# RTPO: REVERSE-TURN POLICY OPTIMIZATION FOR STABILIZING AGENTIC RL TRAINING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Training multi-turn agentic workflows with reinforcement learning (RL) enables large language models to perform complex reasoning, use external tools, and conduct iterative search beyond single-turn settings. Yet multi-turn RL training remains highly unstable, often causing severe performance degradation as the number of turns increases. Through theoretical analysis, we identify three tightly coupled sources of instability: rollout–training context mismatch, weak turn-level credit assignment under sparse terminal rewards, and asynchronous policy drift when short and long trajectories are optimized under different policy versions. We show that these issues share a common structural origin in flattened trajectory optimization and address them through a unified reverse-turn formulation. We propose *Reverse-Turn Policy Optimization (RTPO)*, which organizes multi-turn rollouts as sparse reverse trees and performs turn-level policy updates in temporal reverse order, aligning each decision with its downstream continuation. RTPO enables causally consistent turn-level credit assignment and on-policy continuation to control asynchronous drift. We provide theoretical results showing that RTPO eliminates rollout–training context mismatch under matched conditioning, reduces turn-level credit bias, and enables drift-free on-policy continuation, while further establishing convergence of reverse-order optimization to recursive optimality. Experiments on multi-turn agentic RL benchmarks show that RTPO improves upon trajectory- and turn-level baselines by 21.50% and 10.76%, respectively, highlighting its potential to support more stable training for tool-using agents.

## 1 INTRODUCTION

Reinforcement learning (RL) has become a central paradigm for post-training large language models (LLMs), especially when supervision comes from final outcome rewards rather than dense token-level labels. Through outcome-based optimization, RL encourages behaviors such as planning, self-reflection, and verification, achieving strong results in single-turn mathematical reasoning (Shao et al., 2024; Yang et al., 2024; Guo et al., 2025; Li et al., 2026) and code generation (Le et al., 2022; Shojaee et al., 2023; Team et al., 2025). Motivated by these advances, recent work extends RL to multi-turn agentic workflows, especially Tool-Integrated Reasoning (TIR) (Yao et al., 2023; Gao et al., 2023; Shinn et al., 2023; Chen et al., 2026; Chang et al., 2026; Liu et al., 2026), where agents iteratively reason, call tools, receive feedback, and refine their behavior across turns.

Despite this promise, *RL training for multi-turn workflows remains unstable*: models that improve on short workflows often degrade as the number of turns increases. This degradation is not merely due to longer sequences; it is amplified by temporal dependencies across turns, where each decision reshapes the context, environment state, and future decision distribution. Existing methods partially mitigate this issue but provide limited analysis of its causes. Trajectory-level methods such as PPO (Schulman et al., 2017), GRPO (Shao et al., 2024), and GSPO (Zheng et al., 2025) retain the flattened trajectory paradigm. Turn-decomposition methods such as SeeUPO (Hu et al., 2026) improve update granularity but still condition on flattened histories. Tree-based methods such as TreeGRPO (Ji et al., 2026) and ARPO (Dong et al., 2026) use shared prefixes or branching rollouts, but their advantage estimates are not fully aligned with each turn's causal contribution to downstream continuation. Overall, the sources of instability remain underexplored at the training-pipeline level, and existing methods do not jointly address their shared origin (see *Appendix A*).



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.

In this paper, **we provide a theoretical analysis that identifies three coupled sources of instability**: context mismatch between rollout and training, which breaks consistency between generated and optimized turn-level contexts; weak turn-level credit assignment, where sparse terminal rewards obscure the contribution of individual decisions; and asynchronous policy drift, where short and long trajectories are optimized under different versions of an evolving policy. Although these issues arise from different components of the training pipeline, we show that they share a common structural origin. We illustrate these sources in **Figure 1** and analyze them formally in *Section 2*.

> **Core Challenge:** Motivated by our theoretical analysis, we ask: How can we improve multi-turn agentic RL performance by stabilizing turn-wise training with rollout–training consistency, turn-level credit assignment, and controlled asynchronous policy drift?

To address this challenge, we propose *Reverse-Turn Policy Optimization (RTPO)*, a policy optimization framework for stabilizing multi-turn agentic RL training, with theoretical details provided in *Section 3*. The key idea is to formalize sampled interactions as sparse trees and optimize turn-level policies in temporal reverse order, propagating continuation values from later turns to earlier ones following the reverse-order optimization principle analyzed in the tabular setting. By constructing sibling continuations for each turn, RTPO estimates turn-level advantages under matched downstream conditions. This removes context inconsistency induced by flattened trajectory optimization, reduces turn-level credit bias through causal action alignment, and controls asynchronous policy drift through on-policy continuation.

**Our contributions are fourfold**: **(1)** We identify coupled sources of instability in multi-turn agentic RL: context mismatch, weak turn-level credit, and asynchronous policy drift. **(2)** We provide a theoretical analysis showing that these sources share a common structural origin in flattened trajectory optimization. **(3)** We propose RTPO, a reverse-turn policy optimization framework with sparse reverse trees, turn-level on-policy updates, with a tabular convergence guarantee for reverse-order optimization, together with theoretical results on context consistency, turn-level credit estimation, and on-policy continuation. **(4)** We validate RTPO on multi-turn agentic RL benchmarks, where it outperforms strong baselines, including GRPO, TreeGRPO, ARPO, and SeeUPO, while further stabilizing the training pipeline.

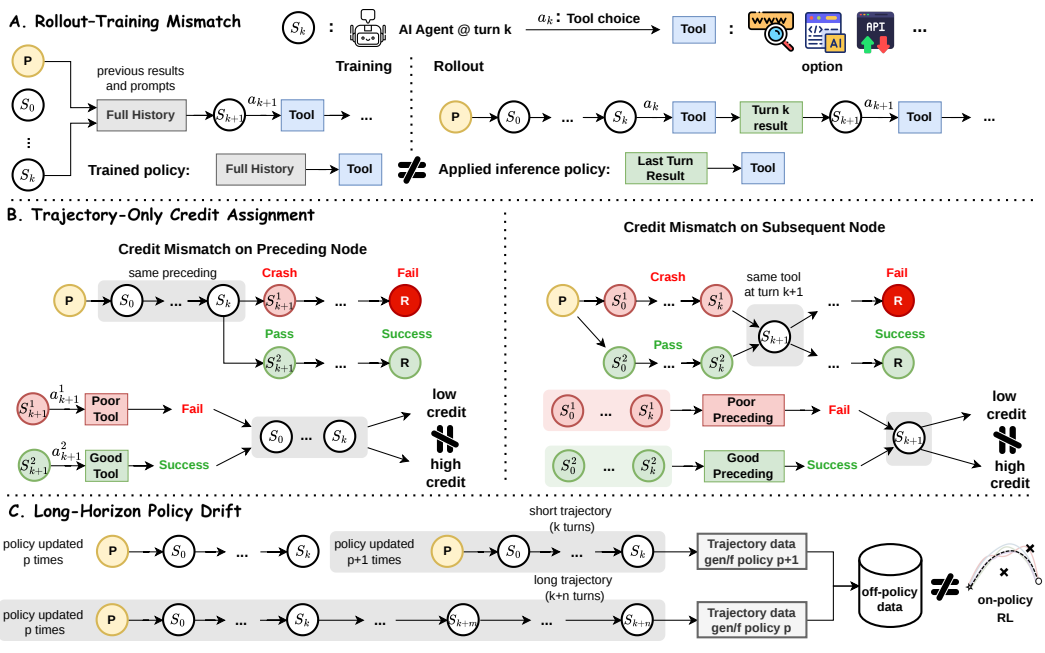



Figure 1: **The identified training instability in multi-turn agentic RL** arises from: (A) rollout–training context mismatch, (B) trajectory-only credit assignment, and (C) long-horizon policy drift.

## 2 THEORETICAL ANALYSIS: TRAINING INSTABILITY

We argue that the instability of multi-turn RL training stems from *rollout–training mismatch*: rollouts are typically generated under truncated or summarized contexts, while training recomputes likelihood ratios under concatenated full-history contexts. This discrepancy biases policy optimization and worsens over long horizons. Moreover, the flattened full-history formulation provides only trajectory-level credit, causing terminal rewards to obscure individual turns. When trajectories are generated asynchronously, this mismatch further induces policy drift, as long trajectories generated under an older policy may be optimized after the policy has already been updated by shorter trajectories.

Building on these observations and insights, we model a multi-turn interaction as a hierarchical Markov decision process (H-MDP) (Hauskrecht et al., 1998). Given an initial prompt $q$, turn $k$ consists of a model response $l_k$ and environment feedback $f_k$, producing the trajectory $(q, l_0, f_0, \ldots, l_{n-1}, f_{n-1})$. The turn-level state is $S_k = (q, l_0, f_0, \ldots, l_{k-1}, f_{k-1})$, and the turn-level action is the response $l_k$. Each response is generated autoregressively as $l_k = (a_{k,1}, \ldots, a_{k,T_k})$, where the token-level state is $s_{k,t} = (S_k, a_{k,<t})$. Existing PPO- and GRPO-style methods (Shao et al., 2024; Zheng et al., 2025; Yue et al., 2025; Yu et al., 2025) typically flatten the full interaction into a single token sequence and optimize the resulting trajectory as follows:

$$J^{\text{flat}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{\sum_t m_{i,t}}\sum_t m_{i,t}\min(\rho_{i,t}A_i, \text{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon)A_i)\right], \tag{1}$$

where $\rho_{i,t} = \pi_\theta(a_{i,t} \mid x_{i,<t})/\pi_{\theta_{\text{old}}}(a_{i,t} \mid x_{i,<t})$ is the token-level importance-sampling (IS) ratio and $A_i$ is a trajectory-level advantage assigned uniformly to all unmasked tokens in trajectory $g_i$ ($g_i$ belongs to a group of $G$ trajectories). Full preliminaries are provided in Appendix B.

### 2.1 ROLLOUT–TRAINING MISMATCH

As illustrated in **Figure 1-A**, in multi-turn interactions, rollouts are often generated under a truncated or summarized context $\phi(\bar{x}_k)$, while training recomputes token probabilities under the full flattened history $\bar{x}_k$. Thus, the IS ratio used in training differs from the true IS ratio induced by the rollout distribution:

$$\rho_{k,t}^{\text{flat}} = \frac{\pi_\theta(a_{k,t} \mid \bar{x}_k)}{\pi_{\theta_{\text{old}}}(a_{k,t} \mid \bar{x}_k)} \neq \frac{\pi_\theta(a_{k,t} \mid \phi(\bar{x}_k))}{\pi_{\theta_{\text{old}}}(a_{k,t} \mid \phi(\bar{x}_k))} = \rho_{k,t}^{\text{true}}. \tag{2}$$

Because the denominator in $\rho_{k,t}^{\text{flat}}$ does not match the distribution that actually sampled the token, the resulting policy-gradient estimate is biased. This mismatch becomes more severe in later turns as the omitted history grows. Moreover, when $\phi$ is non-injective, distinct full-history states can collapse into the same truncated observation, inducing state aliasing and restricting optimization to an observation-induced policy class whose optimum may be strictly below the full-history optimum.

See *Appendix B.1* for the full theoretical analysis of rollout–training mismatch.

### 2.2 TRAJECTORY-ONLY CREDIT ASSIGNMENT

Flattened training assigns a single trajectory-level advantage to all turns, even though different turns may contribute unequally to the final outcome, as shown in **Figure 1-B**. For trajectory $g_i$, the population trajectory advantage can be decomposed at turn $k$ as

$$R_i - \mathbb{E}[R] = \underbrace{R_i - Q^\pi(S_{i,k}, l_{i,k})}_{\text{downstream stochasticity}} + \underbrace{A^\pi(S_{i,k}, l_{i,k})}_{\text{true turn advantage}} + \underbrace{V^\pi(S_{i,k}) - \mu_R}_{\text{upstream state effect}}. \tag{3}$$

Here, $R_i$ denotes the final return of trajectory $g_i$, $Q^\pi(S_{i,k}, l_{i,k}) = \mathbb{E}[R \mid S_{i,k}, l_{i,k}]$, $V^\pi(S_{i,k}) = \mathbb{E}[R \mid S_{i,k}]$, and $\mu_R = \mathbb{E}[R]$. This decomposition shows that the true turn-level advantage is only one component of the trajectory-level signal. When upstream state effects or downstream stochasticity dominate, the sign of the trajectory advantage may disagree with the true turn-level advantage, changing individual samples' contributions to the clipped objective. Group-relative baselines provide state-matched comparisons when trajectories share the same turn-level state. Under cross-state grouping, the estimator incurs an additional conditional advantage-estimation bias $\text{Bias}_{\text{cross}} = \frac{G-1}{G}(V^\pi(S_{i,k}) - \mu_R)$, determined by the upstream state.

See *Appendix B.2* for the full theoretical analysis of trajectory-only credit assignment issues.

### 2.3 Long-Horizon Policy Drift

In asynchronous training with parallel sampled trajectories (see **Figure 1-C**), shorter trajectories may complete and update the policy while longer trajectories are still being generated under an older policy. When these longer trajectories are later optimized, they become off-policy with respect to the current model. The unbiased correction would require the full-trajectory importance weight $\omega_i = \prod_{t=1}^{T_i} \rho_{i,t}$. However, the PPO or GRPO method applies clipping independently at the token level, so the resulting correction differs from the true full-trajectory IS ratio:

$$\prod_{t=1}^{T_i} \mathrm{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon) \neq \prod_{t=1}^{T_i} \rho_{i,t} = \omega_i. \tag{4}$$

Therefore, token-level clipping cannot faithfully correct long-horizon policy drift. The discrepancy compounds with trajectory length, while using the exact full-trajectory ratio is impractical because its variance grows rapidly with $T_i$. This explains why standard PPO- or GRPO-style training methods can become unstable or even collapse in long-horizon agentic RL.

The full theoretical analysis of long-horizon policy drift is provided in *Appendix B.3*.

## 3 Method: Reverse-Turn Policy Optimization (RTPO)

To address the training instability revealed by our theoretical analysis in *Section 2*, *we propose Reverse-Turn Policy Optimization (RTPO) with theoretical guarantees for stabilizing agentic RL training*, as shown in **Figure 2**. RTPO first models multi-turn interaction as a turn-boundary MDP and defines an independent sub-policy optimization objective for each turn (*Section 3.1*). This formulation enables reverse-order training to mitigate the mismatch between rollout and training contexts. RTPO then develops sparse tree rollouts based on maximum-value decomposition (*Section 3.2*) to estimate true turn-level advantages, enabling causally consistent turn-level credit assignment for the case of cross-trajectory comparison without state bias. Finally, RTPO designs an on-policy continuation mechanism (*Section 3.3*) that eliminates the need for downstream IS-ratio correction under PPO clipping, thereby addressing policy drift induced by asynchronous turns.

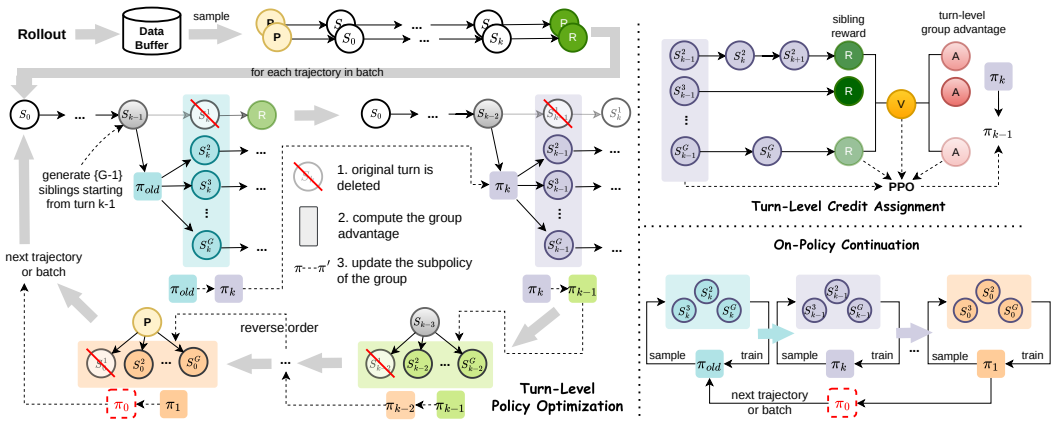



Figure 2: **Overview of RTPO in agentic RL training.** After rollout, RTPO performs reverse-order *turn-level policy optimization* for each trajectory in the batch. Starting from the final turn $k$, the rollout (old) policy $\pi_{\text{old}}$ generates $G-1$ sibling rollouts from the same turn boundary to estimate a group advantage and update the training (turn-level) policy to $\pi_k$. The procedure then proceeds backward through turns $k-1, k-2, \ldots, 0$, where each updated turn policy as $\pi_{k-1}, \pi_{k-2}, \ldots, \pi_0$. To generate state-matched siblings for *turn-level credit assignment*, we design a sparse tree structure that assigns group-relative advantages to individual sibling rollouts at each turn and propagates optimization backward across the trajectory. Finally, RTPO applies *on-policy continuation* to coordinate asynchronous short- and long-trajectory updates from $\pi_k$ to $\pi_0$, reducing policy drift.

### 3.1 TURN-LEVEL POLICY OPTIMIZATION

**Turn-boundary MDP for rollout-training match.** We model a $K$-turn agent episode as a turn-boundary MDP $\mathcal{M} = \langle \bar{\mathcal{X}}, \mathcal{A}_H, P_H, R_H, \gamma_H \rangle$, where the augmented state $\bar{x}_k = (S_k, k)$ encodes the interaction history $S_k$ and the turn index $k$. At the turn-$k$ boundary, the agent selects a macro-action $u_k \equiv l_k \in \mathcal{A}_{H,k}(S_k)$, corresponding to the complete turn-$k$ response $l_k = (a_{k,1}, \ldots, a_{k,T_k})$. Executing $u_k$ consumes $\tau_k = T_k$ token steps, after which the environment returns to the external tool feedback $f_k$, and the interaction history is updated as $S_{k+1} = S_k \circ (l_k, f_k)$. To define the conditioning context received by the model at turn $k$, we let $c_k = \psi(S_k)$, where $\psi$ can be the identity map, a truncation operator, or a summarization operator. The turn-level policy is factorized as $\pi_\theta = (\pi_{\theta,0}, \ldots, \pi_{\theta,K-1})$, where each sub-policy is autoregressive at the token level: $\pi_{\theta,k}(u_k|c_k) = \prod_{t=1}^{T_k} \pi_\theta(a_{k,t}|c_k, a_{k,<t})$.

This turn-boundary MDP decomposition factorizes the episode policy into $K$ turn-level sub-policies, with each sub-policy mapping the conditioning context $c_k = \psi(S_k)$ to a complete response $l_k$. Here, $c_k$ is kept identical between rollout and training. During rollout, RTPO records the exact context $c_k$ received by the model, including any truncation or summarization, together with the corresponding old-policy log-probabilities. During training, the same $c_k$ is used as input, and the loss is computed only over the output tokens in $l_k$. Hence, the denominator of the IS ratio is evaluated under the same conditioning context as in rollout:

$$\rho_{k,t}^{\mathrm{RTPO}} = \frac{\pi_\theta(a_{k,t}|c_k, a_{k,<t})}{\pi_{\theta_{\mathrm{old}}}(a_{k,t}|c_k, a_{k,<t})} \tag{5}$$

**Per-turn policy optimization under reverse-order training.** Training proceeds in reverse order through the turns $k = K-1, K-2, \ldots, 0$. After turn $k$ is completed, its output tokens are excluded from subsequent losses.

During reverse-order training for turn-$k$, the environment is forked from the trunk trajectory's boundary state $S_k$ to generate $G-1$ sibling rollouts in the sparse tree. Each sibling rollout $j$ receives a turn-level advantage $A_{j,k}^H$ (the details of sibling rollout generation are provided in *Section 3.2*). The policy optimization objective for turn $k$ is then defined as:

$$J_k(\theta) = \frac{1}{G-1} \sum_{j=1}^{G-1} \frac{1}{T_{j,k}} \sum_{t=1}^{T_{j,k}} \min\Big( \rho_{j,k,t}\, A_{j,k}^H,\ \mathrm{clip}\left(\rho_{j,k,t},\, 1-\epsilon,\, 1+\epsilon\right) A_{j,k}^H \Big) \tag{6}$$

where $\rho_{j,k,t} = \pi_\theta(a_{j,k,t}|c_k, a_{j,k,<t}) / \pi_{\theta_{\mathrm{old}}}(a_{j,k,t}|c_k, a_{j,k,<t})$ is the token IS ratio and $T_{j,k}$ is the number of tokens generated by sibling rollout $j$ at turn $k$. Only these $G-1$ sibling rollouts are used for gradient updates; the trunk trajectory is excluded.

In **Theorem 1**, we establish convergence guarantees for reverse-order tabular value updates in the turn-boundary MDP.

**Theorem 1: Convergence to Recursive Optimality**

Under standard assumptions, finite state and macro-action spaces, Robbins–Monro step sizes, sufficient per-turn exploration, frozen downstream policies, and bounded rewards, reverse-order tabular updates satisfy:

(a) **Per-turn convergence.** For each turn $k$, fixing downstream policies reduces optimization to single-step Q-learning with a stationary continuation target; hence, tabular value estimates converge, and greedy policies attain optimal turn value asymptotically.

(b) **Recursive optimality.** Under reverse-order training, applying the above argument recursively from turn $K-1$ to turn $0$ yields a sequence of per-turn policies that is recursively optimal.

(c) **Global optimality.** If the turn-level macro-action spaces are complete, i.e., the per-turn policy class can represent any globally feasible trajectory-level policy, then recursive optimality is equivalent to global optimality over the full trajectory.

The proof of Theorem 1 is provided in *Appendix C.2*.

### 3.2 TURN-LEVEL CREDIT ASSIGNMENT

**Turn-level advantage function.** We construct sparse tree rollouts at turn-level boundaries, elevating the advantage granularity from trajectory-only reward to turn-level credit while ensuring that all compared rollouts share the same state and are free from state bias. In turn $k$, $G-1$ siblings independently generate turn-$k$ responses from the shared boundary state $S_k$ and continue to the terminal. The turn-level advantage function for sibling $j$ is: $A^H_{j,k} = \hat{Q}_{j,k} - \hat{V}_k$.

Here, all siblings share the same state $S_k$, so advantage differences can only arise from two sources: different action choices at turn $k$ and independent downstream sampling noise. The downstream noise has zero mean ($\mathbb{E}[\xi_{\text{down}}|S_k, l_{j,k}] = 0$) and introduces no systematic bias; the upstream state term from Eq. (3) cancels in sibling comparisons, because all siblings share $S_k$. Furthermore, $A^H_{j,k}$ is assigned only to the output tokens of turn $k$; the shared prefix $S_k$ serves as the prompt input but does not enter the loss, so each token appears exactly once in the training batch.

**Turn-level value estimation.** To identify turn-level values, we separate the local reward at each turn from the downstream completion value over the full trajectory. Based on the MAXQ principle (Dietterich, 2000), we define the following action-value decomposition for turn $k$: $\tilde{Q}^\pi_k(S_k, u_k) = r_k + \gamma^{\tau_k} F^\pi_k(S_{k+1})$. $r_k$ is the immediate reward at turn $k$ ($r_k = 0$ for $k < K-1$ under sparse rewards), and $F^\pi_k(S_{k+1})$ is the downstream continuation value representing the expected cumulative return from turn $k+1$ onward under policy $\pi$, with base case $F^\pi_{K-1} \equiv 0$. Then, the terminal reward $R_j$ obtained by sibling $j$ rolling out the policy from $S_k$ to the terminal is a single Monte Carlo sample of the above:

$$\hat{Q}_{j,k} = R_j = r_{j,k} + \gamma^{\tau_k} \hat{F}_{j,k} \tag{7}$$

where $\hat{F}_{j,k}$ is a single sample of $F^\pi_k(S_{k+1})$. $R_j$ is an unbiased estimator of $\tilde{Q}^\pi_k$. The sampled continuation $\hat{F}_{j,k}$ evaluates the current action under the downstream policy. *Section 3.3* describes on-policy continuation using the current policy. After estimating the action values at each turn, we compute a state-specific value baseline from the sibling rollouts. Specifically, we define $\hat{V}_k$ as the mean of the estimated $Q$-values across the $G-1$ siblings: $\hat{V}_k = \frac{1}{G-1} \sum_{j=1}^{G-1} R_j$.

In **Theorem 2**, we show that RTPO isolates turn-level credit by removing upstream state contamination and characterize its finite-group bias.

**Theorem 2: Causally Consistent Turn-Level Advantage Estimation**

Consider turn $k$ and suppose that the $G-1$ sibling rollouts are forked from the same boundary state $S_k$, with each sibling rollout $j \in \{1, \ldots, G-1\}$ independently sampling a turn-$k$ macro-action $u_{j,k} \equiv l_{j,k}$. Let $A^H_{j,k}$ denote the turn-level advantage assigned to sibling rollout $j$. Then, under bounded rewards and the sampling assumptions in *Appendix C.3*, the turn-level advantage estimator satisfies:

(a) **Local unbiasedness up to finite-group bias.** Conditional on $(S_k, u_{j,k})$, the expectation of $A^H_{j,k}$ is proportional to the true turn-level advantage, up to a finite-group bias of order $O(1/G)$. Because all comparisons are made from the same boundary state, the estimator removes the upstream state-contamination term that appears in cross-trajectory comparisons.

(b) **Reduced cross-state estimation error.** The estimator $A^H_{j,k}$ removes the cross-state value contribution to the MSE of trajectory advantage estimation. This contribution scales with the variance of values across boundary states, which can be large in long-horizon, multi-turn tasks.

(c) **State-matched causal actions.** Since all siblings share the same boundary state $S_k$, differences in their conditional expected returns are attributable to the sampled turn-$k$ macro-actions $u_{j,k}$, rather than to upstream trajectory differences. The resulting advantage is assigned only to turn-$k$ output tokens, excluding prefix tokens from the direct loss.

The proof of Theorem 2 is provided in *Appendix C.3*.

### 3.3 On-Policy Continuation

**On-policy evolution.** Asynchronous turn updates can induce policy drift that is not fully corrected by per-token IS clipping. In principle, one could correct this drift using the full trajectory-level IS product, but its variance grows exponentially with the horizon length, making it unstable for long-horizon multi-turn training. This drift directly affects the downstream continuation value $F_k$ introduced in *Section 3.2*: under a stale or mismatched downstream policy, $\hat{F}_{j,k}$ can systematically deviate from the true continuation value $F_k^{\pi_{\theta_{>k}}}$, while existing IS-based corrections are insufficient to remove this deviation. To avoid this issue, RTPO re-generates sibling continuations on-policy at each turn, so that their returns estimate the current continuation value conditional on the retained boundary state. Let $\theta_0$ denote the parameters at the start of the policy evolution. During reverse-order training, the parameters are updated sequentially across turns. By the time optimization reaches turn $k$, the policies for downstream turns $K{-}1, \ldots, k{+}1$ have already been updated; we denote the resulting current parameters by $\theta_{>k}$. Thus, $\pi_{\theta_{>k}}$ incorporates downstream-turn updates and provides the current continuation policy; turn $k$ and upstream turns have not yet received their own stage updates. Note that the trunk is a complete trajectory generated at the start of the policy evolution using $\pi_{\theta_0}$. It does not participate in gradient updates, nor does it enter the computation of $\hat{V}_k$. Its sole role is to provide boundary states $S_k$ and environment snapshots $\mathrm{snap}_k$ as anchoring points for sibling forking. The trunk's policy nature affects which states $S_k$ are visited during training (state coverage), but does not affect the correctness of advantage estimation, since all siblings contributing to the estimate are generated on-policy.

**On-policy sibling generation.** At the start of turn $k$, RTPO synchronizes the latest parameters $\theta_{>k}$ to the inference engine, forks the environment from $\mathrm{snap}_k$, and generates $G{-}1$ siblings using $\pi_{\theta_{>k}}$. Each sibling rollout $j$ generates a turn-$k$ response, then continues with $\pi_{\theta_{>k}}$ through turns $k{+}1$ to $K{-}1$, obtaining terminal reward $R_j$. Since the sampling policy equals the current policy, the trajectory-level IS weight is identically one: $\omega_j = \prod_{h=k+1}^{K-1} \prod_{t=1}^{T_{j,h}} \frac{\pi_{\theta_{>k}}(a_{j,h,t}|s_{j,h,t})}{\pi_{\theta_{>k}}(a_{j,h,t}|s_{j,h,t})} \equiv 1$.

Since turn-level value estimation is $\hat{Q}_{j,k} = r_{j,k} + \gamma^{\tau_k} \hat{F}_{j,k}$ from *Section 3.2*, the sibling's downstream continuation is performed under the current policy $\pi_{\theta_{>k}}$, the sample $\hat{F}_{j,k}$ is an unbiased draw from $F_k^{\pi_{\theta_{>k}}}(S_{j,k+1})$. Therefore:

$$\hat{Q}_{j,k} = r_{j,k} + \gamma^{\tau_k} \hat{F}_{j,k}^{\pi_{\theta_{>k}}} \tag{8}$$

is an unbiased Monte Carlo estimate of $\tilde{Q}_k^{\pi_{\theta_{>k}}}(S_k, u_{j,k})$. The entire value estimate requires no IS correction, is unaffected by clip truncation, and is free of multiplicative variance explosion.

In **Theorem 3**, we show that RTPO conducts on-policy continuation to avoid policy drift and further reduces advantage-estimation errors.

**Theorem 3: On-Policy Continuation under Asynchronous Turns**

In the reverse-order training procedure of RTPO, at the start of each asynchronous turn $k$, sibling rollouts $j \in \{1, \ldots, G{-}1\}$ are generated from the shared boundary state $S_k$ using the current downstream policy $\pi_{\theta_{>k}}$ and are then continued to termination under the same policy. Then the following two properties hold:

(a) **Drift-free on-policy continuation.** Each sibling's terminal return $R_j$ provides an unbiased Monte Carlo estimate of the turn-level Q-value under the current downstream policy $\pi_{\theta_{>k}}$. Because the sampling policy and the evaluation policy coincide throughout the sibling continuation, the trajectory-level importance-sampling weight $\omega_j$ is identically one, and no trajectory-level IS correction is required.

(b) **Dynamic error reduction in advantage estimation.** As reverse-order training improves the downstream policies, the continuation value estimates $\hat{F}_{j,k}^{\pi_{\theta_{>k}}}$ become aligned with the current downstream policy rather than a stale policy. Under bounded binary or normalized rewards, when the induced success probabilities move away from the high-uncertainty region around one-half, the variance of the Monte Carlo Q-value estimator decreases, thereby improving the signal-to-noise ratio of the resulting turn-level advantage estimates $A_{j,k}^H$.

The proof of Theorem 3 is provided in *Appendix C.4*.

# 4 Experimental Results and Analysis

**Experimental Setting.** We use Qwen3-8B (Yang et al., 2025a) as the backbone for the main experiments on RTPO and all baselines, enabling Qwen3's thinking mode for all multi-turn agentic RL rollouts. We compare RTPO with trajectory-level methods, including GRPO (Shao et al., 2024), and turn/tree-level credit methods, including ARPO (Dong et al., 2026), TreeGRPO (Ji et al., 2026), and SeeUPO (Hu et al., 2026). We consider 12 experiments covering comprehensive mathematical and knowledge reasoning benchmarks, where agents perform multi-turn RL training with external calculation and web-search tools. All methods are implemented in VeRL (Sheng et al., 2025), with vLLM (Kwon et al., 2023) for rollout generation and FSDP (Zhao et al., 2023) for distributed training. We evaluate training performance and stability using task accuracy, tool-call statistics, log-probability comparisons, turn-level credit effects, and the hit rates of on-policy versus off-policy outputs. Detailed experimental setup, including base models, baselines, datasets, configurations, and evaluation metrics, is provided in *Appendix D*. Implementation details, including pseudocode, source code, training details, and the Qwen3 chat template, are provided in *Appendix E*. *Appendix G* further shows that RTPO attains the best held-out Pass@1 and Pass@4 on the stateful agentic benchmark (Barres et al., 2026), and analyzes the group size $G$ and compute-matched training cost.

## 4.1 Main Results

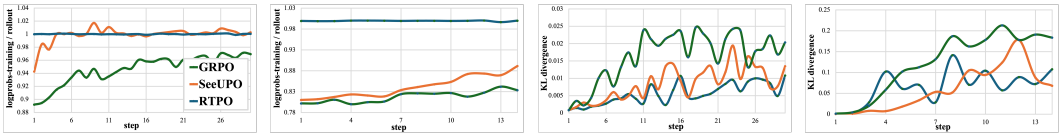
We compare RTPO against the trajectory-level method GRPO and the turn-level method SeeUPO, and additionally include the untuned vanilla model as a non-RL reference. **Table 1** reports both accuracy and the corresponding number of tool calls on each benchmark. We evaluate overall performance on mathematical reasoning and knowledge-intensive question answering across three difficulty tiers: easy (GSM8K), medium (AMC23, MATH500), and hard (AIME24, AIME25, OE-Math, HotpotQA, and 2Wiki). Our RTPO achieves the best performance across all eight benchmarks, improving overall accuracy over vanilla by 66.78%, outperforming GRPO (+21.50%) and SeeUPO (+10.76%). Further discussion and insights on tool calls are provided in *Appendix F.1*.

Table 1: **Performance comparison across 8 tool-use agentic RL benchmarks.** *Acc* denotes Pass@1 for mathematical tasks and best-span F1 for knowledge tasks, while *Calls* denotes the number of tool calls rounded to the nearest integer, such as Python and web-search calls. *Overall* subscripts indicate relative changes in accuracy compared with the vanilla model, along with total tool calls (M, K).

| Method | M: Mathematical Reasoning | | | | | | | | | | | | K: Knowledge Reasoning | | | | Overall | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AIME24 | | AIME25 | | AMC23 | | MATH500 | | GSM8K | | OE-Math | | HotpotQA | | 2Wiki | | | |
| | Acc | Calls | Acc | Calls | Acc | Calls | Acc | Calls | Acc | Calls | Acc | Calls | Acc | Calls | Acc | Calls | Acc | Calls |
| Vanilla | 3.33 | 17 | 13.33 | 23 | 12.50 | 66 | 51.00 | 341 | 76.95 | 731 | 24.33 | 453 | 54.52 | 204 | 59.83 | 224 | $36.97_{+00.00\%}$ | $2059_{(1631,428)}$ |
| GRPO | 10.00 | 12 | 20.00 | 8 | 57.50 | 18 | 82.33 | 94 | 93.86 | 59 | 50.74 | 29 | 53.39 | 214 | 61.85 | 226 | $53.71_{+45.28\%}$ | $660_{(220,440)}$ |
| SeeUPO | 20.00 | 8 | 23.33 | 6 | 67.50 | 14 | 84.20 | 87 | 94.47 | 87 | 53.86 | 147 | 54.27 | 254 | 63.78 | 253 | $57.68_{+56.02\%}$ | $856_{(349,507)}$ |
| RTPO | **33.33** | 4 | **26.67** | 2 | **67.50** | 9 | **86.20** | 168 | **94.84** | 482 | **55.49** | 106 | **64.89** | 613 | **64.35** | 867 | $\mathbf{61.66}_{+66.78\%}$ | $2251_{(771,1480)}$ |

## 4.2 Rollout–Training Consistency Analysis

To examine RTPO's training-time advantage, we measure at each step the geometric-mean ratio between training-stage and rollout-stage token *logprobs*. This ratio captures rollout–training consistency: values near 1 indicate matched conditioning distributions, while deviations suggest that the full-history training policy favors outputs different from those sampled during rollout. The Kullback–Leibler (KL) divergence further quantifies the distributional gap, with smaller values indicating stronger rollout–training consistency. **Figure 3** shows the log-probability ratio and KL divergence during training. On mathematical tasks, RTPO stays stable at 1.0, while SeeUPO fluctuates around 1.0 and GRPO recovers from 0.89 to 0.97 after 30 steps. On knowledge tasks, RTPO again remains at 1.0, whereas SeeUPO and GRPO recover only from around 0.80 to 0.89 and 0.83 after 14 steps.




(a) Rollout-train ratio [M] (b) Rollout-train ratio [K] (c) KL divergence (M) (d) KL divergence (K)

Figure 3: **Rollout–training consistency comparison** using log-probability ratios and KL divergence on mathematical (M) and knowledge (K) reasoning tasks. Zoom in for better visualization.

RTPO also achieves the lowest average KL divergence, indicating more consistent rollout-training contexts than the baselines. A noteworthy observation is that, despite using different rollout and training contexts, baseline ratios still drift slowly toward 1. This resembles the bootstrapped alignment mechanisms in DAgger (Ross et al., 2011) and SCoRe (Kumar et al., 2025). However, unlike RTPO's structural consistency, this empirical alignment is incomplete, task-dependent, noisy, and consumes additional optimization budget. We provide further discussion and insights in *Appendix F.2*.

### 4.3 Effect of Turn-Level Credit Assignment

To isolate the effect of RTPO's turn-level credit signal, we feed the full interaction history as the rollout input for all methods, controlling for the rollout–training context mismatch presented in *Section 4.2*. Under this control setting, performance differences mainly reflect the effect of credit assignment (CA) on advantage estimation. Because full interaction histories are used, the accuracy scores in Table 2 are substantially higher than those in Table 1. RTPO-CA achieves the highest overall average across the four mathematical reasoning benchmarks, as shown in **Table 2**. It achieves the best results on AMC23 (Pass@1 93.33, Pass@4 100.0), AIME25 (Pass@1 70.00, Pass@4 76.67), and MATH500 (Pass@1 86.60, Pass@4 89.60), while matching the best Pass@4 on AIME24 (80.00). These results show that RTPO improves both solution coverage and the preference for correct answers, with the largest gain on the harder AIME25 benchmark. This supports the value of turn-level credit assignment: by forking sibling rollouts from the same boundary state $S_k$, RTPO forms a local baseline and attributes advantage directly to the current turn decision.

Table 2: **Mathematical reasoning performance under controlled turn-level credit assignment (CA)**. Overall reports the average across benchmarks. Bold indicates the best result.

| Method | *AMC23* | | *AIME24* | | *AIME25* | | *MATH500* | | *Overall* | |
|---|---|---|---|---|---|---|---|---|---|---|
| | P@1 | P@4 | P@1 | P@4 | P@1 | P@4 | P@1 | P@4 | P@1 | P@4 |
| ARPO | 90.00 | 97.50 | **73.33** | 73.33 | 63.33 | 73.33 | 86.00 | 88.80 | 78.15 | 83.23 |
| TreeGRPO | 92.50 | 96.67 | 63.33 | **80.00** | 56.67 | 70.00 | 86.20 | 89.40 | 74.67 | 84.00 |
| SeeUPO | 92.50 | 97.50 | **73.33** | **80.00** | 53.33 | 70.00 | 86.20 | 89.20 | 76.33 | 84.18 |
| RTPO-CA | **93.33** | **100.0** | 66.67 | **80.00** | **70.00** | **76.67** | **86.60** | **89.60** | **79.13** | **86.55** |

### 4.4 Policy Drift Correction

To isolate the effect of on-policy continuation (*Section 3.3*), we compare the default RTPO with an off-policy variant that reuses downstream continuations generated by $\pi_{\theta_0}$ during the initial rollout and corrects staleness using a clamped trajectory-level IS weight; see *Appendix E.2* for details. We evaluate both variants on four knowledge-reasoning deep-search benchmarks (GAIA, WebWalkerQA, HLE, and XBench) using output-hit accuracy. As shown in **Table 3**, default RTPO outperforms the off-policy variant on GAIA ($+5.83\%$), WebWalkerQA ($+3.50\%$), and XBench ($+7.00\%$). On HLE, the difference is negligible ($-0.33\%$), indicating a near tie. This pattern supports *Theorem 3(b)*: on-policy continuation is most beneficial when downstream policies change substantially during reverse-order training. GAIA, WebWalkerQA, and XBench involve longer retrieval and interaction horizons, where stale-rollout IS correction can introduce clamp-truncation bias that on-policy re-sampling avoids. In contrast, HLE is more closed-ended and often requires shorter search horizons, leading to smaller gains; additional results in *Appendix F.3* further support this interpretation.

Table 3: **Output-hit comparison between on- and off-policy RTPO variants on knowledge tasks.** Values are rounded to the nearest integer. $\Delta$ denotes the change from off-policy to on-policy hit rates.

| **Benchmark** | QS | Off-Policy Hit | Off-Policy Rate | On-Policy Hit | On-Policy Rate | $\Delta$ |
|---|---|---|---|---|---|---|
| *GAIA* | 103 | 24 | 23.30% | 30 | 29.13% | **+5.83%** |
| *XBench* | 100 | 8 | 8.00% | 15 | 15.00% | **+7.00%** |
| *WebWalker* | 200 | 12 | 6.00% | 19 | 9.50% | **+3.50%** |
| *HLE* | 2096 | 261 | 12.45% | 254 | 12.12% | -0.33% |

## 5 Concluding Remarks

This work identifies rollout–training mismatch as a key source of instability in multi-turn agentic RL. We theoretically analyze its effects and propose RTPO, which unifies rollout–training consistency, turn-level credit assignment, and on-policy continuation. Theoretical and empirical results demonstrate improved policy optimization stability, providing a principled framework for training long-horizon tool-using agents.

### AI USE STATEMENT

Generative AI tools were used solely for language editing, LaTeX debugging, and assistance with research code implementation and refinement. All AI-assisted text and references were manually reviewed by the authors, all AI-assisted code was tested, and the reported results were independently verified. The authors take full responsibility for the accuracy, integrity, and final content of this work.

### ETHICS STATEMENT

This work uses publicly available models, benchmarks, and software and does not involve human subjects, personal information, or sensitive data. We comply with the applicable licenses and terms of use for all models, datasets, and software used in this study.

### REPRODUCIBILITY STATEMENT

The Appendix includes the complete methodological formulation and theoretical analysis, including assumptions and proofs, as well as details of the experimental setup, training configuration, evaluation protocol, and implementation. Source code required to reproduce the experiments is provided as supplementary material.

## REFERENCES

David Abel, Nate Umbanhowar, Khimya Khetarpal, Dilip Arumugam, Doina Precup, and Michael Littman. Value preserving state-action abstractions. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, 2020. URL https://proceedings.mlr.press/v108/abel20a.html.

Cameron Allen, Neev Parikh, Omer Gottesman, and George Konidaris. Learning markov state abstractions for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/454cecc4829279e64d624cd8a8c9ddf1-Abstract.html.

J. Andrew Bagnell, Sham M. Kakade, Jeff G. Schneider, and Andrew Y. Ng. Policy search by dynamic programming. In *Advances in Neural Information Processing Systems*, volume 16, pp. 831–838, 2003. URL https://proceedings.neurips.cc/paper_files/paper/2003/hash/3837a451cd0abc5ce4069304c5442c87-Abstract.html.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. $\tau^2$-bench: Evaluating conversational agents in a dual-control environment. In *International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/64377.

Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 2012. URL https://doi.org/10.1109/TCIAIG.2012.2186810.

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

Ruike Cao, Shaojie Bai, Fugen Yao, Liang Dong, Jian Xu, and Li Xiao. Atpo: Adaptive tree policy optimization for multi-turn medical dialogue. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/34b3a40ec9752c1ae48fe85fef8fe8dc-Abstract-Conference.html.

Center for AI Safety, Scale AI, and HLE Contributors Consortium. A benchmark of expert-level academic questions to assess AI capabilities. *Nature*, 649:1139–1146, 2026. doi: 10.1038/s41586-025-09962-4. URL https://www.nature.com/articles/s41586-025-09962-4.

Qikai Chang, Zhenrong Zhang, Pengfei Hu, Jun Du, Jiefeng Ma, Yicheng Pan, Jianshu Zhang, Quan Liu, and Jianqing Gao. THOR: Tool-integrated hierarchical optimization via RL for mathematical reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/245b7f8357fdc26797ed6714cda5bd94-Abstract-Conference.html.

Kaiyuan Chen, Yixin Ren, Yang Liu, Xiaobo Hu, Haotong Tian, Tianbao Xie, Fangfu Liu, Haoye Zhang, Hongzhang Liu, Yuan Gong, et al. xbench: Tracking agents productivity scaling with profession-aligned real-world evaluations. *arXiv preprint arXiv:2506.13651*, 2025. URL https://arxiv.org/abs/2506.13651.

Yifei Chen, Guanting Dong, and Zhicheng Dou. Toward effective tool-integrated reasoning via self-evolved preference learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/788f0d336eeddf698c8d527b1794fca4-Abstract-Conference.html.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *Proceedings of the 5th International Conference on Computers and Games*, 2006. URL https://doi.org/10.1007/978-3-540-75538-8_7.

Thomas G Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of artificial intelligence research*, 2000. URL https://doi.org/10.1613/JAIR.639.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic reinforced policy optimization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/1c58e53bdf1fb045440256fd567531ae-Abstract-Conference.html.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International conference on machine learning*. PMLR, 2023. URL https://proceedings.mlr.press/v202/gao23f.html.

Zhaolin Gao, Wenhao Zhan, Jonathan Chang, Gokul Swamy, Kianté Brantley, Jason Lee, and Wen Sun. Regressing the relative future: Efficient policy optimization for multi-turn rlhf. In *International Conference on Learning Representations*, volume 2025, pp. 8475–8514, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/186094ef879dcbd5ef6879782916c309-Abstract-Conference.html.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://www.nature.com/articles/s41586-025-09422-z.

Milos Hauskrecht, Nicolas Meuleau, Leslie Pack Kaelbling, Thomas Dean, and Craig Boutilier. Hierarchical solution of markov decision processes using macro-actions. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 1998. URL https://research.google/pubs/hierarchical-solution-of-markov-decision-processes-using-macro-actions/.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850. Association for Computational Linguistics,

2024. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *Thirty-Fifth Annual Conference on Neural Information Processing Systems*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, 2020. URL https://aclanthology.org/2020.coling-main.580/.

Tianyi Hu, Qingxu Fu, Yanxi Chen, Zhaoyang Liu, and Bolin Ding. Seeupo: Sequence-level agentic-rl with convergence guarantees. *arXiv preprint arXiv:2602.06554*, 2026. URL https://arxiv.org/abs/2602.06554.

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for LLM agent reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/8c7304e77c832ddc70075dfee081ca6c-Abstract-Conference.html.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, JD Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pp. 54523–54549, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/871ac99fdc5282d0301934d23945ebaa-Abstract-Conference.html.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023. URL https://doi.org/10.1145/3600006.3613165.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *Advances in Neural Information Processing Systems*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/8636419dea1aa9fbd25fc4248e702da4-Abstract-Conference.html.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, et al. Websailor: Navigating super-human reasoning for web agent. *arXiv preprint arXiv:2507.02592*, 2025. URL https://arxiv.org/abs/2507.02592.

Yugu Li, Zehong Cao, Jianglin Qiao, and Siyi Hu. SSVPO: Effective step-level credit assignment for RL training of language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/4da4dd613cc9e63932d643b9c6e8f86f-Abstract-Conference.html.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Tobias Lindenbauer, Igor Slinko, Ludwig Felder, Egor Bogomolov, and Yaroslav Zharov. The complexity trap: Simple observation masking is as efficient as llm summarization for agent context management. *arXiv preprint arXiv:2508.21433*, 2025. URL https://arxiv.org/abs/2508.21433.

Mugeng Liu, Xiaojun Ma, Yuhang Xie, Qin Chen, Xuanzhe Liu, and Yun Ma. ROGA: Scaling generalist agents for office productivity tasks via tool generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/pdf?id=KTyLxtODB9.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Miao Lu, Weiwei Sun, Weihua Du, Zhan Ling, Xuesong Yao, Kang Liu, and Jiecao Chen. Beyond the context window: Scaling agentic RL via end-to-end optimized context compression. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21074–21125, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.966. URL https://aclanthology.org/2026.acl-long.966/.

Mathematical Association of America. American mathematics competitions (AMC), 2023. URL https://maa.org/student-programs/amc/.

Mathematical Association of America. American invitational mathematics examination (AIME), 2024. URL https://maa.org/maa-invitational-competitions/.

Mathematical Association of America. American invitational mathematics examination (AIME), 2025. URL https://maa.org/maa-invitational-competitions/.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/25ae35b5b1738d80f1f03a8713e405ec-Abstract-Conference.html.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741. Curran Associates, Inc., 2023. doi: 10.52202/075280-2338. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

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

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems*. Association for Computing Machinery, 2025. doi: 10.1145/3689031.3696075. URL https://doi.org/10.1145/3689031.3696075.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 2023. URL https://papers.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Parshin Shojaee, Aneesh Jain, Sindhu Tipirneni, and Chandan K. Reddy. Execution-based code generation using deep reinforcement learning. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=0XBuaxqEcG.

David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016. URL https://doi.org/10.1038/nature16961.

Shuang Sun, Huatong Song, Yuhao Wang, Ruiyang Ren, Jinhao Jiang, Junjie Zhang, Fei Bai, Jia Deng, Wayne Xin Zhao, Zheng Liu, Lei Fang, Zhongyuan Wang, and Ji-Rong Wen. SimpleDeepSearcher: Deep information seeking via web-powered reasoning trajectory synthesis. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 13705–13720. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-emnlp.739. URL https://aclanthology.org/2025.findings-emnlp.739/.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025. URL https://arxiv.org/abs/2501.12599.

Wanxin Tian, Shijie Zhang, Kevin Zhang, Xiaowei Chi, Chun-Kai Fan, Junyu Lu, Yulin Luo, Qiang Zhou, Yiming Zhao, Ning Liu, Siyu Lin, Zhiyuan Qin, Xiaozhu Ju, Shanghang Zhang, and Jian Tang. SEEA-r1: Tree-structured reinforcement fine-tuning for self-evolving embodied agents. In *Advances in Neural Information Processing Systems*, volume 38, pp. 78458–78499. Curran Associates, Inc., 2025. doi: 10.52202/085713-2628. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/7103cd82de95a7b30983fcf74ba499ac-Abstract-Conference.html.

Xingyao Wang, Simon Rosenberg, Juan Michelini, Calvin Smith, Hoang H. Tran, Engel Nyst, Rohit Malhotra, Xuhui Zhou, Valerie Chen, Robert Brennan, and Graham Neubig. The OpenHands software agent SDK: A composable and extensible foundation for production agents. In A. Chowdhery and Z. Jia (eds.), *Proceedings of Machine Learning and Systems*, volume 8, pp. 1843–1855. MLSys, 2026. URL https://proceedings.mlsys.org/paper_files/paper/2026/hash/8ae9cf363ea625161f885b798c1f1f78-Abstract-Conference.html.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025. URL https://arxiv.org/abs/2504.20073.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, et al. Webwalker: Benchmarking llms in web traversal. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025a. URL https://aclanthology.org/2025.acl-long.508/.

Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Xinmiao Yu, Dingchu Zhang, Yong Jiang, et al. Resum: Unlocking long-horizon search intelligence via context summarization. *arXiv preprint arXiv:2509.13313*, 2025b. URL https://arxiv.org/abs/2509.13313.

Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun MA, and Bo An. SimpleTIR: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 8424–8449, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/0e769ec2c2cd99b6ad69c9d75113e386-Abstract-Conference.html.

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a. URL https://arxiv.org/abs/2505.09388.

Zhicheng Yang, Zhijiang Guo, Yinya Huang, Xiaodan Liang, Yiwei Wang, and Jing Tang. Treerpo: Tree relative policy optimization. *arXiv preprint arXiv:2506.05183*, 2025b. URL https://arxiv.org/abs/2506.05183.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018. URL https://aclanthology.org/D18-1259/.

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*, 2023. URL https://react-lm.github.io/.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. MemAgent: Reshaping long-context LLM with multi-conv RL-based memory agent. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/4264ee4376776907c0b87ed70b959585-Abstract-Conference.html.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3775. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a4277440d50f1f15d2cb4c14f7e0c0d2-Abstract-Conference.html.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025. URL https://arxiv.org/abs/2504.05118.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. PyTorch FSDP: Experiences on scaling fully sharded data parallel. *Proceedings of the VLDB Endowment*, 16(12):3848–3860, 2023. doi: 10.14778/3611540.3611569. URL https://www.vldb.org/pvldb/vol16/p3848-huang.pdf.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025. URL https://arxiv.org/abs/2507.18071.

Yifan Zhong, Jakub Grudzien Kuba, Xidong Feng, Siyi Hu, Jiaming Ji, and Yaodong Yang. Heterogeneous-agent reinforcement learning. *Journal of Machine Learning Research*, 2024. URL https://www.jmlr.org/papers/volume25/23-0488/23-0488.pdf.

# A Related Work

**Policy optimization in agentic RL.** Recent post-training of large language models (LLMs) has shifted from supervised fine-tuning toward reinforcement learning with verifiable rewards (RLVR). Among existing approaches, GRPO (Shao et al., 2024), built upon PPO, reduces variance through token-level importance ratios and group-relative advantage estimation, and has become a representative algorithm for agentic RL. Its sequence-level refinement, GSPO (Zheng et al., 2025), further defines importance ratios and clipping operations at the sequence level, leading to more stable training dynamics on models such as Qwen3. However, when these RL methods are directly transferred from single-turn to multi-turn agentic settings, a structural mismatch emerges between rollout and training. Existing methods often treat the entire multi-turn interaction as a single concatenated trajectory and distribute a single scalar reward uniformly across all tokens, thereby ignoring the actual contribution of each turn. More importantly, the rollout may operate on truncated or summarized contexts, whereas the training recomputes importance ratios over the full interaction history. This discrepancy induces a mismatch in conditioning distributions and weakens the assumptions under which PPO-style optimization is expected to remain stable. In long-horizon multi-turn scenarios, these issues can manifest as training divergence or eventual policy collapse.

A noteworthy latest work is SeeUPO (Hu et al., 2026), which first models multi-turn interaction as a sequentially executed multi-agent bandit problem. Under the heterogeneous-agent RL framework (Zhong et al., 2024), SeeUPO updates turn-level virtual agents in reverse execution order ($T \to T-1 \to \cdots \to 1$), thereby inheriting the monotonic-improvement property and proving convergence to the globally optimal policy. This provides strong motivation for our proposed RTPO design, particularly its reverse-order turn updates. However, each turn-level agent in SeeUPO is still trained on the full chat history rather than on the turn-level conditioning context $c_k$ actually observed by the model during rollout; therefore, the policy-forward mismatch is not eliminated. Moreover, within each turn, SeeUPO corrects downstream advantages using token-level importance-sampling reweighting. As the reverse training recursion proceeds and the number of involved turns accumulates, the variance of the importance-sampling ratio can grow multiplicatively. Although clipping bounds this ratio from above, it also introduces systematic bias into the per-turn advantage estimate.

**Tree-based credit assignment.** To mitigate the sparse-credit problem caused by flattened trajectories in multi-turn agentic RL training, a recent line of work reorganizes rollouts into tree structures with shared prefixes, thereby constructing finer-grained credit signals. ARPO (Dong et al., 2026) adaptively tree branches at high-entropy nodes following tool calls and, through advantage attribution estimation, applies branch averaging to shared-prefix tokens while estimating advantages independently for tokens on disjoint branches. Afterward, TreeGRPO (Ji et al., 2026) abstracts each turn as a tree node and combines intra-tree and inter-tree group-relative advantages, theoretically establishing gradient-level equivalence with step-level DPO (Rafailov et al., 2023). This tree-based formulation is extended in SEEA-R1 (Tian et al., 2025), which integrates MCTS (Coulom, 2006; Browne et al., 2012; Silver et al., 2016) into embodied-agent settings and trains a multimodal generative reward model to densify sparse outcome rewards. Tree-structured credit assignment has also been explored in step/token levels. TreeRPO (Yang et al., 2025b) adopts an $N$-ary tree for mathematical reasoning and constructs step-level rewards through bottom-up Bellman expectations. TreeAdv (Cao et al., 2026a) employs entropy-triggered branching and redistributes leaf advantages to tokens using inverse-descendant-count weighting. Similarly, ATPO (Cao et al., 2026b) operates under a hierarchical MDP and uses Bellman error and $Q$-value variance as uncertainty measures for adaptive expansion, while applying visit-count-based down-weighting to suppress update imbalance caused by repeated nodes.

Although these tree-based rollouts improve the granularity of credit assignment in multi-turn training, they do not fully eliminate the propagation of trajectory-level credit mismatch to individual turns through the reconstructed tree. For instance, in Tree-GRPO and SEEA-R1, advantages are still derived from leaf returns, typically in the form $A_i = \frac{R_i - \mathrm{mean}(R)}{\mathrm{std}(R)}$. As a result, prefix nodes that appear on multiple paths inherit trajectory-level signals from all descendant outcomes, causing the gradient direction at shared tokens to be perturbed by trajectory-level scalars originating from different rollouts. In TreeAdv, the $1/|S|$ normalization attenuates the signal more aggressively near the root, precisely suppressing early decisions where informative gradients are often most needed. In addition, ARPO averages multi-branch advantages on shared tokens, which can dilute the signal linearly with the number of branches, while ATPO relies on a learned critic and is therefore exposed to critic-induced

bias. Therefore, under the rollout–training contextual mismatch settings, shared-prefix tokens may still carry trajectory-level credit signals with significant residual bias.

**Policy gradient correction.** Another line of work studies stable training from the perspective of correcting (off-)policy drift through interventions. SimpleTIR (Xue et al., 2026), by decomposing the policy gradient on softmax logits, attributes gradient explosion in multi-turn tool-integrated reasoning (TIR) to the accumulation of low-probability tokens under the distributional shift induced by tool feedback. It proposes filtering out entire trajectories that contain *void turns*: turns produce neither a complete code block nor a final answer to block harmful gradients. This method is empirically effective and plug-and-play; however, the void-turn criterion is tightly coupled with the code-execution setting of mathematical reasoning and is difficult to transfer to other scenarios, such as web search. It also inevitably discards a non-trivial fraction of training data. RAGEN (Wang et al., 2025) discovers the inconsistency between rollout and training engines for mismatch correction through numerical mechanisms such as truncated importance sampling. However, these approaches mainly address numerical discrepancies while overlooking the contextual mismatch between rollout conditioning and training recomputation. In contrast, our proposed RTPO does not rely on trajectory dropping or post-hoc numerical correction. Instead, it restructures the training paradigm from a turn-boundary MDP perspective, maintaining the on-policy optimization throughout the entire training process.

## B Full Theoretical Analysis: Training Instability

Here, we provide the full theoretical analysis of training instability in multi-turn agentic RL: **Single flattened-trajectory policy optimization (B.1):** Standard rollouts are typically generated under truncated or summarized contexts for efficiency, whereas training re-evaluates tokens under concatenated full-history contexts without truncation. This rollout–training context mismatch induces biased importance-sampling (IS) ratios, thereby undermining stable policy optimization. **Trajectory-only credit assignment (B.2):** In multi-turn interactions, a given state may admit multiple valid actions, requiring accurate credit assignment for each action. However, credit computed from a single trajectory-level advantage can obscure the contribution of individual turns and therefore cannot provide a proper comparison among alternative actions. **Long-horizon off-policy drift (B.3):** In long-horizon tasks, standard policy optimization with per-token clipping is insufficient to correct policy drift under asynchronous training. Since later states depend on earlier generated actions and tool feedback, once the policy drifts, the later-turn states visited during rollout may no longer match those induced by the current policy.

**Preliminaries.** A multi-turn interaction with TIR is represented as a standard trajectory $(q, l_0, f_0, \ldots, l_{n-1}, f_{n-1})$ spanning $n$ turns, where $q$ is the initial prompt, $l_k$ is the response generated, and $f_k$ is the corresponding external tool feedback returned by the environment for each turn $k \in \{0, \ldots, n-1\}$. We formulate this process as a *Hierarchical Markov Decision Process (H-MDP)*, which captures turn-level (high-level) planning and token-level (low-level) execution in multi-turn agentic RL. At the turn level, for each turn $k$, the agent state $S_k$ represents all interaction history $(q, l_0, f_0, \ldots, l_{k-1}, f_{k-1})$, available before the current turn, including previous responses, tool calls, and corresponding environment feedback. The corresponding turn-level action is the decision of what response to produce at the current turn, denoted by $l_k \in \mathcal{A}_H$. After executing this action and receiving environmental feedback $f_k$, the turn-level state evolves as $S_{k+1} = S_k \circ (l_k, f_k)$, where $\circ$ denotes concatenation.

Each turn-level action (response) $l_k$ is generated autoregressively at the token level as $l_k = (a_{k,1}, a_{k,2}, \ldots, a_{k,T_k})$, where $a_{k,t} \in \mathcal{A}_L$ denotes the token generated at step $t$, and $T_k$ is the number of tokens in the response at turn $k$. The corresponding low-level state is $s_{k,t} = (S_k, a_{k,1}, \ldots, a_{k,t-1}) \in \mathcal{S}_L$, which consists of the turn history $S_k$ together with the token prefix generated so far in the current turn. Since the token only serves to generate the turn-level action (response) and does not itself receive intermediate reward, we set the low-level reward to $R_L = 0$ and set $\gamma_H = \gamma_L = 1$ as the discount factor.

Our hierarchical formulation differs from existing policy optimization methods, which do not explicitly distinguish turn-level planning from token-level execution. Instead, they flatten the entire multi-turn interaction into a single token sequence, i.e., a single trajectory, and optimize it using clipped policy optimization methods in the PPO family. Let $q \sim \mathcal{D}$ be an input prompt sampled from

the task distribution, and let $\{g_i\}_{i=1}^{G}$ be a group of $G$ trajectories of this form sampled from the old policy $\pi_{\theta_{\text{old}}}$ conditioned on $q$, where $\theta$ denotes the current policy parameters and $\theta_{\text{old}}$ denotes the rollout policy parameters. For trajectory $g_i$, we denote its turn-$k$ state and response by $S_{i,k}$ and $l_{i,k}$, respectively:

$$J^{\text{flat}}(\theta) = \mathbb{E}_{\substack{q\sim\mathcal{D} \\ \{g_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)}} \left[ \frac{1}{G}\sum_{i=1}^{G} \frac{1}{\sum_{t'} m_{i,t'}} \sum_{t} m_{i,t} L_{i,t}^{\text{CLIP}}(\theta) \right] \tag{9}$$

where $i$ indexes the sampled trajectory, $t$ indexes flattened token positions, and $L_{i,t}^{\text{CLIP}}(\theta) = \min(\rho_{i,t}(\theta)A_i,\ \text{clip}(\rho_{i,t}(\theta),\ 1-\epsilon,\ 1+\epsilon)\, A_i)$ is the standard clipped surrogate objective with clipping threshold $\epsilon$. $a_{i,t}$ is the token generated at flattened position $t$ in the $i$-th sampled trajectory, $m_{i,t}$ is a binary mask indicating whether that token contributes to the policy gradient, and $\rho_{i,t}(\theta) = \frac{\pi_\theta(a_{i,t}|x_{i,<t})}{\pi_{\theta_{\text{old}}}(a_{i,t}|x_{i,<t})}$ is the IS ratio. The context $x_{i,<t}$ is the flattened interaction context preceding token $a_{i,t}$, including previously generated response tokens and inserted environment feedback. The scalar $A_i$ denotes the trajectory-level group-relative advantage, which is uniformly assigned to all unmasked tokens in trajectory $g_i$.

### B.1 Single flattened-trajectory policy optimization: mismatch from rollout to training

In multi-turn interactions, the accumulated history can become too long for the model to process in full. For example, an agent may search for information, call a tool, revise its plan based on the returned result, and repeat this process over many turns. By later turns, the prompt, previous responses, and tool feedback may already span tens of thousands of tokens. In practice, rollouts therefore often rely on a truncated or summarized context rather than the complete interaction history (Wang et al., 2026; Lindenbauer et al., 2025; Wu et al., 2025b). Therefore, existing methods typically reconstruct the whole interaction as a single concatenated sequence, or flattened trajectory, during training. This flattened training formulation evaluates each generated token under the concatenated prefix, rather than under the original context that was actually used during rollout. Consequently, tokens generated at turn $k$ are optimized under a conditioning context that can differ from the rollout context that produced them. We refer to this discrepancy as a rollout-to-training mismatch induced by flattened-trajectory policy optimization. We next analyze how this mismatch distorts the likelihood-ratio estimation underlying clipped policy optimization in the PPO/GRPO family.

**Policy optimization mismatch across turns.** Under flat training, the optimization mismatch across turns arises because the same generated token is conditioned on different contexts during rollout and training. We now formalize this mismatch and show that it induces a biased IS ratio. Let $\bar{x}_k$ denote the full interaction history before turn $k$, and let $\phi : \bar{\mathcal{X}} \to \mathcal{Z}$ be an observation map that truncates or summarizes history beyond the model's effective context length. During rollout, token-level actions are sampled conditioned on $\phi(\bar{x}_k)$, so the true sampling distribution is $\pi_{\theta_{\text{old}}}(a_{k,t} \mid \phi(\bar{x}_k))$. In clipped policy optimization of the PPO/GRPO family, the denominator of the IS ratio must match this rollout distribution. However, under flat training, the same token is re-evaluated under the concatenated full-history context, yielding $\pi_{\theta_{\text{old}}}(a_{k,t} \mid \bar{x}_k)$. This leads to the mismatch

$$\rho_{k,t}^{\text{flat}} = \frac{\pi_\theta(a_{k,t} \mid \bar{x}_k)}{\pi_{\theta_{\text{old}}}(a_{k,t} \mid \bar{x}_k)} \neq \frac{\pi_\theta(a_{k,t} \mid \phi(\bar{x}_k))}{\pi_{\theta_{\text{old}}}(a_{k,t} \mid \phi(\bar{x}_k))} = \rho_{k,t}^{\text{true}}. \tag{10}$$

The denominator used in flat training, $\pi_{\theta_{\text{old}}}(a_{k,t} \mid \bar{x}_k)$, does not match with the true rollout sampling probability, $\pi_{\theta_{\text{old}}}(a_{k,t} \mid \phi(\bar{x}_k))$. Consequently, $\rho_{k,t}^{\text{flat}}$ is a biased estimate of the correct IS ratio. Since clipped policy-gradient updates in the PPO/GRPO family depend on the IS ratio, the bias propagates into the gradient estimate and can distort the update direction, leading to instability in multi-turn agentic RL training. This mismatch becomes more severe for tokens generated in later turns, since the amount of history omitted during rollout typically grows with $k$, while flat training continues to re-evaluate these tokens under the concatenated training prefix. In the worst case, the resulting discrepancy in token probability can become extremely large.

**State aliasing and projected suboptimality.** Beyond gradient bias, truncated contexts also introduce *state aliasing*, where distinct full-history states are mapped to the same truncated representation.

When $\phi$ is non-injective, different interaction histories may collapse into an identical observation $z_k = \phi(\bar{x}_k)$ (Coulom, 2006). The induced process over $z_k$ may not preserve the Markov property of the original full-history process (Allen et al., 2021). As a result, any policy conditioned only on $z_k$ is confined to the observation-induced policy class $\Pi\phi$. Even if optimized exactly within this restricted class, such a policy can achieve only the projected optimum $V^*_{\Pi_\phi}$ (Abel et al., 2020), which can be strictly lower than the true optimum $V^*$. Therefore, projected suboptimality can arise, with $V^*_{\Pi_\phi} < V^*$. Under observation inconsistency, flattened-trajectory training may be limited to the projected optimum $V^*_{\Pi_\phi}$, which is strictly suboptimal relative to the full-history optimum $V^*$.

### B.2 Trajectory-only credit assignment: mismatch across low- and high-quality turns

In multi-turn interactions, trajectories generated from the same query can reach substantially different states by turn $k$. As a result, a trajectory-level return no longer provides a reliable credit signal for evaluating actions taken at that turn. Under flattened-trajectory training, credit is assigned only at the trajectory level and then shared across tokens within a single trajectory, which mismatches the turn-level structure of the decision process. Here, we analyze how a trajectory advantage entangles the contribution of the current turn with both upstream state effect and downstream stochasticity.

**A trajectory advantage entangles turn-level credit with context effects.** A single trajectory advantage is typically defined as $A_i = R_i - \bar{R}$, where $R_i$ denotes the final return of trajectory $g_i$ and $\bar{R}$ is the average return over the sampled group. This trajectory advantage is then assigned to all turns in the trajectory, without identifying each turn's individual contributions. To expose the turn-level credit hidden in this trajectory return, we analyze its population counterpart, $R_i - \mathbb{E}[R]$, by introducing the conditional expectations $\mathbb{E}[R \mid S_{i,k}, l_{i,k}]$ and $\mathbb{E}[R \mid S_{i,k}]$:

$$R_i - \mathbb{E}[R] = \underbrace{R_i - Q^\pi(S_{i,k}, l_{i,k})}_{\xi_{\text{down}}\text{: downstream stochasticity}} + \underbrace{A^\pi(S_{i,k}, l_{i,k})}_{\text{true turn-}k\text{ advantage}} + \underbrace{V^\pi(S_{i,k}) - \mu_R}_{\xi_{\text{up}}\text{: upstream state effect}}, \tag{11}$$

where $Q^\pi(S_{i,k}, l_{i,k}) = \mathbb{E}[R \mid S_{i,k}, l_{i,k}]$, $V^\pi(S_{i,k}) = \mathbb{E}[R \mid S_{i,k}]$, and $\mu_R = \mathbb{E}[R]$. The true turn-$k$ advantage $A^\pi(S_{i,k}, l_{i,k})$ is therefore only one component of the trajectory advantage. The term $\xi_{\text{down}}$ captures downstream stochasticity after turn $k$, while $\xi_{\text{up}}$ captures variation induced by the upstream state reached before turn $k$. Whenever $\xi_{\text{down}} + \xi_{\text{up}}$ dominates $A^\pi(S_{i,k}, l_{i,k})$ in magnitude, the sign of the trajectory advantage can disagree with that of the true turn advantage, i.e., $\text{sign}(A_i) \neq \text{sign}(A^\pi(S_{i,k}, l_{i,k}))$, thereby reversing the gradient direction for turn $k$. Therefore, a trajectory advantage is not a valid turn-level credit signal for turn $k$, because it entangles the context effect of the current turn with both upstream state effects and downstream stochasticity. When these two terms dominate, the resulting policy update can assign incorrect credit to the current turn and may even reverse the intended policy gradient direction.

**Cross-trajectory baseline with state bias.** Since group-based policy optimization methods generate multiple trajectories from the same input query by sampling several rollouts from the current or old policy, they typically compute advantages of the form $A_i = R_i - b$, where $R_i$ is the return of trajectory $g_i$ and $b = \frac{1}{G}\sum_{j=1}^{G} R_j$ is the group baseline. To determine whether the compared trajectories provide a valid turn-level credit signal, we examine whether they share the same turn-level state $S_{i,k}$. If they do, the baseline compares alternative outcomes from the same context and is therefore matched. Otherwise, the baseline mixes returns from different states and no longer reflects the local effect of the current turn, leading to mismatched credit assignment.

Formally, viewing $A_i$ as an estimator of the true turn-$k$ advantage $A^\pi(S_{i,k}, l_{i,k})$ gives

$$\mathbb{E}[A_i \mid S_{i,k}, l_{i,k}] = \frac{G-1}{G}\Big(Q^\pi(S_{i,k}, l_{i,k}) - \underbrace{\mathbb{E}[R_j \mid S_{i,k}]}_{\text{depends on grouping}}\Big), \qquad j \neq i. \tag{12}$$

In the case of same-state grouping, where all trajectories in the group share the same state at turn $k$, we have $\mathbb{E}[R_j \mid S_{i,k}] = V^\pi(S_{i,k})$. In this case, the group baseline is anchored to the correct local decision context, and the estimator's conditional expectation equals the true turn-level advantage multiplied by $(G-1)/G$. However, under cross-state grouping, trajectories within the same group

may already reach different states by turn $k$. Then the baseline is no longer tied to the local state $S_{i,k}$, but is effectively centered around the global mean value as return, i.e., $\mathbb{E}[R_j] = \mu_R$. This introduces the additional bias:

$$\text{Bias}_{\text{cross}} = \frac{G-1}{G}\big(V^\pi(S_{i,k}) - \mu_R\big). \tag{13}$$

This bias is determined entirely by the upstream trajectory and has no causal connection to the action taken at turn $k$. In multi-turn interactions, trajectories often diverge into semantically distinct environment states by turn $k$. For example, one trajectory may issue a search query while another is executing code. As a result, the variance of $V^\pi(S_{i,k})$ across states can be large, making the cross-state bias comparable to, or even larger than, the true turn advantage $A^\pi(S_{i,k}, l_{i,k})$. Therefore, a cross-trajectory baseline cannot guarantee a valid local comparison signal for turn-level credit assignment.

### B.3 Long-horizon policy drift: PPO clipping mismatch asynchronous turns

In long-horizon multi-turn training, policy updates can become asynchronous across turns, inducing turn policy drift. In practice, PPO-style methods use token clipping to constrain the IS ratio and limit policy drift during updates. However, this clipping mechanism is designed for near-on-policy updates with synchronized rollout data and does not explicitly account for turn-wise discrepancies when different parts of a trajectory are generated or optimized under different policy versions. Over long horizons, such mismatches can accumulate, shifting the update away from the near-on-policy learning and toward an off-policy setting.

This occurs because, in asynchronous multi-turn training, policy updates may be performed before all trajectories have completed their rollouts. For instance, shorter trajectories may complete first and immediately contribute to an update, while longer trajectories are still being generated under an older policy. Specifically, short trajectories may finish first and update the policy from $\theta_0$ to $\theta_1$, while long trajectories are still being rolled out under $\pi_{\theta_0}$. By the time these longer trajectories are used for training, the current policy $\pi_{\theta_1}$ no longer matches the policy that generated them.

In principle, this mismatch can be corrected by the full trajectory IS ratio:

$$\omega_i = \prod_{t=1}^{T_i} \rho_{i,t} = \prod_{t=1}^{T_i} \frac{\pi_{\theta_1}(a_{i,t} \mid x_{i,<t})}{\pi_{\theta_0}(a_{i,t} \mid x_{i,<t})}, \tag{14}$$

where $\rho_{i,t}$ is the token IS ratio, $x_{i,<t}$ is the flattened multi-turn interaction context preceding token $a_{i,t}$, and $T_i$ is the total number of tokens in the flattened trajectory $g_i$. Weighting the loss by $\omega_i$ would yield an unbiased correction to the policy objective. However, clipped policy optimization in the PPO/GRPO family does not employ an IS ratio for the full trajectory. Instead, it clips each token ratio $\rho_{i,t}$ independently:

$$\mathcal{L}_{\text{clip}} = \sum_{t=1}^{T_i} \min\big(\rho_{i,t}\, A_i,\ \text{clip}(\rho_{i,t},\, 1{-}\epsilon,\, 1{+}\epsilon)\, A_i\big). \tag{15}$$

Therefore, if one composes the clipped token-level ratios into a trajectory-level correction, it generally differs from the true full-trajectory IS ratio:

$$\prod_{t=1}^{T_i} \text{clip}(\rho_{i,t},\, 1{-}\epsilon,\, 1{+}\epsilon) \neq \prod_{t=1}^{T_i} \rho_{i,t} = \omega_i. \tag{16}$$

Because PPO clipping is nonlinear, the product of clipped token-level ratios does not equal the true trajectory-level importance weight. In particular, each clipped ratio lies in $[1{-}\epsilon,\, 1{+}\epsilon]$, so their product is restricted to $[(1{-}\epsilon)^T,\, (1{+}\epsilon)^T]$, whereas the true $\omega$ can in principle take any value in $(0, \infty)$. Whenever $\omega$ falls outside this interval, per-token clipping necessarily yields a biased trajectory correction. Even when $\omega$ lies within the interval, the product of individually clipped ratios will generally differ from $\omega$ as soon as any $\rho_t$ is clipped, since clipping and multiplication do not commute. This discrepancy compounds with trajectory length $T$: as more tokens are clipped, the gap between $\prod_t \text{clip}(\rho_t)$ and $\omega$ can grow progressively larger. Removing clipping and using

the exact $\omega$ is not a practical solution, because its variance grows exponentially with $T$, making gradient estimates increasingly uninformative over the long horizons typical of multi-turn agentic RL. Hence, token-level clipping cannot faithfully reproduce the trajectory-level correction required for long-horizon policy drift.

# C METHOD: RTPO THEORETICAL PROOFS

## C.1 NOTATION, FORMAL PROBLEM SETUP, AND TECHNICAL ASSUMPTIONS

**Notation convention.** We create a multi-turn agentic episode, which consists of $K$ turn-level interactions, written as $(q, l_0, f_0, \ldots, l_{K-1}, f_{K-1})$, where $q$ is the initial prompt, $l_k$ is the complete response generated by the agent at turn $k$, and $f_k$ is the external tool or environment feedback returned after executing $l_k$. The turn index is denoted by $k \in \{0, \ldots, K-1\}$, while the token index within a response is denoted by $t$. When multiple rollouts are sampled, we use $i$ or $j$ to index the rollout or sibling trajectory. Thus, $S_{i,k}$ denotes the turn-$k$ state in rollout $i$, and $a_{i,k,t}$ denotes the $t$-th token generated at turn $k$ in rollout $i$. When no rollout index is needed, we can write $S_k$, $l_k$, and $a_{k,t}$ for a generic trajectory. We distinguish between a turn-level macro-action and its token-level realization. The macro-action at turn $k$ is denoted by $u_k \equiv l_k$, where $l_k = (a_{k,1}, \ldots, a_{k,T_k})$ is the complete response and $T_k$ is the number of generated tokens in that response. The notation $u_k$ is used in the MDP and value-function definitions, while $l_k$ emphasizes that the macro-action is implemented as a language-model response. In addition, the reward in a task is assigned at the turn or trajectory level. Tokens inside a response are treated as the low-level realization of the turn-level macro-action and do not receive separate intermediate rewards. Thus, the low-level token reward is set to zero. We write $r_k$ for the immediate turn-level reward at turn $k$. In sparse-reward tasks, we typically have $r_k = 0$ for $k < K-1$, and the final task reward is observed only after the terminal turn.

**Turn-boundary MDP for rollout-training match.** A $K$-turn agentic episode is modelled at turn boundaries as $\mathcal{M} = \langle \bar{\mathcal{X}}, \mathcal{A}_H, P_H, R_H, \gamma_H \rangle$, where $\bar{\mathcal{X}}$ is the augmented turn-boundary state space, $\mathcal{A}_H$ is the high-level action space of complete responses, $P_H$ is the transition kernel induced by executing a complete response and receiving tool feedback, $R_H$ is the turn-level reward function, and $\gamma_H$ is the turn-level discount factor. The augmented state at turn $k$ is $\bar{x}_k = (S_k, k) \in \bar{\mathcal{X}}$, where $S_k = (q, l_0, f_0, \ldots, l_{k-1}, f_{k-1})$ is the interaction history available before the current turn. Adding the turn index $k$ in $\bar{x}_k$ can make the process Markov over a finite horizon, since the remaining number of turns can affect both the available decisions and the continuation value. At turn $k$, the agent selects a macro-action $u_k \equiv l_k \in \mathcal{A}_{H,k}(S_k)$, where $\mathcal{A}_{H,k}(S_k)$ denotes the set of feasible complete responses at state $S_k$ and turn $k$. The macro-action is realized autoregressively as $l_k = (a_{k,1}, \ldots, a_{k,T_k})$ and consumes $\tau_k = T_k$ token-generation steps. After executing $u_k$ and receiving environment feedback $f_k$, the turn-level history is updated by concatenation as $S_{k+1} = S_k \circ (l_k, f_k)$. Equivalently, since $u_k \equiv l_k$, one may write $S_{k+1} = S_k \circ (u_k, f_k)$. In Sec 3, we adopt this MDP terminology for the turn-boundary process. Strictly speaking, because each macro-action may span a variable number of token-level steps $\tau_k$, this process can alternatively be formulated as a finite-horizon semi-MDP. This distinction does not affect our analysis, since policy optimization and credit assignment are defined exclusively over turn-boundary states and macro-actions.

To maintain rollout–training consistency under conditional contexts, the policy need not condition on the full interaction history $S_k$. Instead, it may condition on a compressed or truncated context, provided that the same conditional context is used consistently during both rollout generation and policy optimization. Instead, the actual conditioning context at turn $k$ is $c_k = \psi(S_k)$, where $\psi$ may be the identity map, a truncation operator, or a summarization operator. This distinction is important in long-context multi-turn training: rollout may be performed under a truncated or summarized context, while training may otherwise recompute log-probabilities under a different context. RTPO avoids this mismatch by recording the exact context $c_k$ used during rollout and reusing the same $c_k$ during training. The turn-level policy factorizes as $\pi_\theta = (\pi_{\theta,0}, \ldots, \pi_{\theta,K-1})$, where each sub-policy maps the turn-level context $c_k$ to a complete response. Each sub-policy is implemented autoregressively:

$$\pi_{\theta,k}(u_k \mid c_k) = \prod_{t=1}^{T_k} \pi_\theta(a_{k,t} \mid c_k, a_{k,<t}). \tag{17}$$

Here, $a_{k,<t} = (a_{k,1}, \ldots, a_{k,t-1})$ is the token prefix generated within the current turn. Because the same $c_k$ is used in rollout and training, the token-level importance-sampling (IS) ratio for turn $k$ is evaluated under matched conditioning contexts:

$$\rho_{k,t} = \frac{\pi_\theta(a_{k,t} \mid c_k, a_{k,<t})}{\pi_{\theta_{\text{old}}}(a_{k,t} \mid c_k, a_{k,<t})}. \tag{18}$$

**Policy optimization with recursive optimality.** A policy sequence $\pi^{\text{rec}} = (\pi_0^*, \ldots, \pi_{K-1}^*)$ is recursively optimal if, for every turn $k \in \{0, \ldots, K-1\}$, the turn-$k$ sub-policy $\pi_k^*$ maximizes the turn-level augmented value given that all downstream sub-policies have already been optimized and fixed. Formally, for each $k$, $\pi_k^* \in \arg\max_{\pi_k} \tilde{Q}_k^{\pi^*}(S_k, u_k)$, where the downstream policies $\pi_{k+1}^*, \ldots, \pi_{K-1}^*$ are treated as fixed during the optimization of $\pi_k^*$. This is a backward-induction notion of optimality: the last turn is optimized first, then the preceding turn is optimized assuming the last-turn policy is fixed, and so on until the first turn.

We establish convergence to recursive optimality in a tabular finite-horizon setting under Assumptions (A1–A11), following backward policy search (Bagnell et al., 2003).

(A1) *Finite turn-boundary spaces.* The augmented state space $\bar{\mathcal{X}}$ is finite, and for every turn $k$ and every reachable state $S_k$, the feasible high-level action set $\mathcal{A}_{H,k}(S_k)$ is finite.

(A2) *Proper finite-horizon episodes.* The number of turns $K$ is finite. For every policy and every turn $k$, the macro-action duration satisfies $\mathbb{E}_{\pi_k}[\tau_k] < \infty$. Thus, each turn terminates almost surely in a finite expected token length.

(A3) *Discounting or finite-horizon boundedness.* Either $\gamma_H \in (0, 1)$, or the problem is finite-horizon with $K < \infty$ and $\gamma_H \in (0, 1]$. The latter case includes the undiscounted finite-horizon setting $\gamma_H = 1$.

(A4) *Tabular value representation.* The value estimate $Q_k(S, u)$ is stored separately for each turn-state-action tuple $(S, u, k)$. This assumption avoids approximation error and allows the proof to focus on the stochastic approximation dynamics induced by reverse-order updates.

(A5) *Reverse-order training with downstream freezing.* Training is proceeding in the order $k = K-1, K-2, \ldots, 0$. During turn $k$, the turn-$k$ policy is updated while the downstream policies $\pi_{k+1:K-1}$ are held fixed. The stage is taken to its asymptotic limit, and a common optimal greedy selector for states sharing $c_k$ is chosen from the limiting values and frozen before moving to turn $k-1$.

(A6) *GLIE exploration within each turn.* Within turn $k$, the exploration schedule is greedy in the limit with infinite exploration (GLIE): every feasible turn-level action at each reachable state is sampled infinitely often almost surely, while the policy becomes greedy in the limit. This ensures that all relevant action values at turn $k$ are sufficiently sampled before the policy is frozen.

(A7) *State coverage.* Every boundary state $S_k$ reachable under any feasible policy is visited infinitely often during turn $k$. This condition ensures that the tabular value estimate for each relevant state-action pair receives infinitely many updates.

(A8) *Robbins–Monro step sizes.* For every tuple $(S, u, k)$, the learning rates satisfy $\sum_{n=1}^{\infty} \alpha_n(S, u, k) = \infty$ and $\sum_{n=1}^{\infty} \alpha_n^2(S, u, k) < \infty$. These are the standard stochastic approximation step-size conditions.

(A9) *Bounded rewards.* The turn-level rewards are uniformly bounded: $|r_k| \le r_{\max}$ almost surely for all $k$.

(A10) *Bounded iterates.* The tabular value iterates remain uniformly bounded: $|Q_k(S, u)| \le Q_{\max}$ throughout learning. This assumption is standard in stochastic approximation analyses and can be enforced by projection if necessary.

(A11) *Decision sufficiency of the conditioning context.* The observation mapping $\psi\colon S_k \mapsto c_k$ preserves all decision-relevant information for following turns. In other words, states that are indistinguishable under $\psi$ share the same feasible macro-actions and optimal action-value function, so the optimal policy at turn $k$ depends on $S_k$ only through $c_k$. This implies that $\tilde{Q}_k^{\pi^*}$ can be written as a function of $(c_k, u)$ without loss. The completeness condition in Theorem 1(c) is imposed separately.

**Relation to the neural implementation.** Theorem 1 intentionally analyzes an idealized tabular counterpart of RTPO to isolate the structural effect of reverse-order optimization from function approximation and parameter sharing. Under Assumptions (A1)–(A11), each tabular turn is optimized asymptotically while downstream policies remain fixed. The practical implementation in Eq. (6) instead performs a finite number of clipped policy updates with shared neural parameters. Accordingly, Theorem 1 should be interpreted as a convergence result for the underlying tabular reverse-order formulation, rather than as a convergence guarantee for the shared-parameter neural optimizer. At each subsequent stage, sibling continuations are regenerated using the current parameter snapshot, preserving the reverse-order continuation structure analyzed above.

**Decision sufficiency (A11) in RTPO.** In practical RTPO, $\psi$ removes internal reasoning traces while preserving tool calls, tool results, and final answers. Assumption (A11) holds if states sharing the retained context have identical feasible macro-action sets and, for each macro-action, identical joint distributions of reward, duration, and next retained context. Backward induction then gives identical optimal action values. When $\psi$ is the identity map, (A11) holds trivially.

**Turn-level value estimation.** We formulate the interaction as a hierarchical MDP, which decomposes the value of each turn into the local effect of the current macro-action and the downstream completion value, following a MAXQ-style value decomposition. The downstream continuation value after executing turn $k$ is defined as

$$F_k^\pi(S_{k+1}) := \mathbb{E}_\pi\left[\sum_{j=k+1}^{K-1} \gamma_H^{\sum_{m=k+1}^{j-1}\tau_m} r_j \,\middle|\, S_{k+1}\right], \qquad F_{K-1}^\pi \equiv 0. \tag{19}$$

The base case $F_{K-1}^\pi \equiv 0$ reflects that there are no downstream turns after the last turn. The exponent $\sum_{m=k+1}^{j-1}\tau_m$ accounts for the number of token-generation steps elapsed between the next state $S_{k+1}$ and the future reward $r_j$. When $\gamma_H = 1$, this reduces to the undiscounted finite-horizon setting used in many sparse-reward agentic RL tasks. The boundary-level augmented action-value function is:

$$\tilde{Q}_k^\pi(S_k, u_k) := \mathbb{E}[r_k + \gamma_H^{\tau_k} F_k^\pi(S_{k+1}) \mid S_k, u_k]\,. \tag{20}$$

This value measures the expected return obtained by choosing the complete turn-$k$ response $u_k$ at state $S_k$, followed by downstream policy execution from $S_{k+1}$ onward. In sparse-reward settings, the immediate term $r_k$ is often zero for nonterminal turns, so the quality of a turn-level action is primarily reflected through the downstream continuation value. The corresponding optimal value satisfies $\tilde{Q}_k^*(S_k, u_k) = \mathbb{E}[r_k + \gamma_H^{\tau_k} F_k^*(S_{k+1}) \mid S_k, u_k]$, where $F_k^*$ is obtained by the usual backward Bellman optimality recursion over turn boundaries.

### C.2 Proof of Theorem 1: Convergence to Recursive Optimality

We formalize Theorem 1 from Sec 3.1 before presenting the proof.

**Reverse-Order Training to Recursive Optimality**

**Theorem 1 (Convergence to Recursive Optimality).** *Under Assumptions (A1)–(A11), including finite state and macro-action spaces, Robbins–Monro step sizes, sufficient exploration at each turn, exact freezing of optimized downstream policies, and bounded rewards, the reverse-order tabular updates below satisfy the following properties:*

*(a)* ***Per-turn convergence under fixed downstream policies.*** *For each turn $k$, once the downstream policies $\pi^*_{k+1:K-1}$ are fixed, the downstream continuation value becomes a stationary function:*

$$F_k^{\pi^*}(S_{k+1}) = \mathbb{E}_{\pi^*_{k+1:K-1}}\left[\sum_{j=k+1}^{K-1} \gamma_H^{\sum_{m=k+1}^{j-1}\tau_m} r_j \,\middle|\, S_{k+1}\right]. \tag{21}$$

*Therefore, the turn-$k$ target*

$$Y_k := r_k + \gamma_H^{\tau_k} F_k^{\pi^*}(S_{k+1}) \tag{22}$$

*is stationary conditional on $(S_k, u_k)$, with conditional mean*

$$\mathbb{E}[Y_k \mid S_k, u_k] = \tilde{Q}_k^{\pi^*}(S_k, u_k). \tag{23}$$

*The corresponding tabular update*

$$Q_k^{(n+1)}(S, u) \leftarrow (1 - \alpha_n) Q_k^{(n)}(S, u) + \alpha_n Y_k \tag{24}$$

*is a single-step stochastic approximation problem and converges to $\tilde{Q}_k^{\pi^*}$ under the stated assumptions. Hence, its greedy action-value gap tends to zero almost surely.*

*(b)* ***Recursive optimality under reverse-order training.*** *Applying the per-turn convergence argument in reverse order,*

$$k = K-1, K-2, \ldots, 0, \tag{25}$$

*yields a sequence of optimized and frozen downstream policies. At each turn $k$, an optimal greedy selector is chosen from the limiting values:*

$$\pi_k^*(S_k) \in \arg \max_{u_k \in \mathcal{A}_{H,k}(S_k)} \tilde{Q}_k^{\pi^*}(S_k, u_k), \tag{26}$$

*given the fixed downstream policies $\pi_{k+1}^*, \ldots, \pi_{K-1}^*$. The resulting policy sequence*

$$\pi^{\mathrm{rec}} = (\pi_0^*, \ldots, \pi_{K-1}^*) \tag{27}$$

*is recursively optimal.*

*(c)* ***Global optimality under context-sufficient macro-action completeness.*** *If the turn-level macro-action spaces are complete with respect to the conditioning contexts $c_k = \psi(S_k)$, i.e., every globally feasible trajectory-level policy can be represented by a sequence of turn-level policies acting on these contexts,*

$$\Pi_{\mathrm{global}} \subseteq \{(\pi_0, \ldots, \pi_{K-1}) : \pi_k(\cdot \mid c_k),\ u_k \in \mathcal{A}_{H,k}(S_k)\}, \tag{28}$$

*Then, recursive optimality is equivalent to global optimality over the full trajectory:*

$$V^{\pi^{\mathrm{rec}}}(\bar{x}_0) = V^*(\bar{x}_0). \tag{29}$$

*Proof.* We prove parts (a)–(c) of Theorem 1 as follows.

**Proof of Theorem 1(a): per-turn convergence under fixed downstream policies.** Fix a turn $k$ and suppose the downstream sub-policies $\pi_{k+1}^*, \ldots, \pi_{K-1}^*$ are already optimized and frozen. By the definition in Theorem 1, the downstream continuation value is

$$F_k^{\pi^*}(S_{k+1}) = \mathbb{E}_{\pi_{k+1:K-1}^*}\left[\sum_{j=k+1}^{K-1} \gamma_H^{\sum_{m=k+1}^{j-1} \tau_m} r_j \,\middle|\, S_{k+1}\right]. \tag{30}$$

Because the downstream policies $\pi_{k+1:K-1}^*$ are frozen, $F_k^{\pi^*}(S_{k+1})$ is a fixed function of $S_{k+1}$ throughout turn $k$. Define the augmented one-step target

$$Y_k = r_k + \gamma_H^{\tau_k} F_k^{\pi^*}(S_{k+1}). \tag{31}$$

Conditional on $(S_k, u_k)$, the distribution of $r_k$, $\tau_k$, and $S_{k+1}$ is induced by the time-invariant turn-boundary transition kernel $P_H$. Since $F_k^{\pi^*}$ is fixed throughout turn $k$, the conditional distribution of $Y_k$ given $(S_k, u_k)$ is stationary.

The conditional mean of the target is exactly the augmented turn-level action value:

$$\mathbb{E}[Y_k \mid S_k, u_k] = \mathbb{E}\Big[r_k + \gamma_H^{\tau_k} F_k^{\pi^*}(S_{k+1}) \,\Big|\, S_k, u_k\Big] = \tilde{Q}_k^{\pi^*}(S_k, u_k). \tag{32}$$

The target also has a bounded second moment. Since rewards are bounded, $|r_j| \le r_{\max}$ almost surely. Hence, for any policy $\pi$,

$$\begin{aligned}\left|F_k^{\pi}(S_{k+1})\right| &= \left|\mathbb{E}_{\pi}\left[\sum_{j=k+1}^{K-1} \gamma_H^{\sum_{m=k+1}^{j-1}\tau_m} r_j \,\middle|\, S_{k+1}\right]\right| \\ &\le \mathbb{E}_{\pi}\left[\sum_{j=k+1}^{K-1} \gamma_H^{\sum_{m=k+1}^{j-1}\tau_m} |r_j| \,\middle|\, S_{k+1}\right] \\ &\le (K-k-1) r_{\max} \le K r_{\max}, \end{aligned} \tag{33}$$

where we used $\gamma_H \in (0,1]$ and the finite horizon $K$. Therefore,

$$|Y_k| \le |r_k| + \gamma_H^{\tau_k} |F_k^{\pi^*}(S_{k+1})| \le (K+1) r_{\max}, \tag{34}$$

so $Y_k$ has uniformly bounded conditional second moment.

During turn $k$, the tabular update is

$$Q_k^{(n+1)}(S,u) \leftarrow (1-\alpha_n) Q_k^{(n)}(S,u) + \alpha_n Y_k. \tag{35}$$

For each fixed $(S,u,k)$, this is a Robbins–Monro stochastic approximation to the stationary conditional mean $\mathbb{E}[Y_k \mid S,u] = \tilde{Q}_k^{\pi^*}(S,u)$. The finite state and macro-action spaces imply that there are finitely many entries to estimate. Sufficient exploration ensures that every relevant $(S,u)$ pair is visited infinitely often during turn $k$. The step sizes satisfy

$$\sum_{n=1}^{\infty} \alpha_n(S,u,k) = \infty, \qquad \sum_{n=1}^{\infty} \alpha_n^2(S,u,k) < \infty. \tag{36}$$

Together with bounded targets and bounded iterates, standard stochastic approximation gives

$$Q_k^{(n)}(S,u) \xrightarrow{\text{a.s.}} \tilde{Q}_k^{\pi^*}(S,u) \qquad \text{for every reachable } (S,u). \tag{37}$$

For a greedy selector $\hat{\pi}_k^{(n)}(S) \in \arg\max_u Q_k^{(n)}(S,u)$, value convergence gives

$$\max_u \tilde{Q}_k^{\pi^*}(S,u) - \tilde{Q}_k^{\pi^*}\left(S, \hat{\pi}_k^{(n)}(S)\right) \xrightarrow[n\to\infty]{\text{a.s.}} 0. \tag{38}$$

This completes the proof of Theorem 1(a).

**Proof of Theorem 1(b): reverse-order recursion gives recursive optimality.** We now apply Theorem 1(a) in reverse order here. At the final turn $k = K-1$, there are no downstream turns, so

$$F_{K-1}^{\pi}(S_K) \equiv 0. \tag{39}$$

The augmented target reduces to

$$Y_{K-1} = r_{K-1}. \tag{40}$$

By Theorem 1(a), the final-turn values converge, and we select

$$\pi_{K-1}^*(S) \in \arg\max_u \tilde{Q}_{K-1}^{\pi^*}(S,u). \tag{41}$$

Now assume that, for some $k \in \{0,\dots,K-2\}$, downstream turns have been solved in the asymptotic limit and optimal selectors $\pi_{k+1}^*, \dots, \pi_{K-1}^*$ have been frozen. Then the continuation value

$$F_k^{\pi^*}(S_{k+1}) = \mathbb{E}_{\pi_{k+1:K-1}^*}\left[\sum_{j=k+1}^{K-1} \gamma_H^{\sum_{m=k+1}^{j-1}\tau_m} r_j \,\middle|\, S_{k+1}\right] \tag{42}$$

is fixed throughout turn $k$. Therefore, claim (a) applies to turn $k$, and its limiting values admit

$$\pi_k^*(S_k) \in \arg\max_{u_k} \tilde{Q}_k^{\pi^*}(S_k, u_k). \tag{43}$$

By backward induction from $K-1$ to 0, an optimal selector is obtained at every turn given the optimized and frozen downstream policies. Therefore, the resulting sequence

$$\pi^{\text{rec}} = (\pi_0^*, \dots, \pi_{K-1}^*) \tag{44}$$

is recursively optimal.

Till here, we have completed the proof of Theorem 1(b).

**Proof of Theorem 1(c): context-sufficient macro-action completeness implies global optimality.** Assume the turn-level macro-action spaces are complete with respect to the conditioning contexts $c_k = \psi(S_k)$, meaning that every globally feasible trajectory-level policy can be represented as a sequence of turn-level sub-policies $(\pi_0, \dots, \pi_{K-1})$, with each $\pi_k$ acting on $c_k$ and selecting a feasible macro-action $u_k \in \mathcal{A}_{H,k}(S_k)$.

We prove by backward induction that

$$V_k^{\pi^{\mathrm{rec}}}(S_k) = V_k^*(S_k) \qquad \text{for every reachable } S_k. \tag{45}$$

At the last turn, $F_{K-1}^{\pi} \equiv 0$, so

$$\tilde{Q}_{K-1}^{\pi}(S_{K-1}, u) = \mathbb{E}[r_{K-1} \mid S_{K-1}, u]. \tag{46}$$

Since $\pi_{K-1}^*$ maximizes this quantity,

$$V_{K-1}^{\pi^{\mathrm{rec}}}(S_{K-1}) = \max_{u \in \mathcal{A}_{H,K-1}(S_{K-1})} \mathbb{E}[r_{K-1} \mid S_{K-1}, u] = V_{K-1}^*(S_{K-1}). \tag{47}$$

Assume now that $V_{k+1}^{\pi^{\mathrm{rec}}}(S_{k+1}) = V_{k+1}^*(S_{k+1})$ for every reachable $S_{k+1}$. Then

$$F_k^{\pi^{\mathrm{rec}}}(S_{k+1}) = F_k^*(S_{k+1}). \tag{48}$$

By Assumption (A11), the optimal action-value $\tilde{Q}_k^{\pi^*}(S_k, u)$ depends on $S_k$ only through $c_k = \psi(S_k)$. Therefore, the greedy policy $\pi_k^* \in \arg\max_u \tilde{Q}_k^{\pi^*}(S_k, u)$ is well-defined as a function of $c_k$ alone, and we write $\pi_k^*(c_k)$ without ambiguity. Applying this to the inductive step:

$$\begin{aligned} V_k^{\pi^{\mathrm{rec}}}(S_k) &= \tilde{Q}_k^{\pi^{\mathrm{rec}}}\big(S_k, \pi_k^*(c_k)\big) \\ &= \mathbb{E}\Big[r_k + \gamma_H^{\tau_k} F_k^{\pi^{\mathrm{rec}}}(S_{k+1}) \;\Big|\; S_k, \pi_k^*(S_k)\Big] \\ &= \mathbb{E}[r_k + \gamma_H^{\tau_k} F_k^*(S_{k+1}) \mid S_k, \pi_k^*(S_k)] \\ &= \max_{u \in \mathcal{A}_{H,k}(S_k)} \mathbb{E}[r_k + \gamma_H^{\tau_k} F_k^*(S_{k+1}) \mid S_k, u] \\ &= V_k^*(S_k). \end{aligned} \tag{49}$$

Thus $V_k^{\pi^{\mathrm{rec}}}(S_k) = V_k^*(S_k)$ for all $k$ by induction. In particular,

$$V^{\pi^{\mathrm{rec}}}(\bar{x}_0) = V^*(\bar{x}_0). \tag{50}$$

This proves Theorem 1(c), and completes the proof of Theorem 1. □

**Interpretation.** As the key technical point of this proof, turn-$k$ target $Y_k = r_k + \gamma_H^{\tau_k} F_k^{\pi^*}(S_{k+1})$ is stationary when the downstream policies $\pi_{k+1}^*, \dots, \pi_{K-1}^*$ are fixed. Assumption (A5) provides this condition.

### C.3 Proof of Theorem 2: Causally Consistent Turn-Level Advantage Estimation

We formalize Theorem 2 from Sec 3.2 before presenting the proof. We compare the state-matched sibling advantage estimator used by RTPO with the trajectory-level advantage estimator used in flat-trajectory training. The key distinction is whether the baseline is computed from rollouts sharing the same boundary state $S_k$, or from complete trajectories that may have reached different boundary states by turn $k$.

**Turn-Level Credit Assignment without State Bias**

**Theorem 2 (Causally Consistent Turn-Level Advantage Estimation).** *Consider turn $k$ and suppose that $G-1$ sibling rollouts are forked from the same boundary state $S_k$. Each sibling rollout $j \in \{1, \dots, G-1\}$ independently samples a turn-$k$ macro-action $u_{j,k} \equiv l_{j,k}$*

*and then continues to the terminal. Let*

$$\hat{Q}_{j,k} = R_j, \qquad \hat{V}_k = \frac{1}{G-1}\sum_{j=1}^{G-1} \hat{Q}_{j,k}, \qquad A^H_{j,k} = \hat{Q}_{j,k} - \hat{V}_k \tag{51}$$

*denote the Monte Carlo turn-level action-value estimate, the state-matched sibling baseline, and the resulting turn-level advantage for sibling rollout $j$, respectively. Let the corresponding flat trajectory-level estimator be*

$$\bar{R} = \frac{1}{G}\sum_{i=1}^{G} R_i, \qquad A^{\mathrm{traj}}_i = R_i - \bar{R}, \tag{52}$$

*where $g_i$ denotes the $i$-th complete trajectory and $S_{i,k}$ is its boundary state at turn $k$. Define*

$$Q^\pi(S_k, u_{j,k}) = \mathbb{E}[R \mid S_k, u_{j,k}], \qquad V^\pi(S_k) = \mathbb{E}_{u\sim\pi(\cdot|c_k)}[Q^\pi(S_k, u)], \tag{53}$$

*and*

$$A^\pi(S_k, u_{j,k}) = Q^\pi(S_k, u_{j,k}) - V^\pi(S_k), \qquad \mu_R = \mathbb{E}[R]. \tag{54}$$

*Assume bounded rewards and $G \geq 3$. For a fixed prompt and policy, let siblings be conditionally i.i.d. given $S_k$ and trajectories be i.i.d., with the same boundary-state marginal. Then:*

*(a)* ***Local unbiasedness up to finite-group bias.*** *Conditional on the shared boundary state $S_k$ and the sampled macro-action $u_{j,k}$, the state-matched sibling estimator satisfies:*

$$\mathbb{E}\big[A^H_{j,k} \mid S_k, u_{j,k}\big] = \frac{G-2}{G-1} A^\pi(S_k, u_{j,k}). \tag{55}$$

*Thus, $A^H_{j,k}$ estimates the true turn-level advantage up to a finite-group bias of order $O(1/G)$ and contains no upstream state-contamination term. In contrast, the trajectory-level estimator satisfies:*

$$\mathbb{E}\Big[A^{\mathrm{traj}}_i \mid S_{i,k}, u_{i,k}\Big] = \frac{G-1}{G} A^\pi(S_{i,k}, u_{i,k}) + \frac{G-1}{G}\big(V^\pi(S_{i,k}) - \mu_R\big), \tag{56}$$

*where the second term is the upstream state-contamination term.*

*(b)* ***Reduced cross-state estimation error.*** *Let*

$$\sigma_V^2 := \mathrm{Var}_{S_k}[V^\pi(S_k)] \tag{57}$$

*denote the variance of boundary-state values. Compared with the trajectory-level estimator, the sibling estimator removes the cross-state value component $V^\pi(S_{i,k}) - \mu_R$. Therefore, the trajectory estimator contains an additional MSE contribution proportional to $\sigma_V^2$, while the sibling estimator does not. Eq.* (75) *gives the exact finite-group MSE difference.*

*(c)* ***State-matched causal actions.*** *All sibling rollouts share the same boundary state $S_k$. Therefore, differences in their returns are attributable to the sampled turn-$k$ macro-actions $u_{j,k}$ and independent downstream sampling noise, rather than to upstream trajectory differences. The resulting advantage $A^H_{j,k}$ is assigned only to the output tokens of turn $k$,*

$$l_{j,k} = (a_{j,k,1}, \ldots, a_{j,k,T_{j,k}}), \tag{58}$$

*while the prefix tokens contained in the shared context $c_k = \psi(S_k)$ are excluded from the direct token-level loss.*

*Proof.* We prove parts (a)–(c) of Theorem 2 as follows.

**Proof of Theorem 2(a): local unbiasedness up to finite-group bias.** Fix a turn $k$ and a shared boundary state $S_k$. RTPO forks $G-1$ sibling rollouts from the same environment snapshot $\mathrm{snap}_k$. Conditional on $S_k$, each sibling rollout independently samples its turn-$k$ macro-action and down-

stream continuation. Thus, for any two distinct sibling rollouts $j \neq j'$, the return $R_{j'}$ of sibling $j'$ is conditionally independent of the action $u_{j,k}$ sampled by sibling $j$:

$$R_{j'} \perp\!\!\!\perp u_{j,k} \mid S_k. \tag{59}$$

Consequently,

$$\mathbb{E}[R_{j'} \mid S_k, u_{j,k}] = \mathbb{E}[R_{j'} \mid S_k] = V^\pi(S_k). \tag{60}$$

Meanwhile, for the own return of sibling $j$,

$$\mathbb{E}[R_j \mid S_k, u_{j,k}] = Q^\pi(S_k, u_{j,k}). \tag{61}$$

By definition,

$$\begin{aligned} A^H_{j,k} &= R_j - \frac{1}{G-1}\sum_{r=1}^{G-1} R_r \\ &= \frac{G-2}{G-1} R_j - \frac{1}{G-1}\sum_{r\neq j} R_r. \end{aligned} \tag{62}$$

Taking conditional expectation given $(S_k, u_{j,k})$ and using Eq. (60) and Eq. (61),

$$\begin{aligned} \mathbb{E}[A^H_{j,k} \mid S_k, u_{j,k}] &= \frac{G-2}{G-1} Q^\pi(S_k, u_{j,k}) - \frac{G-2}{G-1} V^\pi(S_k) \\ &= \frac{G-2}{G-1}\big(Q^\pi(S_k, u_{j,k}) - V^\pi(S_k)\big) \\ &= \frac{G-2}{G-1} A^\pi(S_k, u_{j,k}). \end{aligned} \tag{63}$$

Therefore, the sibling estimator is locally unbiased up to the finite-group multiplicative factor $(G-2)/(G-1)$. Equivalently, the conditional bias relative to the true turn-level advantage is

$$\mathbb{E}[A^H_{j,k} \mid S_k, u_{j,k}] - A^\pi(S_k, u_{j,k}) = -\frac{1}{G-1} A^\pi(S_k, u_{j,k}), \tag{64}$$

which is $O(1/G)$ and contains no term depending on $V^\pi(S_k) - \mu_R$.

We now contrast this with the trajectory-level estimator. In flat-trajectory training, the $G$ trajectories are independently sampled from the same prompt but may reach different boundary states by turn $k$. The trajectory-level advantage is

$$\begin{aligned} A^{\mathrm{traj}}_i &= R_i - \frac{1}{G}\sum_{r=1}^{G} R_r \\ &= \frac{G-1}{G} R_i - \frac{1}{G}\sum_{r\neq i} R_r. \end{aligned} \tag{65}$$

The own return satisfies

$$\mathbb{E}[R_i \mid S_{i,k}, u_{i,k}] = Q^\pi(S_{i,k}, u_{i,k}). \tag{66}$$

For $r \neq i$, trajectory $r$ is generated by an independent upstream rollout and reaches its own boundary state $S_{r,k}$. Thus,

$$\mathbb{E}[R_r \mid S_{i,k}, u_{i,k}] = \mathbb{E}_{S_{r,k}}[V^\pi(S_{r,k})] = \mu_R. \tag{67}$$

Substituting Eq. (66) and Eq. (67) into Eq. (65) gives

$$\begin{aligned} \mathbb{E}[A^{\mathrm{traj}}_i \mid S_{i,k}, u_{i,k}] &= \frac{G-1}{G}\big(Q^\pi(S_{i,k}, u_{i,k}) - \mu_R\big) \\ &= \frac{G-1}{G}\big(Q^\pi(S_{i,k}, u_{i,k}) - V^\pi(S_{i,k})\big) + \frac{G-1}{G}\big(V^\pi(S_{i,k}) - \mu_R\big) \\ &= \frac{G-1}{G} A^\pi(S_{i,k}, u_{i,k}) + \frac{G-1}{G}\big(V^\pi(S_{i,k}) - \mu_R\big). \end{aligned} \tag{68}$$

The second term in Eq. (68) depends only on the upstream boundary state $S_{i,k}$ and the global mean return $\mu_R$. It has no causal dependence on the current turn action $u_{i,k}$ and is precisely the cross-state contamination term. Therefore, the sibling estimator removes the upstream state-contamination term present in the trajectory-level estimator. This proves part (a) of Theorem 2.

**Proof of Theorem 2(b): reduced cross-state estimation error.** We compare the error of each estimator against its corresponding true turn-level advantage. For the sibling estimator, the target is $A^\pi(S_k, u_{j,k})$; for the trajectory estimator, the target is $A^\pi(S_{i,k}, u_{i,k})$. The trajectory estimator contains the additional cross-state term

$$C_{i,k} := \frac{G-1}{G}\big(V^\pi(S_{i,k}) - \mu_R\big), \tag{69}$$

whereas the sibling estimator does not contain such a term. Marginalizing over the boundary-state distribution,

$$\mathbb{E}[C_{i,k}] = 0, \qquad \mathrm{Var}(C_{i,k}) = \left(\frac{G-1}{G}\right)^2 \mathrm{Var}_{S_k}[V^\pi(S_k)] = \left(\frac{G-1}{G}\right)^2 \sigma_V^2. \tag{70}$$

Thus, cross-trajectory baselines introduce an additional MSE component proportional to $\sigma_V^2$, while state-matched sibling baselines remove it.

In addition to the squared-bias contribution above, the other-trajectory contribution $\frac{1}{G}\sum_{r\neq i} R_r$ introduces further variance from cross-state effects. Conditional on $(S_{i,k}, u_{i,k})$, each $R_r$ $(r \neq i)$ is independent of the conditioning, so its conditional variance equals its unconditional variance:

$$\mathrm{Var}(R_r \mid S_{i,k}, u_{i,k}) = \mathrm{Var}(R_r) = \mathbb{E}_{S_{r,k}}[\mathrm{Var}(R_r \mid S_{r,k})] + \ \sigma_V^2, \tag{71}$$

where the $\sigma_V^2$ term arises because $S_{r,k}$ varies across trajectories. This contributes $\frac{G-1}{G^2}\sigma_V^2$ to the conditional variance of $A_i^{\mathrm{traj}}$. By contrast, in the sibling estimator, all returns share the same boundary state $S_k$, so $\mathrm{Var}(R_r \mid S_k)$ is a purely within-state quantity and contains no $\sigma_V^2$ component.

More explicitly, marginal MSE averages conditional variance and squared conditional bias:

$$\begin{aligned}\mathrm{MSE}(A_i^{\mathrm{traj}}) = \mathbb{E}\big[\,&\mathrm{Var}(A_i^{\mathrm{traj}} \mid S_{i,k}, u_{i,k}) \\ &+ \mathrm{Bias}(A_i^{\mathrm{traj}} \mid S_{i,k}, u_{i,k})^2\big].\end{aligned} \tag{72}$$

For the trajectory estimator, the bias contains the cross-state term $C_{i,k}$ from Eq. (69). For the sibling estimator, the only systematic bias from part (a) is the finite-group term

$$-\frac{1}{G-1}A^\pi(S_k, u_{j,k}), \tag{73}$$

which is $O(1/G)$. Therefore, the total $\sigma_V^2$ coefficient in the trajectory estimator's marginal MSE combines the squared-bias contribution (Eq. (70)) with the baseline-variance contribution (Eq. (71)):

$$\underbrace{\left(\frac{G-1}{G}\right)^2}_{\mathrm{Bias}^2} + \underbrace{\frac{G-1}{G^2}}_{\mathrm{Var}} \ = \ \frac{(G-1)^2 + (G-1)}{G^2} \ = \ \frac{G-1}{G}. \tag{74}$$

The sibling estimator incurs no $\sigma_V^2$ contribution. Using $\mathbb{E}[A^\pi(S_{i,k}, u_{i,k}) \mid S_{i,k}] = 0$ and the law of total variance, the exact MSE difference is:

$$\begin{aligned}\mathrm{MSE}(A_i^{\mathrm{traj}}) - \mathrm{MSE}(A_{j,k}^H) = \frac{G-1}{G}\sigma_V^2 \\ + \frac{\mathbb{E}[\mathrm{Var}(R_i \mid S_{i,k}, u_{i,k})] - \mathbb{E}[(A^\pi(S_{i,k}, u_{i,k}))^2]}{G(G-1)}.\end{aligned} \tag{75}$$

Hence the sibling estimator has strictly lower MSE if and only if the right-hand side is positive. This proves part (b) of Theorem 2.

**Proof of Theorem 2(c): state-matched causal actions.** All sibling rollouts are forked from the same boundary state $S_k$. Therefore, conditional on $S_k$, sibling rollout $j$ and sibling rollout $j'$ differ only in their sampled turn-$k$ macro-actions and their independent downstream randomness:

$$(u_{j,k}, \xi_j) \quad \text{versus} \quad (u_{j',k}, \xi_{j'}), \tag{76}$$

where $\xi_j$ denotes the downstream randomness of sibling $j$. Since the shared prefix state $S_k$ is identical across siblings, any systematic difference in conditional expected return is attributable to the sampled turn-$k$ macro-action:

$$\mathbb{E}[R_j - R_{j'} \mid S_k, u_{j,k}, u_{j',k}] = Q^\pi(S_k, u_{j,k}) - Q^\pi(S_k, u_{j',k}). \tag{77}$$

The downstream randomness contributes to Monte Carlo noise but does not create an upstream state bias, because the upstream state is shared and fixed.

Finally, RTPO assigns the resulting advantage only to the output tokens of turn $k$. For sibling rollout $j$, these tokens are

$$l_{j,k} = (a_{j,k,1}, \ldots, a_{j,k,T_{j,k}}). \tag{78}$$

The shared prefix $c_k = \psi(S_k)$ provides the conditioning input and has no direct token-level loss. The token positions contributing terms to $J_k(\theta)$ in Eq. (6) are

$$\{(j,k,t) : j \in \{1,\ldots,G-1\},\ t \in \{1,\ldots,T_{j,k}\}\}. \tag{79}$$

Thus, direct credit is assigned only to turn-$k$ output tokens. This proves part (c) and completes the proof of Theorem 2. □

**Interpretation.** The proof shows that the key difference between RTPO and flat-trajectory training is the baseline state. The sibling estimator compares alternative turn-$k$ actions from the same boundary state $S_k$, so the baseline estimates the local value $V^\pi(S_k)$. In contrast, a flat-trajectory baseline compares returns from trajectories that may have reached different boundary states by turn $k$, so it is centered around the global mean $\mu_R$. The resulting term $V^\pi(S_{i,k}) - \mu_R$ is an upstream state-contamination term: it can dominate the true turn-level advantage even though it is unrelated to the current turn action. RTPO removes this term by constructing state-matched sibling rollouts and assigning the resulting advantage only to the output tokens of the current turn.

### C.4 Proof of Theorem 3: On-Policy Continuation under Asynchronous Turns

We formalize Theorem 3 from Sec 3.3 before presenting the proof. The purpose is to show that RTPO avoids the off-policy drift induced by stale downstream rollouts. In asynchronous multi-turn training, a trajectory generated under an old policy may later be evaluated under an updated policy, which would require a long-horizon trajectory-level importance-sampling correction. RTPO avoids this issue by regenerating sibling continuations on-policy at each reverse-order turn.

**Setup.** Let $\theta_0$ denote the rollout parameters at the beginning of a training iteration, and let $\theta_{>k}$ denote the current parameters available at the start of turn $k$, after downstream turns $k+1, \ldots, K-1$ have already received their stage updates. A stale-rollout alternative would retain sibling continuations generated under $\pi_{\theta_0}$ and correct them using the trajectory-level importance-sampling weight:

$$\omega_j^{\text{old}} = \prod_{h=k+1}^{K-1} \prod_{t=1}^{T_{j,h}} \frac{\pi_{\theta_{>k}}(a_{j,h,t} \mid c_{j,h}, a_{j,h,<t})}{\pi_{\theta_0}(a_{j,h,t} \mid c_{j,h}, a_{j,h,<t})}. \tag{80}$$

In contrast, RTPO synchronizes the latest parameters $\theta_{>k}$ to the inference engine at the start of turn $k$, forks $G-1$ sibling rollouts from the shared boundary state $S_k$, and samples each sibling rollout $j \in \{1,\ldots,G-1\}$ using $\pi_{\theta_{>k}}$ from turn $k$ until termination. We write $u_{j,k} \equiv l_{j,k}$ for the sampled turn-$k$ macro-action of sibling $j$, and $R_j$ for its terminal return. The Monte Carlo turn-level Q-value estimate is

$$\hat{Q}_{j,k} = R_j = r_{j,k} + \gamma_H^{\tau_{j,k}} \hat{F}_{j,k}^{\pi_{\theta_{>k}}}, \tag{81}$$

where $\hat{F}_{j,k}^{\pi_{\theta_{>k}}}$ is the sampled downstream continuation value generated under the same current policy $\pi_{\theta_{>k}}$. The corresponding sibling advantage is

$$A_{j,k}^H = \hat{Q}_{j,k} - \hat{V}_k, \qquad \hat{V}_k = \frac{1}{G-1} \sum_{r=1}^{G-1} \hat{Q}_{r,k}. \tag{82}$$

**On-Policy Continuation without Drift**

**Theorem 3 (On-Policy Continuation under Asynchronous Turns).** *Consider turn $k$ in the reverse-order training procedure of RTPO. At the start of turn $k$, RTPO synchronizes the current downstream policy parameters $\theta_{>k}$ to the inference engine, forks $G-1$ sibling rollouts*

*from the shared boundary state $S_k$, and continues each sibling rollout $j$ to termination under the same policy $\pi_{\theta_{>k}}$. Then the following two properties hold:*

*(a)* ***Drift-free on-policy continuation.*** *Each sibling's terminal return $R_j$ is an unbiased Monte Carlo estimate of the current-policy turn-level Q-value:*

$$\mathbb{E}\Big[\hat{Q}_{j,k} \mid S_k, u_{j,k}\Big] = \tilde{Q}_k^{\pi_{\theta_{>k}}}(S_k, u_{j,k}). \tag{83}$$

*Moreover, because the sampling policy and the evaluation policy coincide throughout the sibling continuation, the trajectory-level importance-sampling weight satisfies*

$$\omega_j = \prod_{h=k+1}^{K-1} \prod_{t=1}^{T_{j,h}} \frac{\pi_{\theta_{>k}}(a_{j,h,t} \mid c_{j,h}, a_{j,h,<t})}{\pi_{\theta_{>k}}(a_{j,h,t} \mid c_{j,h}, a_{j,h,<t})} \equiv 1. \tag{84}$$

*(b)* ***Dynamic error reduction in advantage estimation.*** *Under bounded binary or normalized rewards, the conditional variance of the Monte Carlo Q-value estimator is controlled by the current-policy success probability. In the binary case, if*

$$p_{j,k} := \tilde{Q}_k^{\pi_{\theta_{>k}}}(S_k, u_{j,k}) \in [0, 1], \tag{85}$$

*then*

$$\mathrm{Var}(\hat{Q}_{j,k} \mid S_k, u_{j,k}) = p_{j,k}(1 - p_{j,k}). \tag{86}$$

*Therefore, when reverse-order training improves downstream policies so that $p_{j,k}$ moves away from the high-uncertainty region around $1/2$, the Q-value estimation variance decreases. Since $A^H_{j,k}$ is computed from the sibling Q-value estimates, this improves the signal-to-noise ratio of the resulting turn-level advantage estimator.*

*Proof.* We prove parts (a) and (b) of Theorem 3 as follows.

**Proof of Theorem 3(a): drift-free on-policy continuation.** Fix a turn $k$ and a sibling rollout $j$. At the start of turn $k$, RTPO synchronizes the current downstream policy parameters $\theta_{>k}$ to the inference engine. The sibling rollout is forked from the shared boundary state $S_k$, samples the turn-$k$ macro-action $u_{j,k} \equiv l_{j,k}$, and then continues through turns $k+1, \ldots, K-1$ under the same policy $\pi_{\theta_{>k}}$.

By the definition of the turn-level augmented action value,

$$\tilde{Q}_k^{\pi_{\theta_{>k}}}(S_k, u_{j,k}) = \mathbb{E}\Big[r_{j,k} + \gamma_H^{\tau_{j,k}} F_k^{\pi_{\theta_{>k}}}(S_{j,k+1}) \;\Big|\; S_k, u_{j,k}\Big], \tag{87}$$

where $S_{j,k+1}$ is the next boundary state reached by sibling $j$ after executing $u_{j,k}$. Since the sampled downstream continuation $\hat{F}_{j,k}^{\pi_{\theta_{>k}}}$ is generated by rolling out the same current policy $\pi_{\theta_{>k}}$ from $S_{j,k+1}$ to termination, it is an unbiased Monte Carlo draw from the downstream continuation value:

$$\mathbb{E}\Big[\hat{F}_{j,k}^{\pi_{\theta_{>k}}} \mid S_{j,k+1}\Big] = F_k^{\pi_{\theta_{>k}}}(S_{j,k+1}). \tag{88}$$

Substituting Eq. (88) into Eq. (81) gives

$$\begin{aligned}
\mathbb{E}\Big[\hat{Q}_{j,k} \mid S_k, u_{j,k}\Big] &= \mathbb{E}\Big[r_{j,k} + \gamma_H^{\tau_{j,k}} \hat{F}_{j,k}^{\pi_{\theta_{>k}}} \;\Big|\; S_k, u_{j,k}\Big] \\
&= \mathbb{E}\Big[r_{j,k} + \gamma_H^{\tau_{j,k}} F_k^{\pi_{\theta_{>k}}}(S_{j,k+1}) \;\Big|\; S_k, u_{j,k}\Big] \\
&= \tilde{Q}_k^{\pi_{\theta_{>k}}}(S_k, u_{j,k}).
\end{aligned} \tag{89}$$

Therefore, each sibling terminal return provides an unbiased Monte Carlo estimate of the turn-level Q-value under the current downstream policy.

Next, because the sibling is both sampled and evaluated under the same policy $\pi_{\theta_{>k}}$, the full trajectory-level importance-sampling weight is

$$\omega_j = \prod_{h=k+1}^{K-1} \prod_{t=1}^{T_{j,h}} \frac{\pi_{\theta_{>k}}(a_{j,h,t} \mid c_{j,h}, a_{j,h,<t})}{\pi_{\theta_{>k}}(a_{j,h,t} \mid c_{j,h}, a_{j,h,<t})} = 1. \tag{90}$$

Thus, no trajectory-level IS correction is required. This proves part (a) of Theorem 3.

**Proof of Theorem 3(b): dynamic error reduction in advantage estimation.** We first consider the binary-reward case, where $R_j \in \{0, 1\}$. Conditional on $(S_k, u_{j,k})$, define

$$p_{j,k} = \mathbb{P}_{\pi_{\theta_{>k}}}(R_j = 1 \mid S_k, u_{j,k}) = \tilde{Q}_k^{\pi_{\theta_{>k}}}(S_k, u_{j,k}). \tag{91}$$

Then $\hat{Q}_{j,k} = R_j$ is a Bernoulli random variable with mean $p_{j,k}$, and therefore

$$\mathrm{Var}(\hat{Q}_{j,k} \mid S_k, u_{j,k}) = p_{j,k}(1 - p_{j,k}). \tag{92}$$

The function $f(p) = p(1-p)$ is maximized at $p = 1/2$ and decreases as $p$ moves toward either 0 or 1. Equivalently,

$$p(1-p) = \frac{1}{4} - \left(p - \frac{1}{2}\right)^2. \tag{93}$$

Therefore, if reverse-order training improves the downstream continuation policy so that the induced success probability moves away from the high-uncertainty region around $1/2$, then the conditional variance of the Q-value estimator decreases. Formally,

$$\left|p_{j,k}^{\mathrm{new}} - \frac{1}{2}\right| > \left|p_{j,k}^{\mathrm{old}} - \frac{1}{2}\right| \quad \Longrightarrow \quad p_{j,k}^{\mathrm{new}}(1 - p_{j,k}^{\mathrm{new}}) < p_{j,k}^{\mathrm{old}}(1 - p_{j,k}^{\mathrm{old}}). \tag{94}$$

For normalized rewards $R_j \in [0, 1]$, the same statement holds as a bounded-variance control rather than an exact Bernoulli identity. In particular, by the Bhatia–Davis bound for random variables supported on $[0, 1]$,

$$\mathrm{Var}(R_j \mid S_k, u_{j,k}) \le \mathbb{E}[R_j \mid S_k, u_{j,k}]\big(1 - \mathbb{E}[R_j \mid S_k, u_{j,k}]\big). \tag{95}$$

Thus, moving the conditional mean away from the high-uncertainty middle region also reduces the worst-case variance bound for normalized returns.

Finally, RTPO computes the turn-level advantage by subtracting the sibling baseline:

$$A_{j,k}^H = \hat{Q}_{j,k} - \hat{V}_k, \qquad \hat{V}_k = \frac{1}{G-1}\sum_{r=1}^{G-1} \hat{Q}_{r,k}. \tag{96}$$

Since $A_{j,k}^H$ is a centered function of the sibling Q-value estimates, reducing the Monte Carlo noise in $\hat{Q}_{j,k}$ reduces the noise entering the advantage estimator. Consequently, as reverse-order training improves downstream policies and the on-policy continuation values become more confident, the signal-to-noise ratio of $A_{j,k}^H$ improves. This proves part (b) of Theorem 3, and completes the proof of Theorem 3. □

**Comparison with stale trajectory-level IS correction.** The drift-free property above should be contrasted with a stale-rollout alternative that reuses continuations generated under $\pi_{\theta_0}$ and then applies the trajectory-level IS weight $\omega_j^{\mathrm{old}}$ in Eq. (80). Let

$$N_j = \sum_{h=k+1}^{K-1} T_{j,h} \tag{97}$$

denote the number of tokens in the continuation from turn $k$ to termination. If the per-token divergence between the current policy and the stale rollout policy is nonzero along this continuation, the variance of the full product IS weight can grow multiplicatively with $N_j$. To see this, suppose that for each token position $(h, t)$, the conditional second moment of the token ratio

$$\rho_{j,h,t}^{\mathrm{old}} = \frac{\pi_{\theta_{>k}}(a_{j,h,t} \mid c_{j,h}, a_{j,h,<t})}{\pi_{\theta_0}(a_{j,h,t} \mid c_{j,h}, a_{j,h,<t})} \tag{98}$$

satisfies

$$\mathbb{E}_{\pi_{\theta_0}}\left[\left(\rho_{j,h,t}^{\mathrm{old}}\right)^2 \mid c_{j,h}, a_{j,h,<t}\right] \ge 1 + \delta \qquad \text{for some } \delta > 0. \tag{99}$$

Then the second moment of the product weight scales as

$$\mathbb{E}_{\pi_{\theta_0}}\left[\left(\omega_j^{\mathrm{old}}\right)^2\right] = \mathbb{E}_{\pi_{\theta_0}}\left[\prod_{h=k+1}^{K-1}\prod_{t=1}^{T_{j,h}}\left(\rho_{j,h,t}^{\mathrm{old}}\right)^2\right] \gtrsim (1+\delta)^{N_j}, \tag{100}$$

up to the usual conditioning on autoregressive histories. Since $\mathbb{E}_{\pi_{\theta_0}}[\omega_j^{\mathrm{old}}] = 1$, this implies

$$\mathrm{Var}_{\pi_{\theta_0}}(\omega_j^{\mathrm{old}}) = \mathbb{E}_{\pi_{\theta_0}}\left[\left(\omega_j^{\mathrm{old}}\right)^2\right] - 1 \gtrsim (1+\delta)^{N_j} - 1. \tag{101}$$

This illustrates the long-horizon instability of trajectory-level IS correction: even a small nonzero per-token policy mismatch can compound into a high-variance product over many tokens. Per-token PPO clipping does not remove this mismatch at the trajectory level, because clipping and multiplication do not commute:

$$\prod_{h,t}\mathrm{clip}\left(\rho_{j,h,t}^{\mathrm{old}}, 1-\epsilon, 1+\epsilon\right) \neq \mathrm{clip}\left(\prod_{h,t}\rho_{j,h,t}^{\mathrm{old}}, 1-\epsilon, 1+\epsilon\right). \tag{102}$$

Therefore, stale-rollout correction remains fundamentally different from RTPO's on-policy sibling continuation, where the corresponding weight is $\omega_j \equiv 1$.

**Interpretation.** Theorem 3 shows why RTPO regenerates sibling continuations on-policy instead of reusing stale downstream rollouts. In asynchronous multi-turn training, stale continuations estimate values under outdated downstream policies and would require a long-horizon trajectory-level IS correction. RTPO avoids this by synchronizing $\theta_{>k}$ before sibling generation and rolling out each sibling to termination under the same current policy. As a result, the full trajectory IS weight is one, the Q-value estimate targets the current downstream policy, and the resulting turn-level advantage avoids policy-drift contamination.

## D Experimental Setup

**Base Models.** We use Qwen3 models (Yang et al., 2025a) as the backbone for RTPO and all baselines. These models support context lengths of up to 32,768 tokens, which is sufficient to accommodate the multi-turn interaction histories required by long-horizon agentic RL tasks. Since Qwen3 natively supports a thinking inference mode, we enable this mode consistently across all multi-turn agentic RL experiments to ensure that the model generates complete tool-integrated reasoning trajectories during rollout. We use Qwen3-8B as the main backbone because search-based agentic tasks require strong base-model capabilities for multi-turn evidence acquisition, information integration, and tool-augmented reasoning. This choice is supported by our main results in Sec. 4.1, the rollout–training consistency analysis in Sec. 4.2, and the policy-drift analysis in Sec. 4.4. To isolate the effect of turn-level credit assignment in Sec. 4.3, we use the smaller Qwen3-4B model for mathematical reasoning tasks. This is because credit assignment in these tasks depends more directly on the model's intrinsic reasoning ability than on external knowledge retrieval, making the gains from turn-level credit assignment more discernible.

**Baselines.** We compare RTPO with two categories of state-of-the-art multi-turn agentic RL methods. The first category consists of trajectory-only policy optimization methods, including GRPO (Shao et al., 2024) and. GRPO uses token-level importance ratios with group-relative advantage estimation. The second category includes turn-level and tree-based policy optimization methods, including ARPO (Dong et al., 2026), Tree-GRPO (Ji et al., 2026), and SeeUPO (Hu et al., 2026). ARPO performs entropy-driven adaptive branching at uncertain tool-call nodes and estimates advantages separately for shared-prefix and branch-specific tokens. Tree-GRPO represents multi-turn agent interaction as a tree and constructs group-relative advantages at both intra-tree and inter-tree levels by sharing prefixes. SeeUPO treats each turn as an independent agent and performs sequential per-turn updates under a heterogeneous multi-agent learning, with a theoretical guarantee of monotonic improvement.

**Datasets.** To ensure a fair comparison across methods in multi-turn agentic scenarios, we consider two representative tool-use tasks: mathematical reasoning and knowledge reasoning from search.

Both require agents to interact with external tools, perform multi-turn reasoning, and adapt their actions based on intermediate feedback.

For the mathematical reasoning task, we use the MATH dataset (Hendrycks et al., 2021) for training, which covers challenging multi-step reasoning problems spanning algebra, geometry, number theory, probability, and other topics. The task requires agents to decompose complex problems, invoke external Python computation tools when necessary, and integrate intermediate results into the final answer, where successful solutions often depend on iterative calculation, verification, and correction. At the evaluation stage, we test generalization at two difficulty levels: (1) standard mathematical reasoning benchmarks, including GSM8K (Cobbe et al., 2021) and MATH-500 (Lightman et al., 2024); and (2) competition-level benchmarks, including AMC'23 (Mathematical Association of America, 2023), AIME'24 (Mathematical Association of America, 2024), AIME'25 (Mathematical Association of America, 2025), and OE-Math (He et al., 2024), which feature problems that demand extended chains of tool-augmented reasoning and advanced problem-solving capabilities. Since no training data are available for the competition-level benchmarks, all evaluations are conducted in a zero-shot setting.

For the knowledge reasoning task on web search, we adopt the hard-search training set constructed by ARPO (Dong et al., 2026), consisting of 1,000 high-difficulty search samples drawn from two open-source deep-search data sources: SimpleDeepSearcher (Sun et al., 2025) and WebSailor (Li et al., 2025). These samples require extensive web retrieval, multi-source evidence integration, long-context reasoning, and frequent tool calls, providing a rigorous testbed for evaluating the model stability and sample efficiency of multi-turn agentic RL. On this task, we evaluate knowledge-intensive multi-hop question answering on HotpotQA (Yang et al., 2018) and 2WikiMultiHopQA (Ho et al., 2020), following the ARPO evaluation protocol (Dong et al., 2026) with LLM-as-Judge scoring based on Qwen2.5-72B-Instruct and report F1 scores. In addition, to examine the effect of on-policy continuation, we compare default RTPO with its off-policy variant, and conduct additional evaluation on four challenging deep-search benchmarks: GAIA (Mialon et al., 2024), which evaluates general AI-assistant capabilities across three difficulty levels (Lv.1–Lv.3); WebWalkerQA (Wu et al., 2025a), which focuses on interactive web navigation and multi-hop question answering; Humanity's Last Exam (Center for AI Safety et al., 2026), which covers expert-level problems in sciences, engineering, and humanities; and xBench (Chen et al., 2025), which evaluates cross-lingual deep-search capability. Since the training data are drawn exclusively from the hard-search source, all four benchmarks serve as held-out evaluation sets. We report Pass@1 with sampling temperature set to $0.6$ and top-$p$ set to $0.95$.

**Configurations.** All methods are implemented on top of the VeRL framework (Sheng et al., 2025), with vLLM (Kwon et al., 2023) for rollout generation and FSDP (Zhao et al., 2023) for post-training. All experiments are conducted on $8\times$ NVIDIA A100 GPUs. Unless otherwise specified, the training batch size is $64$ and the maximum single-turn response length is $4096$ tokens. The maximum interaction horizon is set according to task type: $K = 3$ turns for mathematical reasoning and $K = 6$ turns for web search. The learning rate is fixed at $1 \times 10^{-6}$ with the AdamW (Loshchilov & Hutter, 2019) optimizer and a weight decay of $0.01$. For branching-based baselines (ARPO and TreeGRPO), we follow the hyperparameters recommended in their original papers, including an entropy threshold of $0.4$ and an initial sampling size of $8$. Full hyperparameter configurations are provided in Appendix E.3.

**Evaluation Metrics.** We evaluate RTPO and all baselines along multiple dimensions. For overall performance, we use Pass@1 accuracy as the primary indicator of final policy quality after optimization (Sec 4.1). We also measure the average number of tool calls to assess behavioral differences across methods under a fixed compute budget. For training stability, rollout–training consistency is evaluated using the log-probability ratio and KL divergence, as described in Sec. 4.2. We further conduct controlled experiments to examine whether RTPO's turn-level credit assignment (Sec 4.3) and on-policy continuation (Sec 4.4) are consistent with the theoretical analysis. Each experiment is repeated three times to mitigate randomness, and we report the average performance.

# E Implementation Details

## E.1 Pseudocode

The pseudocode for our proposed RTPO is shown in Algorithm 1.

**Algorithm 1** Reverse-Turn Policy Optimization (RTPO)

**Require:** Query dataset $\mathcal{D}$; policy $\pi_\theta$; inference engine $\mathcal{E}$; sibling count $G$; PPO clip $\epsilon$; max turns $K$; observation map $\psi$; PPO epochs $E$.

**Ensure:** Updated parameters $\theta$.

**Phase 1: Trunk Rollout**

For each query $q \in \mathcal{D}$, generate a complete trajectory $(S_0, u_0, f_0, \ldots, u_{K-1}, f_{K-1})$ under $\pi_\theta$ by interacting with the environment. Record boundary states and environment snapshots $\{S_k, \mathrm{snap}_k\}_{k=0}^{K-1}$, and set conditioning contexts $c_k = \psi(S_k)$. The trunk provides anchors only and receives no gradient.

**Phase 2: Reverse-Order Training**

**for** $k = K{-}1,\ K{-}2,\ \ldots,\ 0$ **do**

**Synchronize.** Push current parameters $\theta$ (denoted $\theta_{>k}$, after the downstream-turn updates) to $\mathcal{E}$; freeze $\theta_{\mathrm{old}} \leftarrow \theta_{>k}$ as the IS denominator.

**Sibling generation (on-policy).** For each $q \in \mathcal{D}$ and each sibling $j = 1, \ldots, G{-}1$: restore environment from $\mathrm{snap}_k$; sample turn-$k$ response $u_{j,k} \sim \pi_{\theta_{>k}}(\cdot \mid c_k)$; continue under $\pi_{\theta_{>k}}$ through turns $k{+}1, \ldots, K{-}1$ to terminal; receive reward $R_j \in \{0, 1\}$. Record turn-$k$ tokens $\{a_{j,k,t}\}_{t=1}^{T_{j,k}}$ and their log-probs under $\theta_{\mathrm{old}}$.

**Turn-level advantage.** For each $q \in \mathcal{D}$, compute the sibling baseline and advantage:

$$\hat{V}_k = \frac{1}{G{-}1} \sum_{j=1}^{G-1} R_j, \qquad A^H_{j,k} = R_j - \hat{V}_k.$$

All siblings share $S_k$, so $A^H_{j,k}$ is free of upstream state contamination (Theorem 2).

**PPO update (turn-$k$ tokens only).**

**for** epoch $= 1, \ldots, E$ **do**

**for** each mini-batch from sibling turn-$k$ tokens **do**

Compute IS ratios $\rho_{j,k,t} = \pi_\theta(a_{j,k,t} \mid c_k, a_{j,k,<t}) \,/\, \pi_{\theta_{\mathrm{old}}}(a_{j,k,t} \mid c_k, a_{j,k,<t})$.

Update $\theta$ via the clipped objective:

$$J_k(\theta) = \frac{1}{G{-}1} \sum_{j=1}^{G-1} \frac{1}{T_{j,k}} \sum_{t=1}^{T_{j,k}} \min\Big( \rho_{j,k,t}\, A^H_{j,k},\ \mathrm{clip}\big(\rho_{j,k,t},\, 1{-}\epsilon,\, 1{+}\epsilon\big)\, A^H_{j,k} \Big).$$

**end for**

**end for**

▷ *Turn $k$ complete; next turn inherits updated $\theta$.*

**end for**

## E.2 Source Code

We provide the source code of RTPO in the **Supplementary Material**. The repository includes complete installation instructions and versioned dependency requirements. Our implementation is built upon VeRL (Sheng et al., 2025), whose core additions are: (1) summarizing the output of each round, so that the next round can rollout without having access to the complete history, (2) on-policy tree rollouts, and (3) computation of per-turn advantage and reverse updates. We also provide out-of-the-box training and evaluation scripts to reproduce our main results on mathematical and knowledge reasoning tasks.

## E.3 Training Details

We provide the detailed hyperparameters for all experiments in Table 4. Unless otherwise noted, the maximum prompt/response lengths are kept the same across all experiments. For computational

fairness, the global sampling budget is set to 80k rollouts for mathematical reasoning (5k prompts $\times$ 16 rollouts per prompt) and 16k for knowledge reasoning (1k prompts $\times$ 16 rollouts per prompt). For the Qwen3 series, we use a sampling temperature of 0.9, the AdamW optimizer with a learning rate of $1 \times 10^{-6}$, weight decay 0.01, and an initial KL coefficient of 0.02. The log-probability clipping range is set to $[-10, 10]$. The training batch size and the rollout batch size are both set to 64. The maximum prompt length is 8192 tokens and the maximum single-turn response length is 4096 tokens for both tasks. The maximum interaction horizon is set to $K = 3$ turns for mathematical reasoning and $K = 6$ turns for web search. All methods are allocated a uniform budget of 16 rollouts per prompt (see Table 4).

Table 4: Hyperparameter settings for all experiments.

| Method | Parameters | Total Rollouts | Structure |
|---|---|---|---|
| GRPO | `n_agent`= 16 | 16 independent chains | Chain |
| SeeUPO | `rollout_n`= 16 | 16 chains + turn-level update | Chain |
| ARPO | $N$=8, $M$=16 | 8 initial + entropy branch to 16 | Entropy tree |
| TreeGRPO | $M$=4, $N$=3, $L$=1 | 4 trees $\times$ 4 leaves = 16 | Random tree |
| RTPO | sibling rollouts | Global Max budget | State-matched tree |

For GRPO, we sample 16 independent chains per prompt. For SeeUPO, we generate 16 independent chains with sequential turn-level updates in reverse execution order. For ARPO, we set the initial sampling size to $N = 8$ and the global rollout budget to $M = 16$, with entropy weight $\beta = 0.2$, base probability $\alpha = 0.5$, and branching threshold $\tau = 0.5$; rollouts that do not trigger entropy-driven branching are supplemented with independent chains until the budget of 16 is reached. For TreeGRPO, we set the number of initial trees to $M = 4$, the number of expansion nodes per iteration to $N = 3$, and the number of expansion iterations to $L = 1$, yielding $M \times (L \times N + 1) = 16$ rollouts per prompt via random node expansion. For RTPO, we first generate multiple trunk trajectories per prompt to establish turn-boundary states: 4 trunks for mathematical reasoning ($K$=3) and 2 trunks for web search ($K$=6), since in practice most trunks do not reach the maximum interaction horizon and terminate early with fewer tool calls. At each realized turn boundary $S_k$, $G-1 = 2$ sibling rollouts are forked and continued to termination.

To ensure computational fairness with baselines, RTPO enforces the same global sampling budget as all other methods (80k for mathematical reasoning, 16k for knowledge reasoning), so that the total number of generated rollouts across all prompts does not exceed that of any baseline. Per-prompt rollout counts may vary depending on early termination, but the global budget constraint guarantees that RTPO consumes no more rollout compute than the 16-chain baselines in aggregate. All methods use a discount factor of $\gamma = 1$.

All experiments are conducted on a single node equipped with 8$\times$ NVIDIA A100 80GB GPUs, 256 CPU cores, and 256GB of system memory, running Ubuntu 22.04 LTS. The math task requires approximately 15 hours of training, whereas the knowledge task requires approximately 5 hours.

### E.4 QWEN3 CHAT TEMPLATE

At each interaction turn, Qwen3 receives the current context and generates three components: (A) an internal reasoning trace enclosed by `<think>...</think>`, (B) textual commentary after `</think>`, and (C) a tool-call instruction, e.g., `<tool_call>...</tool_call>`. The tool call is executed by the external environment, which returns the result as an observation. Before constructing the next-turn input, the chat-template filter removes the `<think>...</think>` segment, while retaining the textual commentary, the tool-call instruction, and the appended tool-execution result. Thus, the input to the second turn consists of: (1) the original user request, (2) the previous textual commentary, (3) the previous tool call, and (4) the tool result. This filtered context enables the model to condition on the action history and environment feedback without carrying redundant internal reasoning traces, keeping the context length manageable while preserving decision-relevant information. The process is illustrated in Figure 4.

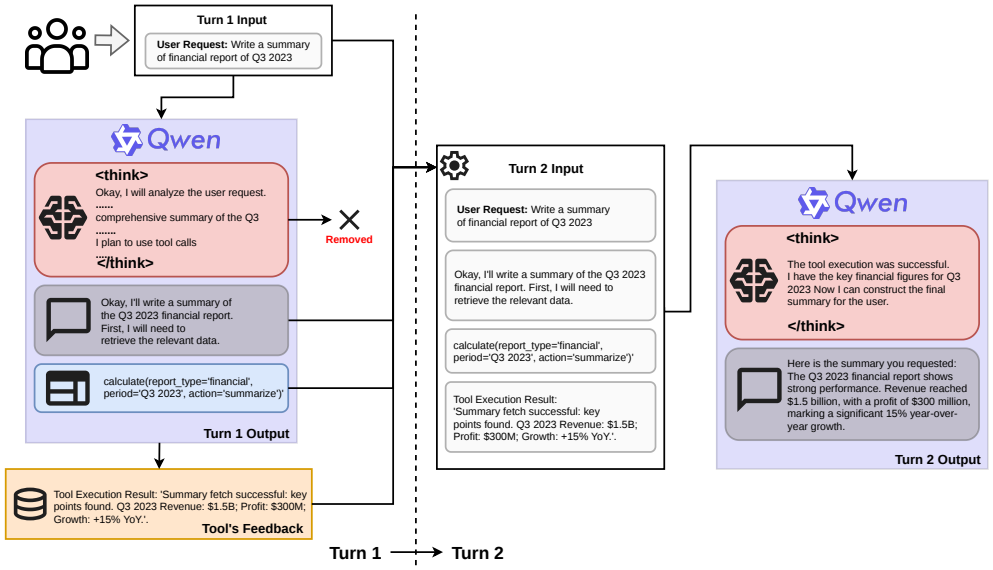



Figure 4: Examples of Qwen3 chat template.

# F Additional Results and Insights

## F.1 Discussion of Main Results

For the main experiments, we use Qwen3-8B as the base model and evaluate on two categories of multi-turn tool-use tasks: mathematical reasoning and web search. Qwen3 models automatically produce chain-of-thought content wrapped in `<think>...</think>` tags during inference. In our setup, each turn's input is constructed using the standard Qwen3 chat template, which post-processes prior assistant content by stripping out the `<think>...</think>` segments so that subsequent turns observe only the final answer content following `</think>`. This design keeps the per-turn input context concise and aligns naturally with the turn-boundary MDP formulation underlying RTPO: each turn-level policy is conditioned on the visible context $c_k$ rather than on the full raw interaction history. In addition, we report the complete accuracy results with mean and standard deviation, extending Table 1 to Table 5.

Table 5: Accuracy comparison across eight benchmarks. Values are reported as mean$_{\pm\text{std}}$ over three evaluation runs. *Acc* denotes Pass@1 (%) for mathematical tasks and best-span F1 (%) for knowledge reasoning tasks. Standard deviations are reported in percentage points.

| Method | M: Mathematical Reasoning | | | | | | K: Knowledge Reasoning | |
|---|---|---|---|---|---|---|---|---|
| | AIME24 | AIME25 | AMC23 | MATH500 | GSM8K | OE-Math | HotpotQA | 2Wiki |
| Vanilla | $3.33_{\pm 0.00\%}$ | $13.33_{\pm 3.33\%}$ | $12.50_{\pm 2.50\%}$ | $51.00_{\pm 0.40\%}$ | $76.95_{\pm 0.15\%}$ | $24.33_{\pm 0.68\%}$ | $\underline{54.52}_{\pm 0.72\%}$ | $59.83_{\pm 0.61\%}$ |
| GRPO | $10.00_{\pm 3.33\%}$ | $20.00_{\pm 3.33\%}$ | $57.50_{\pm 2.50\%}$ | $82.33_{\pm 0.50\%}$ | $93.86_{\pm 0.23\%}$ | $50.74_{\pm 1.12\%}$ | $53.39_{\pm 0.93\%}$ | $61.85_{\pm 0.84\%}$ |
| SeeUPO | $\underline{20.00}_{\pm 3.33\%}$ | $\underline{23.33}_{\pm 3.33\%}$ | $\underline{67.50}_{\pm 2.50\%}$ | $\underline{84.20}_{\pm 0.35\%}$ | $\underline{94.47}_{\pm 0.13\%}$ | $\underline{53.86}_{\pm 0.45\%}$ | $54.27_{\pm 0.55\%}$ | $\underline{63.78}_{\pm 0.49\%}$ |
| RTPO | $\mathbf{33.33}_{\pm 0.00\%}$ | $\mathbf{26.67}_{\pm 0.00\%}$ | $\mathbf{67.50}_{\pm 0.00\%}$ | $\mathbf{86.20}_{\pm 0.20\%}$ | $\mathbf{94.84}_{\pm 0.08\%}$ | $\mathbf{55.49}_{\pm 0.30\%}$ | $\mathbf{64.89}_{\pm 0.34\%}$ | $\mathbf{64.35}_{\pm 0.27\%}$ |

**Performance and tool-use patterns on mathematical reasoning tasks.** On mathematical tasks, tool-call frequency exhibits a trend that runs almost opposite to accuracy: Vanilla makes 17 Python calls on AIME24 yet solves only one problem, whereas RTPO solves ten with just four calls. This pattern reflects an intrinsic property of mathematical reasoning: the knowledge and derivations required to solve a math problem are primarily internalized in the model's parameters, while the external Python tool serves as an auxiliary aid for verification and numerical computation rather than as a source of new information. Vanilla's frequent tool use, therefore, largely reflects an inefficient strategy of *repeated verification used to mask reasoning uncertainty*. After RL training, the model gains stronger control over its own reasoning process, and tool calls collapse from redundant repeated verification into precise invocations at critical computation steps; consequently, the number of calls decreases while accuracy increases. RTPO's advantage becomes particularly pronounced on the hardest math benchmarks: it reaches 33.33% Pass@1 on AIME24, surpassing SeeUPO (20.00%) and GRPO (10.00%) by 13.3 and 23.3 absolute points, respectively, and maintains a clear lead on AIME25 and OE-Math. RTPO simultaneously achieves the fewest tool calls and the highest accuracy on these hard problems, indicating that it learns the most efficient tool-use strategy. We provide an example to support our discussion, as shown in Figure 5.

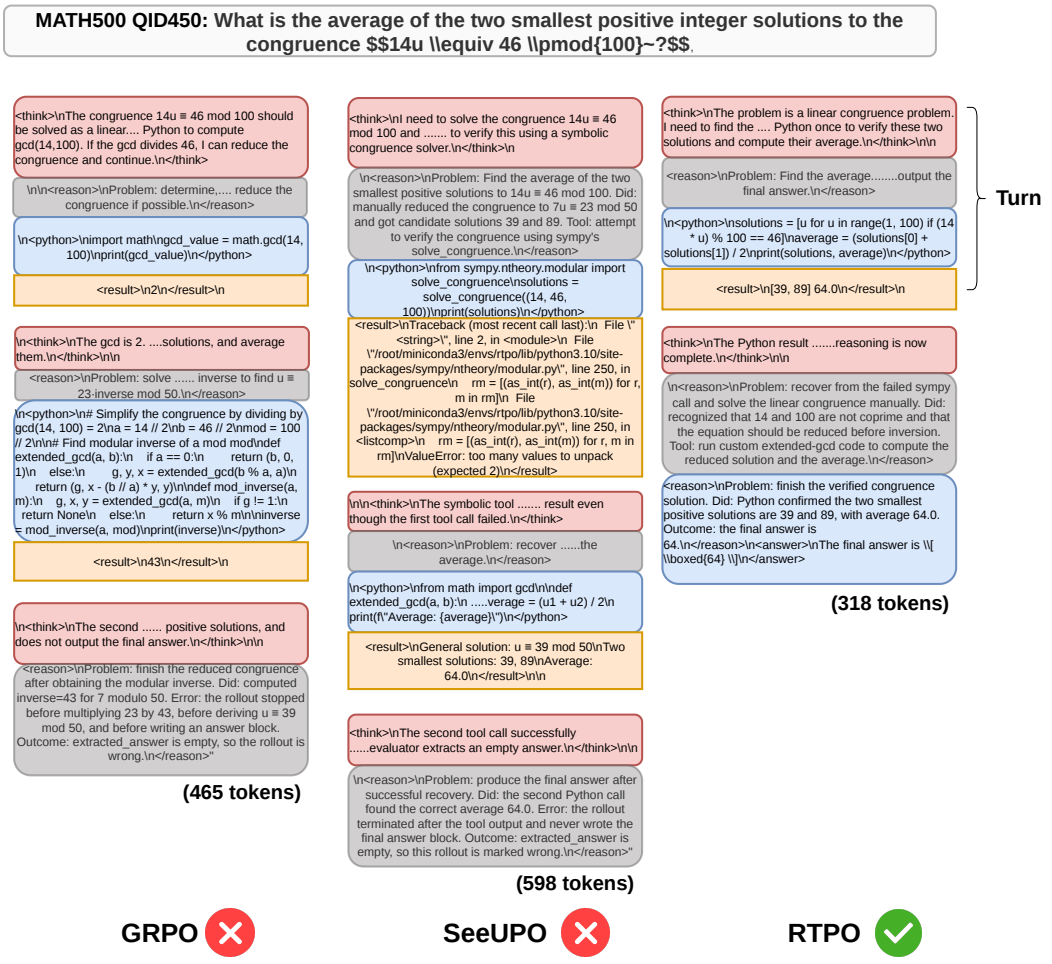



Figure 5: An example of different policy performance on a mathematical-reasoning task through Python tools.

**Performance and tool-use patterns on knowledge reasoning tasks.** On knowledge-intensive question answering, RTPO reaches 64.89 F1 on HotpotQA, exceeding SeeUPO (54.27) and GRPO (53.39) by 10.6 and 11.5 points, respectively, and maintains a clear lead on 2Wiki. In contrast to mathematical tasks, the tool-use pattern on knowledge reasoning exhibits a positive correlation between call count and accuracy: RTPO issues 613 and 867 search calls on HotpotQA and 2Wiki,

far exceeding the roughly 200 calls observed for the other methods. This contrast admits a natural explanation grounded in task structure: the factual information required to answer such questions does not reside in the model's parametric knowledge and must be acquired through external retrieval, while multi-hop questions further demand cross-turn integration of multiple pieces of evidence. The lower retrieval counts of GRPO and SeeUPO indicate that they fail to learn to issue sustained follow-up queries and to expand retrieval across multiple turns. The underlying cause is that such hard tasks require longer interaction horizons and more frequent reasoning revision, so the rollout–training context mismatch is amplified as turns accumulate, and baseline methods must spend a larger share of their optimization budget compensating for this drift. RTPO eliminates this source of bias structurally, allowing the entire optimization budget to act directly on the task objective, which yields more stable improvements on long-horizon hard tasks. We provide an example to support our discussion, as shown in Figure 6.

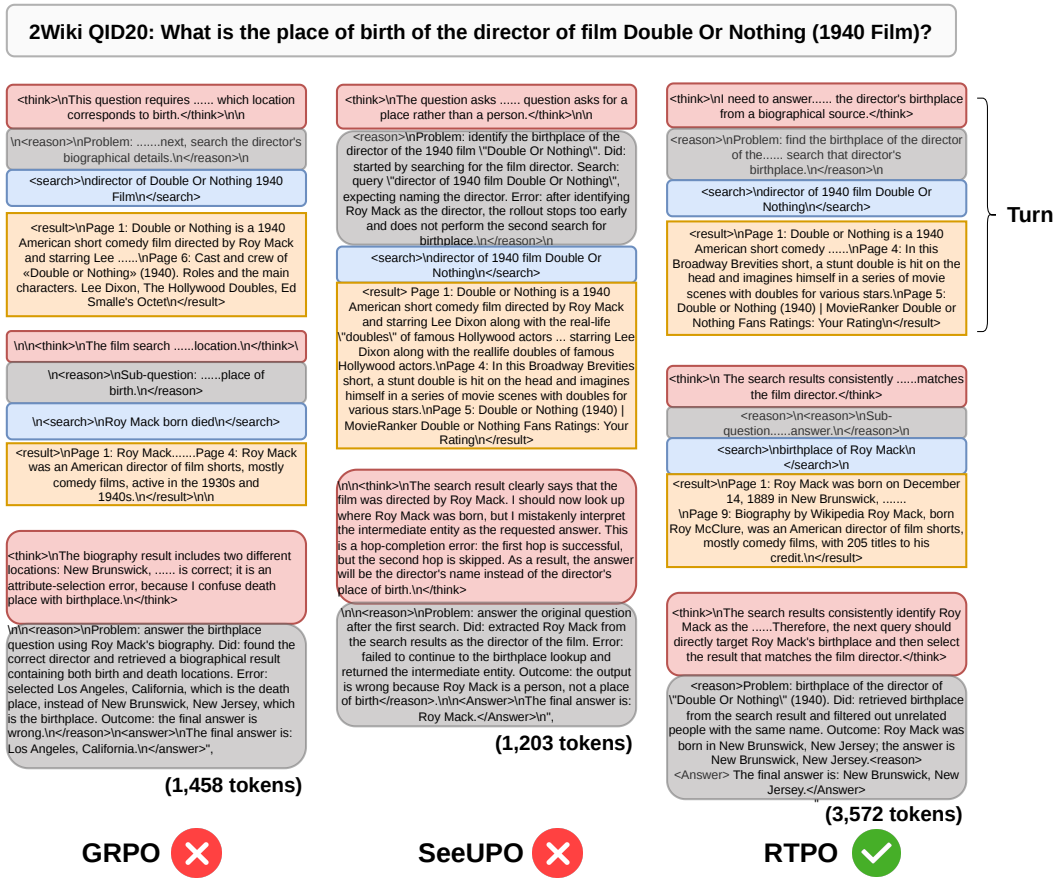



Figure 6: An example of different policy performance on a knowledge-reasoning task through web search tools.

Taking both task categories together, RTPO learns to perform mathematical reasoning with *fewer calls but more reliable internal reasoning*, while performing knowledge reasoning with *denser calls and more thorough external information integration*. This bidirectional adaptation of tool-use behavior indicates that the advantage of RTPO does not stem from simply encouraging or suppressing tool calls, but rather from its turn-level optimization objective, which is able to learn a tool-use strategy matched to the underlying nature of each task.

### F.2 Insights from Rollout–Training Consistency Analysis

Follow-up discussion from Sec 4.2, a noteworthy observation is that, despite using different conditioning contexts at the rollout and training stages, the baselines' ratios still exhibit a slow drift toward 1. We find that this phenomenon is closely related to the mechanism revealed by DAgger (Ross

et al., 2011): when the training data is continually drawn from the distribution induced by the policy itself, the model gradually adapts to the distribution it actually operates on. In our setting, trajectories generated by summary-rollouts are reinforced under full-history-training, and parameter sharing causes the behaviors under the two conditionings to converge indirectly; the next rollout therefore falls closer to the region considered reasonable under the training context. SCoRe (Kumar et al., 2025) observes the same bootstrapped alignment process in multi-turn online RL. In addition, some works point out that the summary can serve as a learnable sufficient statistic: if the summary preserves the decision-relevant information, the optimal policies under the two contexts can converge in an information-theoretic sense (Allen et al., 2021); recent work on multi-turn RL further shows that the summary context can evolve into a learnable compact decision state under end-to-end optimization (Lu et al., 2026; Yu et al., 2026).

However, this empirical alignment differs from the structural consistency of RTPO in three fundamental ways:

(1) Alignment is incomplete and strongly depends on task complexity. GRPO recovers only to $0.97$ on math after $30$ steps and only to $0.83$ on search after $14$ steps; the longer the horizon and the faster the context accumulates, the harder the alignment becomes, which is reflected in the knowledge reasoning task in Table 1.

(2)The alignment process is accompanied by oscillation. SeeUPO produces a $1.02$ spike at step $9$ on math and exhibits sustained small fluctuations, because the bootstrapped loop is driven by the advantage signal, and the variance of advantage estimates propagates directly into step-to-step jitter of the ratio. RTPO's ratio is structurally guaranteed and is only affected by differences between the training and inference engines.

(3) Empirical alignment consumes additional optimization budget. Bootstrapped alignment essentially allocates part of the policy's capacity to an implicit objective—pulling the behavior under the summary toward the optimal behavior under the full history. RTPO removes this hidden cost, so that the optimization budget can act directly on the task objective. This is consistent with the lower search-call counts of baselines on long-horizon knowledge reasoning observed in Table 1: part of their training dynamics is diverted to *patching the mismatch*.

Overall, the rollout–training ratio reveals not that baselines necessarily fail, but that baselines must rely on training dynamics to compensate for a gap that RTPO does not have by construction, and this compensatory mechanism becomes substantially less effective on long-horizon tasks.

### F.3 Additional Findings for Hard-Search Scenarios

To quantify the independent contribution of on-policy continuation (Sec. 3.3), we use Qwen3-8B as the base model and compare standard RTPO with its off-policy variant. Both share the same reverse-order turn-level training and state-matched sibling structure; the only difference lies in how sibling downstream continuations are obtained. RTPO (on-policy) synchronizes the latest parameters $\theta_{>k}$ to the inference engine at the start of turn $k$ and regenerates sibling continuations under $\pi_{\theta_{>k}}$ until termination, so the trajectory-level IS weight is identically one, and the Q-value estimate is

$$\hat{Q}^{\text{on}}_{j,k} = r_{j,k} + \gamma^{\tau_{j,k}}\, \hat{F}^{\pi_{\theta_{>k}}}_{j,k}. \tag{103}$$

RTPO with off-policy variant instead reuses the downstream continuations already generated under $\pi_{\theta_0}$ during the initial rollout stage and corrects for the staleness via a clamped trajectory-level importance weight:

$$\hat{Q}^{\text{off}}_{j,k} = r_{j,k} + \gamma^{\tau_{j,k}}\, \bar{\omega}_{j,k}\, \hat{F}^{\pi_{\theta_0}}_{j,k}, \qquad \bar{\omega}_{j,k} = \text{clamp}\left( \prod_{h=k+1}^{K-1} \prod_{t=1}^{T_{j,h}} \frac{\pi_{\theta_{>k}}(a_{j,h,t} \mid s_{j,h,t})}{\pi_{\theta_0}(a_{j,h,t} \mid s_{j,h,t})},\ \rho_{\min},\ \rho_{\max} \right). \tag{104}$$

We compare the on-policy RTPO and its off-policy variant on four knowledge-reasoning deep-search benchmarks, as shown in Table 6. We use output-hit accuracy, which marks a sample correct if the full output contains the gold answer or an alias. The benchmarks include GAIA with three difficulty levels, WebWalkerQA, XBench, and HLE with eight subject subsets.

Table 6: Rechecked output-hit comparison between standard RTPO (on-policy) and its off-policy variant. Values are rounded to the nearest integer for better readability. Output-hit marks a sample correct if the output contains the gold answer or an alias.

| **Benchmark** | **QS** | **Off Hit** | **Off Rate** | **On Hit** | **On Rate** | **Δ** |
|---|---|---|---|---|---|---|
| **GAIA** | 103 | 24 | 23.30% | 30 | 29.13% | +5.83% |
| L1 | 39 | 7 | 17.95% | 10 | 25.64% | +7.69% |
| L2 | 52 | 16 | 30.77% | 18 | 34.62% | +3.85% |
| L3 | 12 | 1 | 8.33% | 2 | 16.67% | +8.33% |
| **WebWalker** | 200 | 12 | 6.00% | 19 | 9.50% | +3.50% |
| **XBench** | 100 | 8 | 8.00% | 15 | 15.00% | +7.00% |
| **HLE** | 2096 | 261 | 12.45% | 254 | 12.12% | -0.33% |
| Biology/Medicine | 228 | 37 | 16.23% | 38 | 16.67% | +0.44% |
| Chemistry | 81 | 6 | 7.41% | 6 | 7.41% | +0.00% |
| Computer Science/AI | 220 | 20 | 9.09% | 24 | 10.91% | +1.82% |
| Engineering | 71 | 5 | 7.04% | 8 | 11.27% | +4.23% |
| Humanities/Social Science | 192 | 32 | 16.67% | 23 | 11.98% | -4.69% |
| Math | 947 | 114 | 12.04% | 112 | 11.83% | -0.21% |
| Other | 158 | 29 | 18.35% | 29 | 18.35% | +0.00% |
| Physics | 199 | 18 | 9.05% | 14 | 7.04% | -2.01% |

# G Extended Experiments

## G.1 Stateful Tool-Agent Evaluation on $\tau^3$-Airline

**Experimental setting.** We evaluate Qwen3-1.7B on a fixed set of 30 training tasks and the 20 held-out test tasks provided by the $\tau^3$-Airline environment (Barres et al., 2026). The agent must query and modify an airline database through multi-turn tool interactions, and earlier actions change the state observed in later turns.

Table 7: Online training success rate on $\tau^3$-Airline.

| Method | Step 10 | Step 20 | Step 30 | Step 40 | Step 50 |
|---|---|---|---|---|---|
| GRPO | 23.44% | 25.00% | 31.26% | 29.69% | 31.26% |
| SeeUPO | 31.23% | 9.38% | 14.83% | 0.78% | 0.78% |
| Tree-GRPO | 18.40% | 14.95% | 27.48% | 24.53% | 14.88% |
| REFUEL | 25.00% | 31.25% | 31.25% | 28.12% | 21.88% |
| **RTPO** ($G = 3$) | 15.63% | **37.50%** | **50.00%** | **56.25%** | **40.63%** |

RTPO obtains the highest online success rate from Step 20 onward, reaching 56.25% at Step 40 and 40.63% at Step 50.

Table 8: Held-out evaluation on the 20 $\tau^3$-Airline test tasks. Pass$^4$ means 4 times all pass.

| Method | Pass@1 | Pass@4 | Pass$^4$ | Normal termination | Generation truncation | Avg. response tokens |
|---|---|---|---|---|---|---|
| SeeUPO | 0.00% | 0.00% | 0.00% | 0.00% | 98.75% | 2,023.7 |
| GRPO | 11.25% | 25.00% | 0.00% | 20.00% | 60.00% | 2,469.7 |
| Tree-GRPO | 16.25% | 25.00% | 10.00% | 30.00% | 47.50% | 2,538.2 |
| REFUEL | 11.25% | 20.00% | 0.00% | 18.75% | 57.50% | 2,621.2 |
| **RTPO** ($G = 3$) | **17.50%** | **30.00%** | **10.00%** | **45.00%** | **42.50%** | **1,688.8** |

RTPO achieves the highest Pass@1 and Pass@4 and the highest normal environment-termination rate. Relative to Tree-GRPO, it reduces the generation-truncation rate from 47.50% to 42.50% and the average response length from 2,538.2 to 1,688.8 tokens. Relative to REFUEL (Gao et al., 2025), RTPO improves Pass@1 from 11.25% to 17.50%, improves Pass@4 from 20.00% to 30.00%, and reduces average response length by approximately 35.6%.

Table 9: Tool-execution quality on the held-out $\tau^3$-Airline tasks.

| Method | Attempted calls | Executed successfully | Tool-error calls | Execution success | Response tokens/success |
|---|---|---|---|---|---|
| GRPO | 118 | 68 | 50 | 57.63% | 21,952.9 |
| SeeUPO | 1 | 0 | 1 | 0.00% | N/A |
| Tree-GRPO | 1 | 0 | 1 | 0.00% | 15,619.5 |
| **RTPO** | 83 | **74** | **9** | **89.16%** | **9,650.5** |

RTPO attempts fewer tool calls than GRPO but completes more calls without execution errors. Tool-error calls decrease from 50 to 9, and the execution success rate increases from 57.63% to 89.16%. RTPO also requires approximately 9,650.5 response tokens per successful trajectory, which is 38.2% lower than Tree-GRPO and 56.0% lower than GRPO.

### G.2 Sensitivity to the Sibling Group Size

The group-size hyperparameter is $G$; at each boundary, RTPO forks $G-1$ sibling continuations from the same turn-boundary state $S_k$. The turn-level estimator is

$$\widehat{A}^{H}_{j,k} = R_j - \frac{1}{G-1}\sum_{r=1}^{G-1} R_r. \tag{105}$$

Theorem 2(a) gives

$$\mathbb{E}\Big[\widehat{A}^{H}_{j,k} \mid S_k, u_{j,k}\Big] = \frac{G-2}{G-1} A^{\pi}(S_k, u_{j,k}). \tag{106}$$

Thus, the finite-group bias is of order $O(1/G)$, and the multiplicative coefficient approaches one as $G$ increases. Averaging more sibling returns also reduces sampling noise in the Monte Carlo baseline. Because all continuations begin from the same boundary state, this comparison does not reintroduce upstream-state contamination.

Table 10: Effect of sibling group size on signal density and held-out performance in $\tau^3$-Airline.

| $G$ | Theoretical coefficient | Pass@1 | Pass@4 | Zero-advantage rate | Tokens/success |
|---|---|---|---|---|---|
| 3 | $1/2$ | 17.50% | 30.00% | 89.61% | 9,650.5 |
| 4 | $2/3$ | **20.00%** | 30.00% | **82.66%** | **7,477.9** |

Increasing $G$ from 3 to 4 reduces the zero-advantage rate from 89.61% to 82.66%, improves Pass@1 from 17.50% to 20.00%, and reduces inference tokens per successful trajectory from 9,650.5 to 7,477.9. The larger group therefore provides denser relative learning signals, but requires more offline exploration.

Table 11: Performance–cost trade-off for sibling group size.

| $G$ | GPU-hours | Training-generation tokens | Pass@1 | Pass@4 | Avg. tool calls | Avg. response tokens |
|---|---|---|---|---|---|---|
| 3 | 21.48 | 846,207 | 17.50% | 30.00% | 1.038 | 1,688.8 |
| 4 | 28.06 | 2,139,039 | 20.00% | 30.00% | 0.662 | 1,495.6 |

Moving from $G=3$ to $G=4$ increases GPU-hours by approximately 30.6% and training-generation tokens by approximately 152.8%, while improving Pass@1 by 2.5 percentage points and leaving Pass@4 unchanged. The resulting policy uses approximately 36.2% fewer tool calls and 11.4% fewer response tokens at inference time. Consequently, $G=3$ provides the stronger default cost–performance trade-off, whereas $G=4$ is useful when Pass@1 and concise inference behavior are prioritized.

For $G=2$, the coefficient in Eq. (106) is zero, so the conditional expectation of the relative advantage degenerates to zero. Therefore, $G=3$, corresponding to two sibling continuations, is the minimum viable configuration that preserves an informative relative signal.

### G.3 Trunk Quality and Failure Dynamics

Table 12: Aggregate trunk outcomes in the $\tau^3$-Airline training run.

| Trunk metric | Percentage of all trunks |
|---|---|
| Final task success | 40.00% |
| Failed within Turn 1 | 6.88% |
| Failed within the first 2 turns | 22.50% |
| Failed within the first 3 turns | 28.75% |
| Failed within the first 5 turns | 39.38% |
| No terminal environment signal | 10.63% |
| Average trunk length | 5.39 turns |

Only 6.88% of trunks fail within the first turn, indicating that catastrophic early failure is not the dominant failure mode. Most errors occur in the middle or later stages.

Table 13: Evolution of failure modes during RTPO training.

| Metric | Step 10 | Step 20 | Step 30 | Step 40 | Step 50 |
|---|---|---|---|---|---|
| Success rate | 15.63% | 37.50% | 50.00% | 56.25% | 40.63% |
| Normal termination but failure | 46.88% | 53.13% | 46.88% | 40.63% | 59.38% |
| No terminal signal | 37.50% | 9.38% | 3.13% | 3.13% | 0.00% |
| Failure within Turn 1 | 3.13% | 6.25% | 9.38% | 6.25% | 9.38% |
| Failure within 2 turns | 15.63% | 15.63% | 31.25% | 25.00% | 25.00% |
| Failure within 3 turns | 21.88% | 25.00% | 34.38% | 25.00% | 37.50% |
| Failure within 5 turns | 25.00% | 40.63% | 40.63% | 31.25% | 59.38% |
| Avg. successful turns | 4.40 | 3.92 | 1.88 | 2.06 | 1.92 |
| Avg. failed turns | 13.30 | 7.30 | 4.06 | 5.50 | 2.84 |

The fraction of trajectories without a terminal environment signal decreases from 37.50% at Step 10 to 0% at Step 50, while the average failed-trajectory length decreases from 13.30 to 2.84 turns. RTPO therefore substantially reduces trajectories that stall for a long time or fail to complete the interaction. Later failures increasingly take the form of fast but incorrect termination rather than persistent generation until truncation.

The implementation does not use heuristic trunk filtering. All trunks are sampled on-policy from the current model. Filtering low-quality trunks could reduce sibling-sampling cost, but would alter the actually visited state distribution and concentrate optimization on manually selected states. Instead, RTPO retains all trunks and uses same-state sibling returns to determine whether a boundary supplies an informative relative signal.

### G.4 Selective-Gradient Optimization and Training Efficiency

**Why the loss is restricted to current-turn tokens.** RTPO applies the policy loss only to the output tokens of the current-turn siblings. Prefix tokens and the trunk trajectory define the state and conditioning context that were actually reached, while the downstream continuation supplies the return used to evaluate the current action. Assigning the same turn-level advantage to prefix or downstream-continuation tokens would reintroduce the trajectory-level credit contamination analyzed in Theorem 2. Each reverse stage applies the direct loss to the current-turn outputs, and the complete reverse sweep optimizes every turn. This selective loss isolates turn-level credit.

Table 14: Training-generation cost and deployment-time response efficiency on $\tau^3$-Airline.

| Method | Training-generation tokens | Pass@1 | Response tokens/success |
|---|---|---|---|
| GRPO | 604,532 | 11.25% | 21,952.9 |
| Tree-GRPO | 781,325 | 16.25% | 15,619.5 |
| **RTPO** ($G = 3$) | 846,207 | **17.50%** | **9,650.5** |

RTPO generates approximately 8.3% more training tokens than Tree-GRPO and 40.0% more than GRPO, while attaining the highest Pass@1. Relative to Tree-GRPO, it reduces response tokens per successful trajectory by 38.2%; relative to GRPO, the reduction is approximately 56.0%. The additional offline continuation sampling therefore does not translate into more verbose deployment-time inference.

**Wall-clock time and GPU-hours.** Under the same Qwen3-1.7B model, four A100 GPUs, and 160 training trunks, the measured cost is:

Table 15: End-to-end training cost under the same hardware.

| Method | Wall-clock | GPU-hours | Generated tokens | Relative cost |
|---|---|---|---|---|
| GRPO | 3.80 h | 15.20 | 604,532 | 1.00× |
| RTPO ($G = 3$) | 5.37 h | 21.48 | 846,207 | 1.41× |

RTPO increases wall-clock time and GPU-hours by approximately 41.3%, and training-generation tokens by approximately 40.0%. The additional cost comes primarily from sibling-continuation generation, turn-by-turn reverse updates, and synchronization of the latest policy between reverse stages. RTPO thus trades higher training-time computation for state-matched credit assignment and current-policy downstream continuations.

**Compute-matched comparison.** The following comparison uses approximately the same total GPU-hour budget. It is distinct from the fully trained RTPO result at 21.48 GPU-hours.

Table 16: GRPO and compute-matched RTPO under approximately equal training cost.

| Metric | GRPO | RTPO ($G = 3$), compute-matched |
|---|---|---|
| GPU-hours | 15.20 | 15.55 |
| Pass@1 | 11.25% | 11.25% |
| Pass@4 | 25.00% | **35.00%** |
| Environment completion | 20.00% | **30.00%** |
| Generation truncation | 60.00% | **47.50%** |
| Total tool calls | 118 | **98** |
| Tool-execution success | 57.63% | **62.24%** |
| Average tool calls | 1.475 | **1.225** |
| Average response tokens | 2,469.7 | **2,437.7** |

With only approximately 2% more GPU-hours, RTPO matches GRPO on Pass@1, improves Pass@4 from 25.00% to 35.00%, and increases environment completion from 20.00% to 30.00%. It also reduces truncation, total tool calls, and average response length while improving tool-execution success.

**Matched maximum inference budget.** All methods below use the same limits of 20 turns, 2,048 tokens per turn, and 8,192 response tokens per trajectory.

Table 17: Performance under the same maximum inference budget.

| Method | Pass@1 | Pass@4 | Avg. tool calls | Avg. response tokens | Successes/1K tokens |
|---|---|---|---|---|---|
| GRPO | 11.25% | 25.00% | 1.475 | 2,469.7 | 0.046 |
| Tree-GRPO | 16.25% | 25.00% | 0.013 | 2,538.2 | 0.064 |
| **RTPO** ($G = 3$) | **17.50%** | **30.00%** | 1.038 | **1,688.8** | **0.104** |

Under the same maximum inference budget, RTPO attains the highest Pass@1, Pass@4, and number of successful trajectories per 1,000 response tokens, while producing the shortest average responses. Its gains therefore do not come from allowing longer test-time trajectories.

### G.5 Tool-Use Behavior on GSM8K

The total number of tool calls should be interpreted relative to dataset size: GSM8K contains 1,319 distinct test problems, while AIME24 and AIME25 each contain 30. Across all three benchmarks, RTPO improves accuracy while reducing the total number of calls relative to the vanilla model.

Table 18: Accuracy and total tool calls across mathematical benchmarks.

| Benchmark | Vanilla accuracy / calls | RTPO accuracy / calls |
|---|---|---|
| AIME24 | 3.33% / 17 | **33.33% / 4** |
| AIME25 | 13.33% / 23 | **26.67% / 2** |
| GSM8K | 76.95% / 731 | **94.84% / 482** |

We further audit the complete trajectories for the first 100 distinct GSM8K test problems generated by the RTPO checkpoint.

Table 19: Audit of Python use in 100 distinct GSM8K trajectories.

| Audit metric | Result |
|---|---|
| Distinct trajectories audited | 100 |
| Trajectories using Python | 36% |
| Average calls per trajectory | 0.36 |
| Trajectories with zero calls | 64 |
| Trajectories with one call | 36 |
| Trajectories with two or more calls | **0** |
| Calls with valid Python syntax | 36/36 |
| Calls executed successfully | 36/36 |
| Correct tool-using trajectories | 34/36 |
| Trajectories with repeated calls | **0** |

There are no repeated questions and no trajectory invokes Python more than once. All 36 calls contain valid syntax and execute successfully. In 34 of the 36 tool-using trajectories, the model has already derived the correct numerical result before invoking Python. The calls are one-shot arithmetic checks rather than multi-step tool search, repeated code, or duplicate execution. For example, on GSM8K QID 55, the model first derives $30 - 2 = 28$ and $28/2 = 14$, then invokes Python once to verify the result before returning $\boxed{14}$. Thus, the GSM8K call total reflects many distinct problems receiving a single low-cost numerical verification, rather than repeated tool use within a small set of trajectories.

### G.6 On-Policy Continuation and Synchronization Frequency

As characterized by Theorem 3, standard RTPO synchronizes the rollout model after every reverse stage. When turn $k$ is optimized, its downstream continuation is therefore generated by the current policy $\pi_{\theta_{>k}}$, after turns $k+1, \dots, K-1$ have been updated. The resulting estimator is

$$\widehat{Q}^{\mathrm{on}}_{j,k} = r_{j,k} + \gamma^{\tau_{j,k}} \widehat{F}^{\pi_{\theta_{>k}}}_{j,k}, \qquad \omega_{j,k} \equiv 1. \tag{107}$$

This construction assigns the turn-$k$ action a return under the current downstream policy and avoids a product of trajectory-level importance ratios.

At the opposite synchronization endpoint, the off-policy variant reuses downstream continuations generated by the initial rollout policy $\pi_{\theta_0}$ and applies a clamped trajectory-level importance weight:

$$\widehat{Q}^{\mathrm{off}}_{j,k} = r_{j,k} + \gamma^{\tau_{j,k}} \overline{\omega}_{j,k} \widehat{F}^{\pi_{\theta_0}}_{j,k}, \tag{108}$$

$$\overline{\omega}_{j,k} = \mathrm{clamp}\left( \prod_{h=k+1}^{K-1} \prod_{t=1}^{T_{j,h}} \frac{\pi_{\theta_{>k}}(a_{j,h,t} \mid s_{j,h,t})}{\pi_{\theta_0}(a_{j,h,t} \mid s_{j,h,t})}, \rho_{\min}, \rho_{\max} \right). \tag{109}$$

Per-stage on-policy continuation yields consistent gains on the long-horizon deep-search tasks, with the largest improvement on GAIA Level 3. The two variants are broadly comparable on HLE, which

Table 20: Output-hit accuracy of off-policy and per-stage on-policy RTPO with Qwen3-8B.

| Benchmark | Off-policy | On-policy | Difference |
|---|---|---|---|
| GAIA | 23.30% | 29.13% | +5.83 pp |
| GAIA Level 3 | 8.33% | 16.67% | +8.33 pp |
| WebWalkerQA | 6.00% | 9.50% | +3.50 pp |
| XBench | 8.00% | 15.00% | +7.00 pp |
| HLE | 12.45% | 12.12% | −0.33 pp |

is dominated by more static and shorter retrieval. These results localize the cost of stale continuations to the long-horizon settings targeted by RTPO.

### G.7 Budget Consumption Across Training

The initial policy produces longer trajectories and more tool calls, so early training stages are more expensive per trunk. The training schedule does not, however, reserve a fixed number of tool calls for each stage. Every stage continues under the predefined trunk-sampling and update schedule, and the actual per-trunk cost decreases as the policy becomes more efficient.

Table 21: Evolution of success, tool use, and generation cost during RTPO training.

| Training point | Task success | Tool calls/trunk | Tool-execution success | Generated tokens/trunk |
|---|---|---|---|---|
| Step 10 | 15.63% | 7.34 | 74.04% | 8,013.5 |
| Step 20 | 37.50% | 2.06 | 72.73% | 7,758.8 |
| Step 30 | 50.00% | 0.97 | 87.10% | 3,533.3 |
| Step 40 | 56.25% | 1.84 | 94.92% | 5,469.0 |
| Step 50 | 40.63% | 0.94 | 93.33% | 1,669.3 |

From Step 10 to Step 50, tool calls decrease from 7.34 to 0.94 per trunk, generated tokens decrease from 8,013.5 to 1,669.3 per trunk, and tool-execution success increases from 74.04% to 93.33%. Tool use therefore becomes less frequent and more reliable as training progresses.