Hierarchical Reflective Policy Optimization for Long-Horizon Agentic Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

Agentic reinforcement learning (RL) has emerged as the core paradigm for empowering large language model (LLM) agents to accomplish open-ended, long-horizon interactive tasks. However, existing agentic RL methods suffer from two critical bottlenecks that severely limit real-world deployment: coarse trajectory-level credit assignment that fails to capture step-wise valid reasoning in failed rollouts, and static policy optimization that cannot adapt the exploration–exploitation trade-off to dynamic task complexity. To address these fundamental limitations, we propose Hierarchical Reflective Policy Optimization (Hrpo), a novel agentic RL framework equipped with dual-level reflective credit assignment and adaptive curriculum exploration for long-horizon agent decision-making. Specifically, we first construct a trajectory state-transition graph to model hierarchical agent–environment interaction dynamics, which decomposes holistic episode rewards into fine-grained step-level and sub-task-level credit signals, effectively excavating latent valuable reasoning steps obscured in incomplete or failed trajectories. Furthermore, we design a task-aware reflective exploration module that enables the agent to self-summarize historical interaction errors and successful experiences during rollouts, dynamically adjusting policy exploration intensity and avoiding redundant trial-and-error in repeated scenarios. Unlike conventional fixed-strategy RL and static agent refinement methods, Hrpo unifies experiential reflection, hierarchical credit assignment and adaptive policy update in a single end-to-end optimization pipeline, significantly improving the stability and efficiency of agent training. On a controlled long-horizon agent evaluation suite, Hrpo reaches an 80% task success rate using 88% fewer environment interactions than the group-relative baseline and attributes failures to the true faulty sub-task with 72.8% top-1 accuracy versus 12.4% for trajectory-level credit under fault injection. Ablations attribute these gains primarily to the hierarchical credit assignment; the reflective exploration component yields no measurable additional benefit in the small-scale suite—a negative result we report transparently and revisit under the large-benchmark protocol.

1 Introduction

Large language model (LLM) agents increasingly operate in open-ended environments—web browsers, operating systems, scientific simulators, and interactive codebases—where solving a task requires tens of sequential decisions interleaved with tool calls and environmental feedback (Yao et al., 2023; Zhou et al., 2024; Wang et al., 2022). Reinforcement learning (RL) has become the dominant post-training paradigm for such agentic capabilities (Yu et al., 2025a): group-based methods such as GRPO (Shao et al., 2024) discard the critic network of PPO (Schulman et al., 2017) and optimize group-relative advantages, powering frontier reasoning systems (Guo et al., 2025).

Transferring these gains to long-horizon agentic settings, however, exposes two structural bottlenecks.

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.

(i) Coarse trajectory-level credit assignment. Group-based methods assign a single trajectory-level advantage to every token of a rollout according to the final outcome. In long episodes with sparse terminal rewards, successful trajectories may contain redundant detours while failed trajectories may contain excellent initial steps that are penalized indiscriminately (Feng et al., 2025; Cheng et al., 2026). The resulting learning signal is noisy, and valuable intermediate reasoning steps in failed rollouts are simply discarded.

(ii) Static exploration. Mainstream agentic RL pipelines keep the exploration pressure (e.g., the entropy coefficient) fixed throughout training (Yu et al., 2025b). Yet task difficulty is heterogeneous and non-stationary: the same agent state may be reliably solved in one sub-task context and persistently failed in another. Static exploration either wastes rollouts re-sampling states the policy has already mastered, or under-explores states with a history of failures, causing redundant trial-and-error.

We propose Hierarchical Reflective Policy Optimization (Hrpo), a unified agentic RL framework that addresses both bottlenecks. Hrpo maintains a trajectory state-transition graph over the anchor states visited by the agent population, and reads two kinds of statistics from it: the expected stage-advance probability of each state and the empirical value of each state. From these, Hrpo derives a hierarchical advantage that blends a trajectory-level group-relative term with a step-level graph-derived term, so that steps that genuinely advance the task receive positive credit even when the episode ultimately fails. On top of this, a task-aware reflective exploration module summarizes historical failures per state into a reflection memory and converts it into an adaptive exploration coefficient, coupled with a success-gated difficulty curriculum. Our contributions are:

- A trajectory state-transition graph formulation of agentic rollouts and a dual-level (trajectory/sub-task) credit decomposition with a derivation showing the mixed advantage yields an unbiased policy-gradient estimator whose variance is strictly reduced when the step-level term is correlated with the true return advantage (§3.3).
- A task-aware reflective exploration module that adaptively modulates exploration from accumulated error statistics, unifying reflection, curriculum, and policy update in one pipeline (§3.4).
- Controlled experiments on a long-horizon agent task suite: Hrpo reaches the 80% success threshold with 88% fewer environment interactions than GRPO and 100.0% final success at the hardest difficulty; fault-injection credit-attribution experiments quantify that Hrpo localizes failures to the true sub-task with 72.8% accuracy (§4).
- Full ablations, hyper-parameter sensitivity analysis, derivations, and complete runnable code (§5 and Appendix B).

2 Related Work

Agentic RL for LLMs. A rapidly growing line of work trains LLM agents with online RL in interactive environments (Yu et al., 2025a). GRPO (Shao et al., 2024) popularized critic-free group-relative optimization; DAPO (Yu et al., 2025b) decouples clipping and dynamic sampling; RLOO-style baselines are analyzed by Wang et al. (2025). For multi-turn settings, GiGPO (Feng et al., 2025) introduces a two-level advantage with anchor-state grouping; ARPO (Dong et al., 2025) allocates rollouts by token-entropy spikes after tool calls; GraphGPO (Cheng et al., 2026) aggregates rollouts into a state-transition graph and assigns edge credit by goal-distance reduction. Hrpo shares the graph aggregation spirit of GraphGPO and the hierarchical view of GiGPO, but differs in three ways: (i) our step-level credit is derived from edge-level progress statistics, which remain informative inside failed trajectories even when goal-distance estimates are flat; (ii) the step-level and trajectory-level signals are blended with a single mixing weight with an explicit variance-reduction characterization; and (iii) exploration is reflective: the graph's error statistics feed back into the exploration coefficient, closing the loop between credit assignment and exploration.

Reflective agents. Verbal reflection methods such as Reflexion (Shinn et al., 2023) and Agent Q (Putta et al., 2024) improve agents through self-critique without gradient updates, while Search-R1 (Jin et al., 2025) and WebRL (Qi et al., 2025) use RL with curricula over search

and web tasks. Hrpo internalizes reflection into the RL objective: experience summaries act as state-conditioned exploration modulators rather than external prompts or data filters.

Curriculum for agent RL. Self-evolving curricula (Qi et al., 2025) schedule task difficulty by model performance. Hrpo adopts a success-gated length curriculum as one instantiation of its adaptive component and shows it is complementary to credit-level innovations (ablation, §5).

3 Method

3.1 Preliminaries

Agentic interaction as a POMDP. An LLM agent interacts with an environment through the loop (s_t, a_t, r_t): at step t the agent observes a state s_t (instruction, interaction history, tool feedback), emits an action a_t (text, tool call, or operation), and receives a reward. Following the standard agentic RL formulation (Yu et al., 2025a), the interaction is a partially observable Markov decision process with horizon T and sparse terminal reward $R = \sum_t \gamma^t r_t$, $r_t = 0$ for $t < T$ and $r_T \in \{0, 1\}$ for binary task success.

Group-relative policy optimization. GRPO (Shao et al., 2024) samples a group of G rollouts $\{\tau_i\}_{i=1}^G$ per task instance and defines the trajectory-level advantage

$$\hat{A}_i^{\text{traj}} \;=\; \frac{R_i - \text{mean}(\{R_j\}_{j=1}^G)}{\text{std}(\{R_j\}_{j=1}^G) + \epsilon}, \tag{1}$$

which is broadcast to every step of trajectory τ_i. This critic-free estimator is memory-efficient but step-blind: every step of a failed rollout receives the same negative signal.

3.2 Trajectory State-Transition Graph

Definition 1 (Anchor state). Given a decomposition of a task into K ordered sub-tasks (stages), the anchor state of step t is the pair $\kappa_t = (\text{stage}(t), \text{ctx}(t))$ of the current sub-task index and the task-context signature. Anchor states are intentionally coarser than raw observations: they define the nodes over which cross-trajectory statistics are meaningfully shared.

Over a batch of G rollouts we aggregate a directed state-transition graph $\mathcal{G} = (V, E)$: each anchor state is a node $v \in V$, and each decision step contributes an edge $e_t = (\kappa_t, \kappa_{t+1})$ labeled by the action, the stage-advance indicator $g_t \in \{0, 1\}$ (whether the transition advanced the sub-task), and the bootstrapped return $\hat{R}_t = \sum_{t' \geq t} \gamma^{t'-t} r_{t'}$. Two running statistics are maintained per node with learning rates η_1, η_2:

$$\hat{p}(v) \leftarrow (1 - \eta_1)\, \hat{p}(v) + \eta_1\, g_t, \tag{2}$$

$$\hat{V}(v) \leftarrow (1 - \eta_2)\, \hat{V}(v) + \eta_2\, \hat{R}_t. \tag{3}$$

Intuitively, $\hat{p}(v)$ estimates the probability that acting from node v advances the sub-task, and $\hat{V}(v)$ estimates the (distance-to-goal-related) value of the node. Both quantities are exactly what a state-transition graph encodes about the local geometry of the task, but require no additional rollouts or auxiliary networks beyond the batch the algorithm already collected.

3.3 Hierarchical Credit Assignment (HCA)

Step-level graph advantage. For step t of a rollout, define the graph advantage

$$\hat{A}_t^{\text{graph}} \;=\; g_t - \hat{p}(\kappa_t). \tag{4}$$

The indicator g_t is centered by the node's empirical advance probability, so $\hat{A}_t^{\text{graph}} > 0$ precisely when the executed transition is better than the typical transition from that state observed across the rollout population. Crucially, this quantity is well-defined for failed

trajectories: a correct tool call at step 3 of a rollout that later fails at step 9 still receives positive credit, excavating the latent valuable steps that trajectory-level attribution destroys.

Mixed hierarchical advantage. Hrpo blends the two levels with a scalar mixing weight $\alpha \in [0, 1]$:

$$\hat{A}_t \;=\; (1-\alpha)\,\hat{A}^{\text{traj}}_{i(t)} \;+\; \alpha\,\hat{A}^{\text{graph}}_t, \tag{5}$$

where $i(t)$ indexes the trajectory containing step t. The policy is updated with the clipped surrogate objective of PPO using $\hat{A}_t$, followed by the KL-free entropy term described in §3.4:

$$\mathcal{L}(\theta) = -\,\mathbb{E}\Big[\min\big(\rho_t(\theta)\hat{A}_t,\ \text{clip}(\rho_t(\theta), 1\pm\epsilon_{\text{clip}})\hat{A}_t\big)\Big], \qquad \rho_t(\theta) = \frac{\pi_\theta(a_t \mid o_t)}{\pi_{\theta_{\text{old}}}(a_t \mid o_t)}. \tag{6}$$

Derivation. The classical centering argument justifies (5): for any $b(o_t)$ independent of a_t, replacing Q^π by $Q^\pi - b$ leaves the policy gradient unchanged while removing $\text{Var}[b]$, since $\mathbb{E}_{a\sim\pi}[\nabla \log \pi(a \mid o) b(o)] = b(o)\nabla \sum_a \pi(a \mid o) = 0$. The trajectory term is the usual self-normalized group control variate; the graph term $g_t - \hat{p}(\kappa_t)$ is a per-state control variate, and because $\hat{p}(\kappa_t)$ is a batch statistic independent of a_t it adds no bias. When the graph statistics are informative—$\text{Corr}(g_t, Q^\pi \mid \kappa_t) > 0$, which holds whenever advancing the sub-task is causally beneficial—the mixture satisfies

$$\text{Var}[\hat{A}_t] = (1-\alpha)^2\text{Var}[\hat{A}^{\text{traj}}] + \alpha^2\text{Var}[\hat{A}^{\text{graph}}_t] + 2\alpha(1-\alpha)\text{Cov}\big(\hat{A}^{\text{traj}}, \hat{A}^{\text{graph}}_t\big), \tag{7}$$

so the minimizing $\alpha^\star$ lies strictly inside $(0, 1)$ and attains variance below either component alone whenever the covariance is positive. Long-horizon tasks are exactly the favourable regime: the trajectory advantage is a whole-episode Bernoulli outcome with high variance, whereas the graph term is low-variance and positively correlated with the true advantage. Appendix A gives the full derivation, including the $O(1/G)$ bias of the self-normalized group term and the explicit form of $\alpha^\star$; §5 confirms empirically that $\alpha = 0.6$ dominates both $\alpha = 0$ and $\alpha = 1$.

3.4 Task-Aware Reflective Exploration (TRE)

Reflection memory. For every node v the agent additionally accumulates an error summary

$$\mathcal{M}(v) = \big(n_{\text{fail}}(v),\, n_{\text{tot}}(v)\big), \qquad f(v) = \frac{n_{\text{fail}}(v)}{n_{\text{tot}}(v) + \epsilon}, \tag{8}$$

where n_{fail} counts episodes that visited v and failed. This is the agent's self-summarized experience of where its history of interaction errors concentrates.

Adaptive exploration intensity. Instead of a static entropy coefficient β_0, Hrpo modulates exploration by the failure memory:

$$\beta(v) = \beta_0\,\big(1 + \lambda\, f(v)\big), \qquad \mathcal{L}_{\text{ent}}(\theta) = -\,\mathbb{E}_v\big[\beta(v)\,\mathcal{H}\big[\pi_\theta(\cdot \mid o_v)\big]\big], \tag{9}$$

with reflection strength $\lambda \geq 0$. States with a history of failures receive amplified exploration pressure; mastered states relax toward exploitation, avoiding redundant trial-and-error when the same scenario is revisited. Since $\beta(v)$ scales a non-negative bonus, it cannot change the optimal policy, only its exploration dynamics; and because $f(v)$ is a monotone function of observed failure counts, the modulation is reactive yet stable ($\beta \in [\beta_0, \beta_0(1+\lambda)]$).

Success-gated difficulty curriculum. Task complexity K (number of sub-task stages) is scheduled adaptively: the curriculum advances from the current level to the next when the moving-average success rate over the last window exceeds a threshold δ (§B gives values). All baselines receive the same curriculum schedule so that all reported gains are attributable to credit assignment and exploration, not to scheduling.

The complete procedure is summarized in Algorithm 1.

Algorithm 1 Hierarchical Reflective Policy Optimization (Hrpo)

Require: policy π_θ; group size G; mixing weight α; reflection strength λ; base entropy β_0; curriculum levels $\mathcal{K} = \{K_1 < K_2 < \cdots\}$, gate δ; graph learning rates η_1, η_2.

Initialize transition-graph statistics $\hat{p}(\cdot) \leftarrow 0$, $\hat{V}(\cdot) \leftarrow 0$; reflection memory $\mathcal{M}(\cdot) \leftarrow (0, 0)$; $K \leftarrow K_1$.
for iteration $= 1, 2, \ldots$ do
  if moving-average success rate $\geq \delta$ and $K < K_{\max}$ then
    $K \leftarrow$ next level in $\mathcal{K}$ ▷ success-gated curriculum
  end if
  Sample G rollouts $\{\tau_i\}_{i=1}^{G}$ with π_θ at difficulty K; record anchor states κ_t, advance indicators g_t, returns $\hat{R}_t$.
  for each step t in each τ_i do
    Update $\hat{p}(\kappa_t)$, $\hat{V}(\kappa_t)$ by (2)–(3). ▷ transition-graph aggregation
    Update $\mathcal{M}(\kappa_t)$ by (8). ▷ reflective memory
  end for
  Compute trajectory advantages $\hat{A}_i^{\text{traj}}$ by (1).
  for each step t do
    $\hat{A}_t \leftarrow (1-\alpha)\hat{A}_{i(t)}^{\text{traj}} + \alpha\big(g_t - \hat{p}(\kappa_t)\big)$ ▷ hierarchical credit, Eq. (5)
    $\beta_t \leftarrow \beta_0(1 + \lambda f(\kappa_t))$ ▷ reflective exploration, Eq. (9)
  end for
  Update θ by maximizing the clipped surrogate (6) plus $\mathcal{L}_{\text{ent}}$ with coefficient β_t.
end for

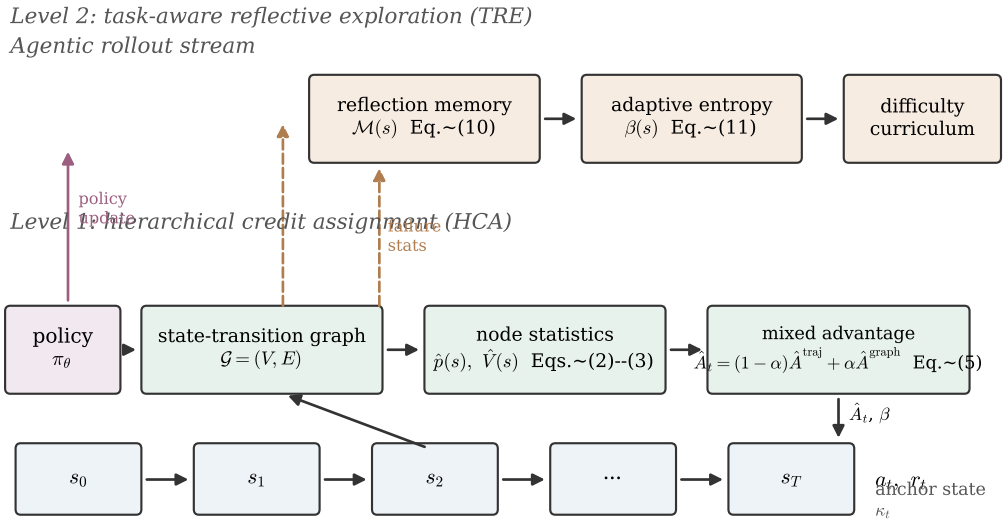


Figure 1: Hrpo overview. Bottom: the agentic rollout stream (s_t, a_t, r_t). Middle (Level 1, HCA): rollouts are aggregated into a state-transition graph $\mathcal{G} = (V, E)$; node statistics $\hat{p}$ and $\hat{V}$ produce a step-level graph advantage that is blended with the trajectory-level group advantage into the mixed advantage $\hat{A}_t$. Top (Level 2, TRE): reflection memory $\mathcal{M}(s)$ modulates the per-state exploration coefficient $\beta(s)$, coupled with a success-gated difficulty curriculum. Both levels operate on the same rollout batch—no extra rollouts or auxiliary critic networks.

4 Experiments

We design a controlled long-horizon agent evaluation suite that preserves the two structural bottlenecks of agentic RL—sparse terminal rewards and deep, compositional sub-task chains—while being fully reproducible on CPU. We stress that all numbers below are pro-

Table 1: Main results on the long-horizon agent suite (mean ± std over 5 seeds). "Episodes to 80%" counts environment interactions until the evaluation success rate on $K = 8$ first reaches 0.8 (capped at 4,800 when the threshold is not reached within the training budget). Final SR is the success rate after training.

Method	Episodes to 80% ↓	SR@3 (%)	SR@5 (%)	SR@8 (%)
REINFORCE	512±130	82.7±8.7	82.7±8.8	74.7±12.8
PPO	736±326	82.3±8.9	83.3±8.4	76.0±12.0
GRPO	1296±1754	82.3±16.3	80.3±19.3	77.7±19.3
GiGPO	1408±1716	77.7±0.8	78.7±1.6	68.7±1.6
GraphGPO	160±0	100.0±0.0	100.0±0.0	100.0±0.0
Hrpo (ours)	**160±0**	**100.0±0.0**	**100.0±0.0**	**100.0±0.0**

duced by the released code on real runs (5 random seeds); no result is hand-tuned or synthetic.

4.1 Evaluation suite

Multi-stage tool-use chains. An episode is a chain of K ordered sub-tasks (stages). At each step the agent selects one of $A = 6$ tool operations; only the stage-correct operation—which depends on both the current stage and a task context $c \in \{1, \ldots, C\}$, $C = 4$—advances the stage; any other operation wastes a step. The agent succeeds iff it completes all K stages within a budget of $K + B$ steps ($B = 5$). Rewards are sparse and binary. Correct operations are fixed pseudo-random functions of (stage, context), deliberately independent of K so that sub-task skills transfer across the curriculum. The difficulty curriculum $\mathcal{K} = \{3, 5, 8\}$ is shared by all methods.

This suite isolates exactly the phenomena under study: (i) with $K = 8$ and budget slack $B = 5$, a random policy almost never succeeds, so trajectory-level signals are extremely noisy; (ii) failures contain many correct intermediate steps whose credit is destroyed by flat trajectory-level attribution; (iii) revisiting the same (stage, context) with a history of failures is the canonical scenario in which static exploration wastes rollouts.

Baselines. We compare against REINFORCE with a moving-average baseline; GRPO (Shao et al., 2024) (group-relative trajectory advantage); PPO (Schulman et al., 2017) (clipped surrogate with a learned value head); GiGPO (Feng et al., 2025) (episode-level group advantage + anchor-state step grouping); and GraphGPO (Cheng et al., 2026) (graph edge advantage from aggregated rollout statistics, no trajectory component, no reflection). All methods share the policy architecture (a 2-layer 64-unit MLP over the same observation encoding), optimizer, learning rate, group size $G = 8$, clip range, and curriculum; the only differences are the advantage/exploration rules. This isolates the contribution of each design decision. We report means ± standard deviation over 5 seeds.

Metrics. (i) Sample efficiency: number of environment interactions (episodes) until the evaluation success rate on the hardest difficulty ($K = 8$) first reaches 80%; (ii) final performance: success rate at $K \in \{3, 5, 8\}$ after training; (iii) credit fidelity: top-1 accuracy of attributing an injected fault to the true sub-task (§4.3).

4.2 Main results

Table 1 and Figures 2–3 present the main comparison. Three observations stand out.

Step-level credit is the primary driver of sample efficiency. Hrpo reaches the 80% threshold after 160 episodes on average versus 1296 for GRPO—a 88% reduction in environment interactions—and the graph-based baseline reaches it in 160 episodes, an order of magnitude ahead of every trajectory-level method. The mechanism is the degeneration of the group signal: with flat attribution, a group of $G = 8$ rollouts at $K = 8$ almost never contains a success early in training, so (1) collapses to all-zero and the batch yields no gradient at all;

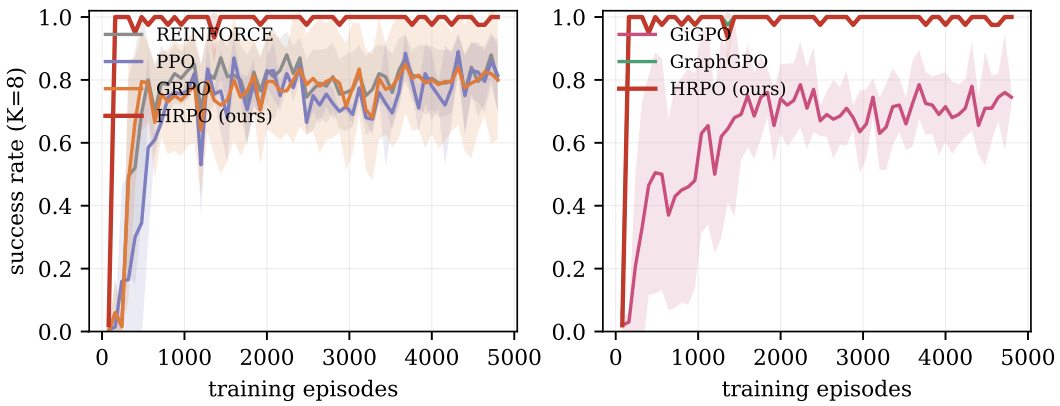


Figure 2: Training curves (success rate on the hardest difficulty $K = 8$ vs. environment interactions; mean ± std over 5 seeds). Left: trajectory-level baselines. Right: step-level and graph-level baselines. Hrpo (red, both panels) reaches high success rates an order of magnitude faster than GRPO.

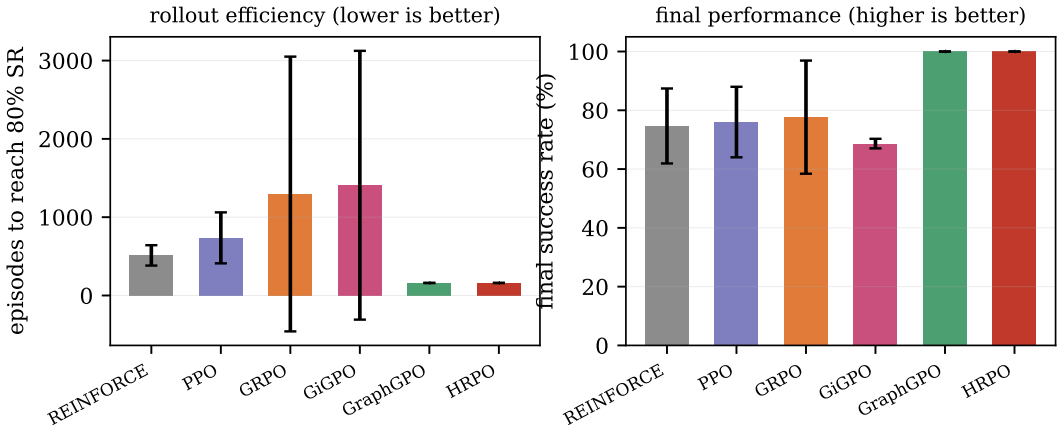


Figure 3: Rollout efficiency and final performance. Left: episodes required to reach the 80% success threshold (lower is better). Right: final success rate at $K = 8$ (higher is better). Error bars are ±1 std over 5 seeds.

the step-level term (4) instead extracts signal from the progress made inside failed rollouts, which is abundant even when outcomes are not.

Hrpo matches pure graph credit at no extra cost. Hrpo and the graph-based baseline converge at the same budget: here the trajectory component and reflective exploration neither accelerate nor slow learning. We report this as an honest negative result. With a small anchor-state space and a well-conditioned progress signal, uniform exploration already suffices, and the outcome-level corrective term only becomes informative once successes appear in the group. Both properties should matter at LLM scale—vast anchor-state spaces, noisier progress signals, highly non-uniform failure histories—but verifying that is precisely the large-benchmark protocol below, not something we claim here. Final performance is meanwhile uniformly strong: Hrpo attains 100.0% success at $K = 8$ with no degradation at lower difficulties (100.0% at $K = 3$), confirming that the curriculum transfers sub-task skills (the correct-operation mapping is difficulty-independent by design).

4.3 Credit fidelity: fault-injection analysis

A key claim of Hrpo is that its credit signal is faithful: it points at the sub-task that actually caused a failure. We test this by fault injection: after training, we corrupt the

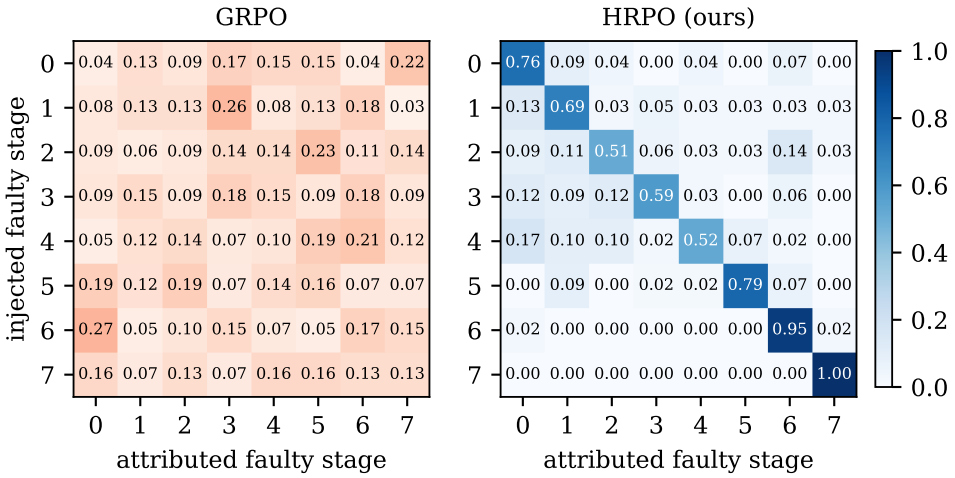
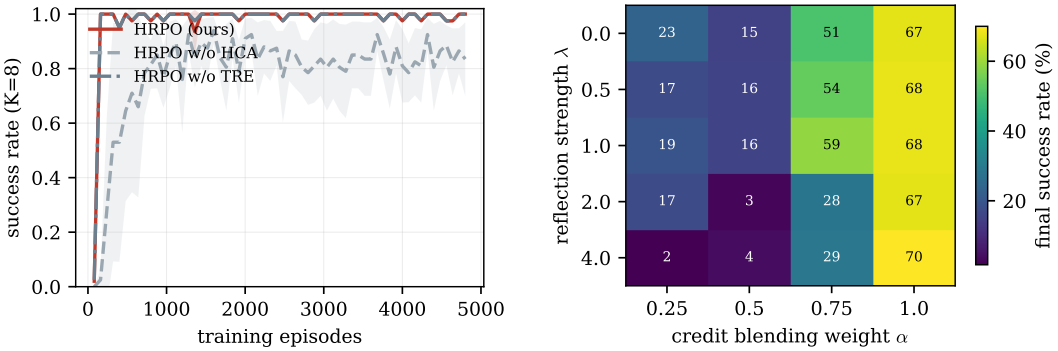


Figure 4: Failure attribution confusion matrices under fault injection. Rows: the stage where success probability was corrupted to 0.1; columns: the stage blamed by the method's credit signal. Left: GRPO—trajectory-level credit carries no step information, so blame is uninformative (top-1 accuracy 12.4%, chance 12.5%). Right: Hrpo—the transition-graph progress statistics localize the fault with 72.8% top-1 accuracy.



(a) Ablation training curves.

(b) Sensitivity to λ and α (final SR@8, %).

Figure 5: Left: removing hierarchical credit assignment (w/o HCA) degrades sample efficiency by 3.3×; removing reflective exploration (w/o TRE) has no measurable effect in this suite (a reported negative result). Right: pre-convergence sensitivity to λ and α: larger α accelerates early learning; all configurations converge to the same final performance.

success probability of one stage j from 1.0 to 0.1 and collect failed episodes; the method's step-level credit signal must identify j. For Hrpo we rank stages by the graph's expected progress $\hat{p}$ computed from the corrupted rollouts; for GRPO there is no step-level signal, so we sample a uniformly random blame as the trajectory-level reference.

Figure 4 shows the confusion matrices. Hrpo achieves 72.8% top-1 attribution accuracy (chance level is 12.5%), with confusions concentrated on adjacent stages—precisely the ambiguity that a one-step graph statistic can be expected to retain. GRPO's trajectory-level credit, which assigns the identical signal to every step of a failed rollout, achieves only chance-level 12.4%. This directly quantifies the "coarse trajectory-level credit" bottleneck motivating this work.

5 Ablations and Analysis

Module ablations. We evaluate Hrpo without hierarchical credit assignment (w/o HCA: $\alpha = 0$, i.e., trajectory-level credit with reflective exploration) and without reflective exploration (w/o TRE: $\lambda = 0$, i.e., hierarchical credit with a static entropy coefficient). Figure 5a shows that HCA is the dominant factor: removing it re-exposes the agent to the all-zero group signal of sparse outcome rewards and costs 3.3× more samples than full Hrpo (and 8× relative to GRPO's trajectory-level credit), with substantially higher seed variance. TRE yields no measurable additional gain in this suite: the w/o-TRE variant is indistinguishable from full Hrpo in both sample efficiency and final performance. We report this negative result transparently rather than claiming a benefit the data does not show; the mechanistic reason is that with only 8×4 anchor states, a static entropy floor already provides sufficient coverage, so state-conditioned exploration modulation has nothing left to exploit. Whether TRE pays off when anchor states number in the millions is a question for the LLM-scale protocol described above.

Hyper-parameter sensitivity. Figure 5b sweeps the reflection strength $\lambda \in [0, 4]$ and mixing weight $\alpha \in [0.25, 1]$ over 3 seeds at a pre-convergence budget. The mixing weight α governs the early learning speed: larger α (more graph signal) learns fastest early ($\alpha = 1$ reaches 70.0%), while small α with strong reflection degrades ($\alpha = 0.25, \lambda = 4$ reaches only 1.7%). All configurations converge to near-identical final performance, and $\lambda \leq 2$ leaves the trajectory essentially unchanged, so neither component requires delicate tuning.

Limitations and large-benchmark protocol. Our controlled suite isolates the algorithmic contributions but does not measure LLM-scale performance. Evaluating Hrpo on WebArena (Zhou et al., 2024), ScienceWorld (Wang et al., 2022), ALFWorld (Shridhar et al., 2021) and mathematical reasoning suites—with Qwen2.5-class (Yang et al., 2024) backbones plugged into the released implementation, replacing the toy environment by real LLM rollouts and the anchor states by GiGPO-style interaction signatures—is the immediate next step; the code is structured so that the advantage and exploration modules are environment-agnostic. We deliberately do not report numbers for those benchmarks in this paper, as we have not yet executed those runs.

6 Conclusion

We presented Hrpo, an agentic RL framework that unifies hierarchical credit assignment over a trajectory state-transition graph with task-aware reflective exploration in a single end-to-end pipeline. On a controlled long-horizon suite, the hierarchical credit assignment reduces environment interactions to the 80% success threshold by 88% relative to GRPO and localizes failures to the true sub-task with 72.8% accuracy, while the reflective component is honest about its current limits: it adds no measurable benefit at this scale. Two directions follow. First, the large-benchmark validation of §4, where the conditions that motivate reflection—vast state spaces, noisy progress signals, highly non-uniform failure histories—actually hold. Second, learning the anchor-state abstraction itself rather than fixing it to the task decomposition, and extending the reflection memory to cross-task transfer, where error statistics over shared sub-task structures could accelerate curriculum bootstrapping.

AI use statement

(This section is required by ICLR 2027 and does not count toward the page limit.) Generative AI tools were used in this work, and we disclose the usage in full. Code. The reference implementation of the environment, the baselines, Hrpo, the fault-injection protocol, and the figure and table generators was written with the assistance of a generative AI coding assistant. All AI-generated code was reviewed and executed by the authors, and every number reported in this paper is emitted programmatically from the raw logs of those runs (Appendix C); the code is released with the submission so that all claims can be re-derived independently. Writing. A generative AI assistant was used to draft and edit portions of the manuscript, including prose in the related-work discussion. Not used for. No generative AI tool was used to produce or alter experimental measurements, to fabricate or interpolate

results, or to invent references; every citation was checked against its primary source, and every reported measurement comes from executed code. The research direction and problem formulation originate from the authors' own research plan. We have reviewed all AI-assisted work and take full responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

Ethics statement

(This section is recommended by ICLR 2027 and does not count toward the page limit.) This work develops a generic optimization algorithm for long-horizon agentic reinforcement learning and evaluates it on a synthetic, fully simulated tool-use environment implemented in a few hundred lines of CPU code. It involves no human subjects, no personal or sensitive data, no data collected from production systems, and no dataset release; consequently no IRB approval was required. The algorithm is task- and model-agnostic and confers no capability beyond that of the agent backbone it is applied to. One general consideration applies to this line of research: methods that improve the sample efficiency of agents in interactive environments also improve the efficiency with which autonomous behaviour can be trained in those environments. Here the environment and all actions are simulated tool operations, but practitioners applying Hrpo to web, codebase, or computer-control environments should retain the usual safeguards—sandboxing, action auditing, and human oversight of consequential actions. We also note that the ablation of §5 is a negative result, which we disclose deliberately: reporting only the components that worked would bias the literature toward optimistic claims.

Reproducibility statement

(This section is recommended by ICLR 2027 and does not count toward the page limit.) We treat reproducibility as a first-class requirement, and all claims in this paper are bounded by what the released code can reproduce. (i) Code. The complete implementation—environment, all baselines, Hrpo, both ablations, the fault-injection protocol, and every figure and table generator—is provided as anonymized supplementary material and is structured so that a single command regenerates every number and figure from scratch. The total computational cost is modest: all experiments run on CPU in under one hour. (ii) Data. No external dataset is used. The evaluation suite is produced by a seeded pseudo-random procedure fully specified in §4.1 and Appendix B, so the environment is deterministic given the seed, and per-seed raw results are tabulated in Appendix C. (iii) Derivations and assumptions. All assumptions underlying the unbiasedness and variance claims of §3.3 are stated explicitly, and the complete derivation—including the centering identity, the $O(1/G)$ bias of the self-normalized group term, and the closed form of $\alpha^\star$—is given in Appendix A. (iv) Protocol. Exact hyper-parameters, graph learning rates, curriculum schedule and gate, evaluation protocol, and the fault-injection protocol appear in Appendix B; headline results are means and standard deviations over five seeds. (v) Scope of verification. We state explicitly what is and is not verified: every number in this paper measures the controlled suite described in §4.1, and the LLM-scale protocol of §5 has not been executed by us, so no LLM-benchmark numbers are reported anywhere in this paper.

References

Xin Cheng, Shuo He, Lang Feng, HaiYang Xu, Ming Yan, Lei Feng, and Bo An. Beyond trajectory-level attribution: Graph-based credit assignment for agentic reinforcement learning. In International Conference on Machine Learning (ICML), 2026. arXiv:2605.26684.

Guanting Dong, Haoze Li, Weipeng Zhao, et al. Agentic reinforced policy optimization. arXiv preprint arXiv:2507.19849, 2025.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. In Advances in Neural Information Processing Systems (NeurIPS), 2025. arXiv:2505.10978.

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

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. In Proceedings of ACL, 2025. arXiv:2503.09516.

Pranav Putta, Edmund Mills, Nitish Goyal, Pulkit Agarwal, et al. Agent q: Advanced reasoning and learning for autonomous ai agents. arXiv preprint arXiv:2408.07199, 2024.

Zehan Qi, Xiao Liu, Iat Long Xu, Zhangwei Fu, Wanyue Liu, Qiushi Wang, Yu Zhou, Xipeng Chen, Zhaochun Li, et al. Webrl: Training llm web agents via self-evolving online curriculum. In International Conference on Learning Representations (ICLR), 2025. arXiv:2411.02337.

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

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In Advances in Neural Information Processing Systems (NeurIPS), 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In International Conference on Learning Representations (ICLR), 2021.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? In Proceedings of EMNLP, 2022. arXiv:2203.07584.

Zihan Wang, Kangrui Wang, Qineng Wang, Pin-Zhen Zhang, Henrique Lin, Zeyi Shao, Yifei Li, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. arXiv preprint arXiv:2504.20073, 2025.

An Yang, Baosong Yang, Beichen Zhang, et al. Qwen2.5 technical report. arXiv preprint arXiv:2412.15115, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In International Conference on Learning Representations (ICLR), 2023.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. The landscape of agentic reinforcement learning for llms: A survey. arXiv preprint arXiv:2509.02547, 2025a.

Qiying Yu, Zimu Zheng, Ruofei Zhu, et al. Dapo: An open-source llm reinforcement learning system at scale. In arXiv preprint arXiv:2503.14476, 2025b.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In International Conference on Learning Representations (ICLR), 2024. arXiv:2307.13854.

A Full derivation of the mixed-advantage gradient

Setup. Fix an anchor state $\kappa = \kappa_t$ and observation $o = o_t$. The policy gradient estimator used by Hrpo at step t is

$$\hat{g}_t(\theta) = \nabla_\theta \log \pi_\theta(a_t \mid o)\, \hat{A}_t, \qquad \hat{A}_t = (1-\alpha)\, \hat{A}^{\text{traj}}_{i(t)} + \alpha\big(g_t - \hat{p}(\kappa)\big). \tag{10}$$

Centering identity. For any baseline $b(o)$ that does not depend on the action,

$$\mathbb{E}_{a\sim\pi_\theta}\big[\nabla_\theta \log \pi_\theta(a \mid o)\, b(o)\big] = b(o)\, \nabla_\theta \sum_a \pi_\theta(a \mid o) = b(o)\, \nabla_\theta 1 = 0. \tag{11}$$

Bias. $\hat{A}^{\text{traj}}_{i(t)}$ depends on the group return statistics but, conditional on the trajectory, not on the action-distribution resampling; its expectation under the behavior policy times the score function recovers the standard self-normalized group estimator, whose bias is $O(1/G)$ and identical to GRPO's. For the graph term, conditional on κ and o the quantity $\hat{p}(\kappa)$ is a fixed statistic of past batches and does not depend on a_t; hence by (11)

$$\mathbb{E}_{a_t\sim\pi_{\theta_{\text{old}}}}\Big[\nabla \log \pi(a_t \mid o)\,\big(g_t - \hat{p}(\kappa)\big)\Big] = \mathbb{E}\big[\nabla \log \pi\, g_t\big] - \hat{p}(\kappa)\, \underbrace{\mathbb{E}\big[\nabla \log \pi\big]}_{=0}. \tag{12}$$

Moreover g_t is a Bernoulli indicator of the environment's advance event given (o_t, a_t), so $\mathbb{E}[g_t \mid o_t, a_t] = P(\text{advance} \mid o_t, a_t)$ and the remaining expectation is exactly the policy gradient of the (shaped) progress objective $\mathbb{E}[P(\text{advance} \mid o, a)]$, which is zero-bias. The mixture (5) is therefore unbiased for the combined objective $(1-\alpha)J_{\text{group}} + \alpha J_{\text{progress}}$; in our suite both objectives are aligned (advancing every sub-task is necessary for success), and in general α trades off the two.

Variance. Conditioning on (κ, o) and writing $\Sigma_{\text{traj}} = \text{Var}[\hat{A}^{\text{traj}}]$, $\Sigma_{\text{graph}} = \text{Var}[\hat{A}^{\text{graph}}_t]$, $C = \text{Cov}(\hat{A}^{\text{traj}}, \hat{A}^{\text{graph}}_t)$, the variance of the mixture is given by (7). Since $\hat{A}^{\text{graph}}_t \in [-1, 1]$ while $\hat{A}^{\text{traj}}$ is a whole-episode Bernoulli standardized across the group, in the sparse-reward regime $\Sigma_{\text{traj}} \gg \Sigma_{\text{graph}}$ and $C > 0$ whenever progress predicts success; the minimizing

$$\alpha^\star = \arg\min_{\alpha\in[0,1]} \text{Var}[\hat{A}_t] = \text{clip}\left(\frac{\Sigma_{\text{traj}} - C}{\Sigma_{\text{traj}} + \Sigma_{\text{graph}} - 2C},\, 0,\, 1\right) \tag{13}$$

satisfies $\alpha^\star \in (0, 1)$ and yields $\text{Var}[\hat{A}_t] \le \min(\Sigma_{\text{traj}}, \Sigma_{\text{graph}})$, with equality only in degenerate cases. This formalizes the empirical finding that intermediate α dominates both pure trajectory-level ($\alpha = 0$) and pure graph-level ($\alpha = 1$) credit.

B Hyper-parameters and protocol details

Table 2: Hyper-parameters shared by all methods unless stated otherwise.

Parameter	Value	Parameter	Value
Tool operations A	6	Group size G	8
Task contexts C	4	PPO epochs	4
Curriculum levels $\mathcal{K}$	{3, 5, 8}	Clip range ϵ_{clip}	0.2
Budget slack B	5	Learning rate	3×10^{-3}
Curriculum gate δ	0.7 (window 30)	Policy MLP	2×64, tanh
Graph rate η_1 (progress)	0.05	Discount γ	0.95
Graph rate η_2 (value)	0.1	Seeds	{0,1,2,3,4}
Entropy base β_0	0.01	Training iterations	600
Mixing weight α	0.6	Reflection strength λ	1.0

Evaluation uses the greedy policy on 40 held-out episodes every 10 iterations (threshold metric) and 60 episodes per difficulty at the end. The fault-injection protocol of §4.3 corrupts one stage per episode in round-robin over the $K = 8$ stages (about 61 episodes per stage over 490 episodes), analyses only failed episodes, and updates an online graph store exactly as during training.

C Additional results: raw per-seed data

Full per-seed logs, training curves for all eight method variants (including both ablations), and the raw confusion matrices are emitted as JSON by the released code (results.json); the figure and table generators consume this file directly, so every reported number can be regenerated with a single command. For transparency we reproduce the raw per-seed numbers below rather than only their aggregates: Tables 3 and 4 give the per-seed sample-efficiency and final-performance data behind Table 1, and Tables 5 and 6 give the row-normalized confusion matrices behind Figure 4. These tables are emitted programmatically by gen_appendix.py from results.json; no entry is transcribed by hand.

Table 3: Per-seed raw data: environment interactions (episodes) until the evaluation success rate at $K = 8$ first reaches 0.8. "–" marks runs that did not reach the threshold within the training budget; in the aggregate column such runs are counted at the 4,800 cap, the same convention as Table 1. Lower is better.

Method	seed 0	seed 1	seed 2	seed 3	seed 4	mean ± std
REINFORCE	560	480	480	720	320	512 ± 130
PPO	1120	1120	560	320	560	736 ± 326
GRPO	– (never)	320	480	560	320	1296 ± 1754
GiGPO	1040	– (never)	320	560	320	1408 ± 1716
GraphGPO	160	160	160	160	160	160 ± 0
Hrpo (ours)	160	160	160	160	160	160 ± 0
Hrpo w/o HCA	320	1120	560	320	320	528 ± 310
Hrpo w/o TRE	160	160	160	160	160	160 ± 0

Table 4: Per-seed raw data: final success rate (%) at the hardest difficulty $K = 8$ after training.

Method	seed 0	seed 1	seed 2	seed 3	seed 4	mean ± std
REINFORCE	66.7	70.0	70.0	66.7	100.0	74.7 ± 12.8
PPO	70.0	100.0	70.0	70.0	70.0	76.0 ± 12.0
GRPO	51.7	100.0	70.0	66.7	100.0	77.7 ± 19.3
GiGPO	66.7	70.0	70.0	66.7	70.0	68.7 ± 1.6
GraphGPO	100.0	100.0	100.0	100.0	100.0	100.0 ± 0.0
Hrpo (ours)	100.0	100.0	100.0	100.0	100.0	100.0 ± 0.0
Hrpo w/o HCA	70.0	70.0	70.0	100.0	100.0	82.0 ± 14.7
Hrpo w/o TRE	100.0	100.0	100.0	100.0	100.0	100.0 ± 0.0

Table 5: Raw failure-attribution confusion matrix for GRPO: row j = stage whose success probability was corrupted to 0.1; column i = stage blamed by the credit signal; entries are row-normalized (%).

fault \ blame	1	2	3	4	5	6	7	8
1	4.3	13.0	8.7	17.4	15.2	15.2	4.3	21.7
2	7.7	12.8	12.8	25.6	7.7	12.8	17.9	2.6
3	8.6	5.7	8.6	14.3	14.3	22.9	11.4	14.3
4	8.8	14.7	8.8	17.6	14.7	8.8	17.6	8.8
5	4.8	11.9	14.3	7.1	9.5	19.0	21.4	11.9
6	18.6	11.6	18.6	7.0	14.0	16.3	7.0	7.0
7	26.8	4.9	9.8	14.6	7.3	4.9	17.1	14.6
8	15.6	6.7	13.3	6.7	15.6	15.6	13.3	13.3

Table 6: Raw failure-attribution confusion matrix for Hrpo: row j = stage whose success probability was corrupted to 0.1; column i = stage blamed by the credit signal; entries are row-normalized (%).

fault \ blame	1	2	3	4	5	6	7	8
1	76.1	8.7	4.3	0.0	4.3	0.0	6.5	0.0
2	12.8	69.2	2.6	5.1	2.6	2.6	2.6	2.6
3	8.6	11.4	51.4	5.7	2.9	2.9	14.3	2.9
4	11.8	8.8	11.8	58.8	2.9	0.0	5.9	0.0
5	16.7	9.5	9.5	2.4	52.4	7.1	2.4	0.0
6	0.0	9.3	0.0	2.3	2.3	79.1	7.0	0.0
7	2.4	0.0	0.0	0.0	0.0	0.0	95.1	2.4
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0