript, including any text drafted or polished with LLM assistance. They reviewed the material for accuracy, originality, and adherence to ethical standards.

REFERENCES

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025. doi: 10.48550/arXiv.2503.04697. URL https://arxiv.org/abs/2503.04697.

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. doi: 10.48550/arXiv.2402.14740. URL https://arxiv.org/abs/2402.14740.

Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, JeongYeon Nam, and Donghyun Kwak. Online difficulty filtering for reasoning oriented reinforcement learning. *arXiv preprint arXiv:2504.03380*, 2025. doi: 10.48550/arXiv.2504.03380. URL https://arxiv.org/abs/2504.03380.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. doi: 10.48550/arXiv.2107.03374. URL https://arxiv.org/abs/2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. doi: 10.48550/arXiv.2110.14168. URL https://arxiv.org/abs/2110.14168.

David R. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–220, 1972. doi: 10.1111/j.2517-6161.1972.tb00899.x.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. doi: 10.48550/arXiv.2501.12948. URL https://arxiv.org/abs/2501.12948.

John C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(2):148–164, 1979. doi: 10.1111/j.2517-6161.1979.tb01068.x.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, et al. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024. doi: 10.48550/arXiv.2402.14008. URL https://arxiv.org/abs/2402.14008.

Xixiang He, Qiyao Sun, Ao Cheng, Xingming Li, Xuanyu Ji, Hailun Lu, Runke Huang, and Qingyong Hu. Advantage collapse in group relative policy optimization: Diagnosis and mitigation. *arXiv preprint arXiv:2605.21125*, 2026. doi: 10.48550/arXiv.2605.21125. URL https://arxiv.org/abs/2605.21125.

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. doi: 10.48550/arXiv.2103.03874. URL https://arxiv.org/abs/2103.03874.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. doi: 10.48550/arXiv.2106.09685. URL https://arxiv.org/abs/2106.09685.

Heyang Jiang, Henry Liu, and Baharan Mirzasoleiman. Learning as reasoning unfolds: Progressive rollout allocation for efficient reinforcement learning. *arXiv preprint arXiv:2607.22002*, 2026. doi: 10.48550/arXiv.2607.22002. URL https://arxiv.org/abs/2607.22002.

Zhihang Lin, Mingbao Lin, Yuan Xie, and Rongrong Ji. CPPO: Accelerating the training of group relative policy optimization-based reasoning models. *arXiv preprint arXiv:2503.22342*, 2025. doi: 10.48550/arXiv.2503.22342. URL https://arxiv.org/abs/2503.22342.

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*, 2025. doi: 10.48550/arXiv.2503.20783. URL https://arxiv.org/abs/2503.20783.

Hieu Trung Nguyen, Bao Nguyen, Wenao Ma, Yuzhi Zhao, Ruifeng She, and Viet Anh Nguyen. Adaptive rollout allocation for online reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2602.01601*, 2026. doi: 10.48550/arXiv.2602.01601. URL https://arxiv.org/abs/2602.01601.

Pixel Nomand, Elena Voss, Marcus Hale, and Sofia Reyes. Early verdicts, better budgets: Sequential adaptive rollout allocation for compute-efficient rlvr. *arXiv preprint arXiv:2607.26253*, 2026. doi: 10.48550/arXiv.2607.26253. URL https://arxiv.org/abs/2607.26253.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015a. doi: 10.48550/arXiv.1502.05477. URL https://arxiv.org/abs/1502.05477.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b. doi: 10.48550/arXiv.1506.02438. URL https://arxiv.org/abs/1506.02438.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. doi: 10.48550/arXiv.1707.06347. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. doi: 10.48550/arXiv.2402.03300. URL https://arxiv.org/abs/2402.03300.

Abraham Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16 (2):117–186, 1945. doi: 10.1214/aoms/1177731118.

Tao Wang, Shuo Li, Yan Sun, Dongsheng Ding, and Edgar Dobriban. Where to spend rollouts: Hit-utility optimal rollout allocation for group-based RLVR. *arXiv preprint arXiv:2605.07114*, 2026. doi: 10.48550/arXiv.2605.07114. URL https://arxiv.org/abs/2605.07114.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. doi: 10.1007/BF00992696.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, et al. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024. doi: 10.48550/arXiv.2409.12122. URL https://arxiv.org/abs/2409.12122.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025. doi: 10.48550/arXiv.2503.14476. URL https://arxiv.org/abs/2503.14476.

Haizhong Zheng, Yang Zhou, Brian R. Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for LLM reasoning via selective rollouts. *arXiv preprint arXiv:2506.02177*, 2025. doi: 10.48550/arXiv.2506.02177. URL https://arxiv.org/abs/2506.02177.

A Notation and Problem Setup

This appendix fixes the notation used by the controller and states the scope of the claims. A prompt group is created by the behavior policy, and all fresh responses are sampled from the same frozen behavior policy until the next policy update. The verifier is treated as an external exact function; it is not trained by HAVER.

A.1 NOTATION

Table 4: Notation used in the method and the allocation protocol.

Symbol	Meaning
q_g	Prompt for group g
$Y_g = \{o_{g,1}, \ldots, o_{g,K}\}$	Initial response group of fixed size K
$R_{g,j} = v(q_g, o_{g,j})$	Exact verifier outcome for an initial response
H_g	All-wrong event, $\sum_j R_{g,j} = 0$
$Z_{g,t}$	Outcome of fresh attempt t after H_g
$c_{g,t}$	Non-padding response-token cost of fresh attempt t
$\mathcal{H}_{g,t}$	History available after t fresh attempts
$h_g(t), \widehat{h}_g(t)$	True and estimated next-response success hazard
$c_g(t)$	Conditional expected next-response token cost
$\widehat{c}_g(t)$	Estimated non-padding cost of the next response
$U_g \geq 0$	Utility of recovering group g; default $U_g = 1$
$I_g(t)$	Estimated recovery value per token
λ	Shared policy-token shadow price
$C_{\max}$	Maximum number of fresh attempts per all-wrong group
Y'_g	Fixed-K group passed to the optimizer
T_{policy}	Tokens from all initial and fresh policy responses

A.2 ASSUMPTIONS AND SCOPE

The formal controller assumes binary verifier outcomes, a finite fresh-rollout cap, and a frozen behavior policy within one optimizer update. The Beta–Bernoulli posterior is a tractable model for the next-success hazard; it is not an assumption that all prompts have the same success probability. Conditional independence with a constant success probability is used only for the geometric special case in Appendix B; the survival identities themselves follow from conditioning and do not require independent attempts. The implementation must use only the initial group, observed fresh outcomes, observed token lengths, and development-frozen controller parameters. Future verifier outcomes, final evaluation labels, and hidden repair labels are not available at decision time.

B THEORETICAL ANALYSIS

B.1 HOMOGENEOUS GROUPS HAVE NO REWARD-DEPENDENT ADVANTAGE

If H_g holds, then $R_{g,j} = 0$ for every j and $\bar{R}_g = 0$. Therefore Equation 4 gives $\widetilde{A}^{\text{GRPO}}_{g,j} = 0$ for all j, including when the stabilizing constant δ is positive. The Dr. GRPO estimator in Equation 5 is also zero. For RLOO, $R_{g,j} - (K-1)^{-1} \sum_{k \neq j} R_{g,k} = 0$. Hence the reward-dependent part of the group gradient is zero for all three estimators. The same argument applies to an all-correct group because all rewards are identical. HAVER targets only the all-wrong case because an exact positive response can then replace a failed slot and create a nonzero contrast without changing the optimizer interface. This statement concerns the reward-dependent surrogate; a separately enabled KL penalty or other regularizer can still contribute a gradient.

B.2 RECOVERY UNDER A CHANGING HAZARD

Let $S_g(t)$ be the probability that group g survives t fresh attempts without a verifier-positive response. To obtain scalar probabilities when the history-dependent hazard varies across surviving trajectories, define the survival-conditioned hazard

$$\bar{h}_g(t) = P(Z_{g,t+1} = 1 \mid Z_{g,1:t} = 0, \mathcal{H}_{g,0}, H_g),$$

which averages over histories induced by the fixed-cap, first-success policy. With $S_g(0) = 1$, conditioning gives

$$S_g(t+1) = S_g(t)\big(1 - \bar{h}_g(t)\big), \qquad S_g(t) = \prod_{u=0}^{t-1} \big(1 - \bar{h}_g(u)\big). \tag{18}$$

For a fixed cap C, the recovery probability is therefore

$$P(\text{recover} \mid \mathcal{H}_{g,0}, C) = 1 - S_g(C) = 1 - \prod_{u=0}^{C-1} \big(1 - \bar{h}_g(u)\big). \tag{19}$$

When $h_g(u) = p_g$ for every u, this reduces to Equation 9. The expected number of fresh responses under the cap is

$$\mathbb{E}[N_g(C)] = \sum_{t=0}^{C-1} S_g(t). \tag{20}$$

If $\bar{c}_g(t)$ is the conditional expected token cost of attempt $t+1$ given survival through t attempts, the expected fresh policy-token cost is

$$\mathbb{E}[T_g(C)] = \sum_{t=0}^{C-1} S_g(t)\, \bar{c}_g(t). \tag{21}$$

These equations are for a fixed cap with sampling until the first success. For a history-dependent controller that may stop before the cap, let $A_{g,t}$ indicate whether fresh attempt $t+1$ is admitted. Its expected token cost is instead

$$\mathbb{E}[T_g] = \sum_{t \geq 0} \mathbb{E}[A_{g,t}\, c_{g,t+1}]. \tag{22}$$

This form accounts for the controller's history-dependent admission decisions. For the fixed-cap baseline that deliberately generates all C draws, the draw count is C regardless of an earlier success, so its cost must not be reported using the first-success stopping formula in Equation 20.

Beta posterior predictive special case. Under the exchangeable Beta–Bernoulli model, take $a_g, b_g > 0$ to be the parameters *at entry to the all-wrong route*. Integrating over the latent success probability gives

$$S_g(t) = \frac{\mathrm{B}(a_g, b_g + t)}{\mathrm{B}(a_g, b_g)}, \qquad \bar{h}_g(t) = \frac{a_g}{a_g + b_g + t}, \tag{23}$$

where B is the beta function. Therefore recovery by C has probability $1 - \mathrm{B}(a_g, b_g + C)/\mathrm{B}(a_g, b_g)$, and the expected first-success draw count follows by substituting this $S_g(t)$ into Equation 20. This is a predictive identity for the assumed model, not a calibration guarantee for an LLM's actual response distribution.

B.3 Derivation of the marginal recovery index

Suppose one more response has conditional success probability $h_g(t)$, utility $U_g \geq 0$ when it recovers the group, and conditional expected token cost $c_g(t) > 0$. Under a shared token price λ, the one-step expected net value is

$$\Delta_g(t) = U_g h_g(t) - \lambda c_g(t). \tag{24}$$

The myopic decision continues when $\Delta_g(t) > 0$. Since $c_g(t) > 0$, this is equivalent to

$$\frac{U_g h_g(t)}{c_g(t)} > \lambda. \tag{25}$$

Replacing the unknown quantities by their development-frozen estimates yields Equation 14. The index is a marginal stopping rule; it is not a claim that the greedy policy is globally optimal for arbitrary correlated rollouts. The shared price is selected on development data to map the local decision rule to a global policy-token budget and is then held fixed during evaluation.

B.4 Sensitivity to hazard and cost estimation error

Let $I_g = U_g h_g / c_g$ and $\widehat{I}_g = U_g \widehat{h}_g / \widehat{c}_g$. If $|\widehat{h}_g - h_g| \leq \varepsilon_h$ and $|\widehat{c}_g - c_g| \leq \varepsilon_c < c_g$, then

$$|\widehat{I}_g - I_g| \leq U_g \frac{\varepsilon_h}{c_g - \varepsilon_c} + U_g \frac{h_g \varepsilon_c}{c_g(c_g - \varepsilon_c)}. \tag{26}$$

To see this, write

$$U_g \left| \frac{\widehat{h}_g}{\widehat{c}_g} - \frac{h_g}{c_g} \right| \le U_g \frac{|\widehat{h}_g - h_g|}{\widehat{c}_g} + U_g \frac{h_g |c_g - \widehat{c}_g|}{\widehat{c}_g c_g},$$

then use $\widehat{c}_g \ge c_g - \varepsilon_c$. Thus a ranking between groups g and g' is unchanged whenever their true index gap exceeds the sum of the two right-hand sides. This condition characterizes when hazard and token-cost errors can change the controller's ranking.

B.5 Fixed-K reconstruction invariant

The group is an indexed list of K response records (slots), not a mathematical set of unique text strings; duplicate response texts remain distinct records. If the first fresh verifier- positive response is o_g^+, HAVER removes one predeclared failed response and forms

$$Y_g' = (Y_g \setminus \{o_{g,m}\}) \cup \{o_g^+\}, \qquad |Y_g'| = K. \tag{27}$$

If the controller stops without success, it sets $Y_g' = Y_g$, which also has K members. All fresh failures are charged to T_{policy} but are never inserted into Y_g'. Therefore rollout allocation can vary while the optimizer always receives the same group cardinality and the configured GRPO advantage formula remains unchanged (the resulting advantage values can change).

C Implementation and Experimental Setup

This section records the experimental configuration and the implementation of the hazard-and-cost controller in Equation 14.

C.1 Experimental configuration

Table 5: Experimental settings for the 300-update comparison.

Setting	Value
Policy models	Qwen2.5-Math-1.5B-Instruct; Qwen2.5-Math-7B-Instruct
Prompt batch and training horizon	$B = 64$; 300 updates
Initial group size	$K = 8$
Fresh-response cap	$C_{\max} = 8$
Hazard prior initialization	Beta$(1, 1)$
LoRA	rank 16, alpha 32
Learning rate and dtype	5×10^{-6}, bf16
GRPO and KL settings	no group-std advantage normalization; KL coefficient 0.001
Optimizer minibatch	8
Training seeds	42, 43, 44; results averaged
Training rollout decoding	temperature 1.0, top-$p = 1.0$
Response length limit	4096 tokens

The hazard posterior follows Equation 12. If a prior is fitted before screening and initial outcomes are exchangeable with fresh attempts, conditioning it on H_g adds K failures once; a prior fitted directly on routed groups must not add them again. The controller stops after the first verified success, when the fresh-response cap is reached, or when the token-normalized index falls below the shared price λ.

C.2 Controller and batch scheduling

At the beginning of an update, the collector generates eight responses for every prompt and applies the exact verifier. Only groups with no positive response are placed in the active set. The collector updates the group state after the round, records the response length and verifier outcome, and removes a group after success or after its stopping condition. This keeps GPU calls batched while preserving the observed-outcome-before-next-round rule. The controller uses the index in Equation 14 to select the highest-index eligible group for each fresh response.

Algorithm 2 Budget-safe controller for routed all-wrong groups

Require: initial groups Y_g, frozen policy π_{old}, verifier v, frozen estimators and utilities U_g
Require: local fresh-token allowance B_f, cap $C_{\max}$, price λ, response length limit $L_{\max}$
$Y'_g \leftarrow Y_g$; $t_g \leftarrow 0$; initialize $\mathcal{H}_{g,0}$ and entry priors; $\mathcal{A} \leftarrow \{g : H_g\}$; $T_f \leftarrow 0$
while $\mathcal{A} \neq \varnothing$ and $B_f - T_f \geq L_{\max}$ **do**
  Remove all $g \in \mathcal{A}$ with $t_g = C_{\max}$
  if $\mathcal{A} = \varnothing$ **then**
    break
  end if
  Estimate $\widehat{h}_g(t_g)$ and $\widehat{c}_g(t_g) > 0$; set $I_g = U_g \widehat{h}_g / \widehat{c}_g$ for each $g \in \mathcal{A}$
  Remove all g with $I_g \leq \lambda$ from $\mathcal{A}$
  if $\mathcal{A} = \varnothing$ **then**
    break
  end if
  Choose $g^* = \arg\max_{g \in \mathcal{A}} I_g$ using a frozen tie order
  Generate one fresh $o \sim \pi_{\text{old}}(\cdot \mid q_{g^*})$ with limit $L_{\max}$
  $T_f \leftarrow T_f + \text{len}_{\text{nonpad}}(o)$; $t_{g^*} \leftarrow t_{g^*} + 1$
  $z \leftarrow v(q_{g^*}, o)$; record the draw, tokens, outcome, and decision
  if $z = 1$ **then**
    Replace one predeclared failed slot of Y'_{g^*} by o; remove g^* from $\mathcal{A}$
  else
    Update $\mathcal{H}_{g^*, t_{g^*}}$ with the failed draw and its length
  end if
end while
return all Y'_g with $|Y'_g| = K$, and the complete fresh-token ledger

The collector replaces a predeclared original failed slot. Only the first verifier-positive fresh response is retained. In the fixed-cap baseline, later responses, including positives, are charged but not added to the optimizer batch. A runtime assertion checks that the concatenated optimizer batch contains exactly 8 responses per prompt.

Budget-safe sequential procedure. Algorithm 2 specifies one fresh response at a time; $L_{\max}$ is the maximum generated response length. Reserving $L_{\max}$ before admission ensures that a fresh draw cannot exceed the *local* fresh-token allowance. This is distinct from the global fixed-budget comparison, which stops at the first completed update and reports the realized overshoot.

C.3 Hazard prior, cost model, and parameter freezing

The controller uses the posterior in Equation 12, a positive estimate of the next response's non-padding token length, and a shared token price λ. Controller parameters are selected on development data and frozen before evaluation.

C.4 Token accounting and matched-compute protocol

For every generated response r, the ledger records $\text{len}_{\text{nonpad}}(y_r)$. The policy-token total is

$$T_{\text{policy}} = \sum_{r \in \mathcal{R}} \text{len}_{\text{nonpad}}(y_r), \tag{28}$$

where $\mathcal{R}$ includes initial responses, verifier-positive fresh responses, and discarded fresh failures. The comparison holds optimizer updates constant at 300 and reports each arm's generated policy-token total. Prompt tokens, padding tokens, and verifier runtime are outside this token metric.

The project records three JSONL streams: `allocation_ledger.jsonl` for stopping decisions and recovery, `generation_ledger.jsonl` for generated responses, and `compute_ledger.jsonl` for phase timing and token accounting. Allocation rows contain per-update totals and a nested group list with prompt identifiers, initial all-wrong flags, extra draws, extra tokens, recovery

flags, and initial optimizer-token counts. Per-decision records store the posterior, predicted token cost, allocation action, and verifier outcome.

The sum of retained optimizer tokens is distinct from generated tokens: their difference also includes displaced initial slots and any later fixed-cap draws. Thus it is not identical to discarded fresh-failure tokens. Verifier time, prompt processing, optimization, and generation timing remain separate from the policy-token budget.

C.5 Models, verifier, and baseline configurations

The training protocol uses LoRA rank 16, alpha 32, learning rate 5×10^{-6}, bf16, gradient checkpointing, group size eight, and seeds 42, 43, and 44 for the 300-update experiments. Evaluation uses AIME24, AIME25, AMC23, MATH-500, and OlympiadBench with eight samples per prompt, temperature 1.0, top-$p = 0.95$, and a 4096-token response limit. The exact verifier and answer normalization are shared across all arms; malformed or unverifiable outputs receive zero outcome reward.

Uniform GRPO samples the initial group and performs no fresh recovery. DAPO is the evaluate-then-filter dynamic-sampling control. Fixed-cap applies the same fresh-response limit to every routed all-wrong group. Cost-only uses the cost estimate with a development-frozen mean hazard; HAVER uses group-specific hazard and cost estimates under the shared price. All arms share the base checkpoint, prompt order, optimizer configuration, verifier, decoding limits, and evaluation procedure within a comparison.

Data and hardware. Training uses the project data source `lora_quarantine_v3`. The five-suite evaluation is configured under `lora_quarantine_v3/paper_eval_v1_full5`. Experiments use NVIDIA A100.

C.6 Evaluation metrics and statistical reporting

For suite d with n_d prompts, let $r_{d,i,j} \in \{0, 1\}$ be the exact verifier outcome of evaluation sample j. The primary quality metric is

$$\widehat{a}_d = \frac{1}{n_d} \sum_{i=1}^{n_d} \frac{1}{8} \sum_{j=1}^{8} r_{d,i,j}, \qquad \widehat{a}_{\text{macro}} = \frac{1}{5} \sum_{d=1}^{5} \widehat{a}_d. \tag{29}$$

The metric is computed separately for each trained seed and then averaged over seeds 42, 43, and 44. Tables show $100\widehat{a}_d$ in percent. The notation pass@1(avg@8) means averaging single-draw correctness across eight samples, not pass@8, majority vote, or best-of-eight selection. The five-suite average weights suites equally rather than pooling all prompts.

Recovery rate divides recovered groups by initially all-wrong groups; recovery yield divides recovered groups by policy tokens, multiplied by 10^6. When a yield uses *extra* tokens only, it is explicitly labeled as in Table 6; it is not the end-to-end policy-token yield. Tokens-to-target uses the Uniform-GRPO 300-update five-suite mean as the common accuracy target at each model scale. It is the mean cumulative token count at the first checkpoint where the three-seed mean accuracy reaches that target, as in Equation 17. If the mean curve never reaches the target, the entry is reported as not reached rather than extrapolated.

The reported benchmark accuracies and 300-update costs are arithmetic means over the three training runs; tokens-to-target follows Equation 17.

C.7 Recovery and extra-token accounting

Table 6 reports recovery counts and extra-token costs for the 300-update comparison. Total policy-token costs are reported in Table 2.

Table 6: **300-update recovery and extra-token comparison.** The setting is $B = 64$, $K = 8$, 300 updates, and seeds 42, 43, and 44; entries are averages across the three runs. Recovery is the number of initially all-wrong prompt groups repaired by a fresh response. Extra tokens are already included in the total policy tokens in Table 2: for DAPO they are generated tokens from filtered-out groups, whereas for Fixed-cap and HAVER-GRPO they are all fresh-response tokens on the recovery route. Bold marks the favorable value between the two same-prompt recovery methods within each model and metric. Fixed-cap values are underlined.

Model	Method	Recovered groups ↑	Recovery rate (%) ↑	Extra tok. (M) ↓	Extra tok./rec. (K) ↓
1.5B	Uniform GRPO	–	–	0.000	–
	DAPO	–	–	187.457	–
	Fixed-cap	**1,240**	**15.081**	59.663	**48.116**
	HAVER-GRPO (Ours)	653	8.214	**33.239**	50.902
7B	Uniform GRPO	–	–	0.000	–
	DAPO	–	–	721.022	–
	Fixed-cap	1,197	**10.283**	189.705	158.484
	HAVER-GRPO (Ours)	**1,203**	10.032	**116.678**	**96.989**

Uniform GRPO performs no extra sampling. DAPO resamples after filtering whole prompt groups, so same-prompt recovery and its rate are not applicable rather than zero. For Fixed-cap and HAVER-GRPO, every fresh attempt is included in the extra-token cost, including responses discarded before optimization.

Figure 4 compares the total generated-token costs in Table 2. Within each model scale, the blue segment is the Uniform GRPO total and the upper segment is the difference from that reference. These upper segments are arithmetic differences between the generated-token totals in Table 2, whereas Table 6 reports extra-token categories.

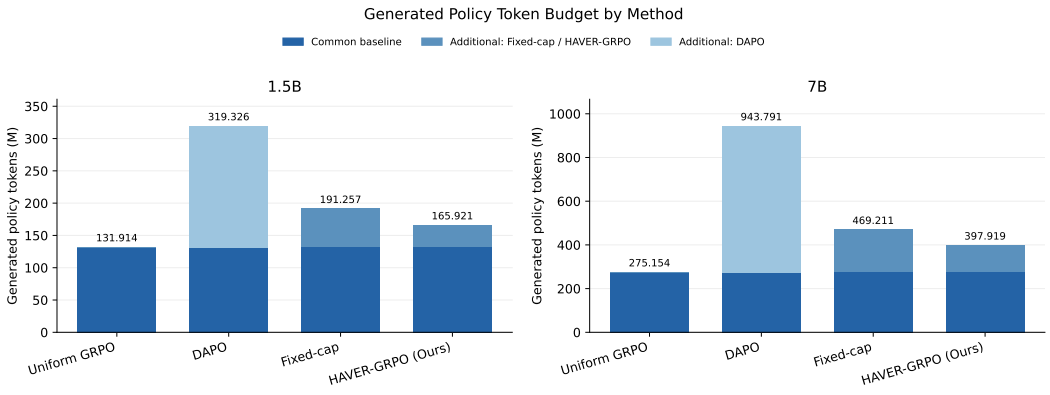


Figure 4: **Generated policy token budget by method.** Each bar totals the corresponding *Tok. (M)* value in Table 2. The common lower segment is the Uniform GRPO total for that model scale; the upper segment is the difference from that reference.

D Discussion and Disclosure

D.1 Limitations

The hazard model can be misspecified when responses are correlated, share a tree prefix, or change distribution rapidly during a policy update. The token cost model can also be biased for prompts with unusually long reasoning traces. The index is a local marginal rule and does not by itself guarantee optimal allocation under arbitrary dependence. Exact verifier access is required, and a recovered response can still fail to improve the downstream policy if its reward contrast is weak or its gradient conflicts with the training distribution. The three-seed comparison does not establish broader superiority or a SOTA claim.

D.2 Extensions

For continuous rewards, one can replace the binary success event with a pre-registered reward threshold or a posterior model for reward spread. For tree or multi-turn rollouts, the history can be attached to a prefix anchor and the same fixed-K reconstruction can be applied at the group boundary. Response length control is complementary: it changes the cost of an individual response, whereas HAVER decides whether another response should be generated.

D.3 Broader impact

By targeting wasted rollout tokens, the method may reduce the compute and energy required for verifier-based post-training. The same reduction can lower the cost of deploying stronger reasoning models, so the usual dual-use and capability risks of language-model post-training still apply. A token reduction is not by itself a measured energy reduction: energy claims would require device-level measurement. The described study concerns model training on mathematical benchmarks rather than a newly conducted participant study; data licensing and provenance still require review against the final manifests.