TREEPPO: LEARNING FROM TREES AND EXPLORING WITH CRITICS FOR MULTI-TURN LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-turn LLM agents typically make sequential decisions over long interaction trajectories, with reward signals often available only upon task completion. Existing reinforcement learning (RL) methods largely rely on these sparse terminal rewards, making it difficult to distinguish the contributions of intermediate actions to final task success. Although a critic can mitigate this limitation by estimating state values learning, accurate values for intermediate states remains challenging when the critic is trained primarily on sparse terminal rewards. To address this challenge, we propose **TreePPO**, a tree-search-based actor-critic RL framework that establishes a bidirectional feedback loop between rollout exploration and critic learning for continual improvement. TreePPO aggregates rollouts with shared action prefixes to provide fine-grained value supervision, while the learned critic guides adaptive rollout allocation toward tasks with greater exploration potential. Specifically, TreePPO organizes trajectories sharing ordered action prefixes into an **Action-Prefix Value Tree (APVT)**. By aggregating terminal rewards from different continuations of each shared prefix, the APVT provides empirical value targets for intermediate states, enabling finer-grained critic learning. APVT provides supervision for critic warm-up, while subsequent on-policy rollouts continually expand the state space covered by APVT. We further use the critic's estimates of task difficulty to adaptively allocate the rollout budget, assigning more samples to moderately difficult tasks that offer greater exploration value. Extensive experiments across Multi-hop QA, ALFWorld, and WebShop show that TreePPO achieves the best overall performance among the evaluated methods with both Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct actors. With Qwen2.5-7B-Instruct, TreePPO achieves an average exact match (EM) of 36.60% across four Multi-hop QA benchmarks and an ALFWorld success rate of 86.07%, exceeding the strongest baselines in these two settings by **2.15** and **5.97** percentage points, respectively.

1 INTRODUCTION

Multi-turn interaction enables large language model (LLM) agents to tackle complex tasks such as mathematical reasoning and coding (Gou et al., 2024; Yang et al., 2024b). By interleaving reasoning and tool use with feedback from the environment, agents can revise their strategies across a sequence of interdependent decisions (Yao et al., 2023). Reinforcement learning (RL) provides a promising framework for optimizing such sequential behavior (Wang et al., 2025; Li et al., 2026). However, credit assignment under sparse terminal rewards remains a key challenge when training agents over long interaction trajectories (Ji et al., 2025; Li et al., 2026).

Existing approaches to learning from sparse terminal rewards can be broadly grouped into two categories according to how they assign credit to intermediate actions. The first derives relative advantages directly from sampled returns. In multi-turn settings, standard GRPO (Shao et al., 2024) assigns the same trajectory-level advantage to every turn, making it difficult to distinguish the contributions of individual actions (Li et al., 2026). Tree-GRPO (Ji et al., 2025) addresses this limitation by organizing rollouts into a tree and computing local relative advantages across alternative continuations. Under a limited sampling budget, however, each branching point may receive too few continuations to yield reliable advantage estimates. The second category, exemplified by Turn-PPO (Li et al., 2026), learns a critic to estimate intermediate-state values and computes turn-level advantages using GAE with bootstrapping. Although this approach produces turn-specific value

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.

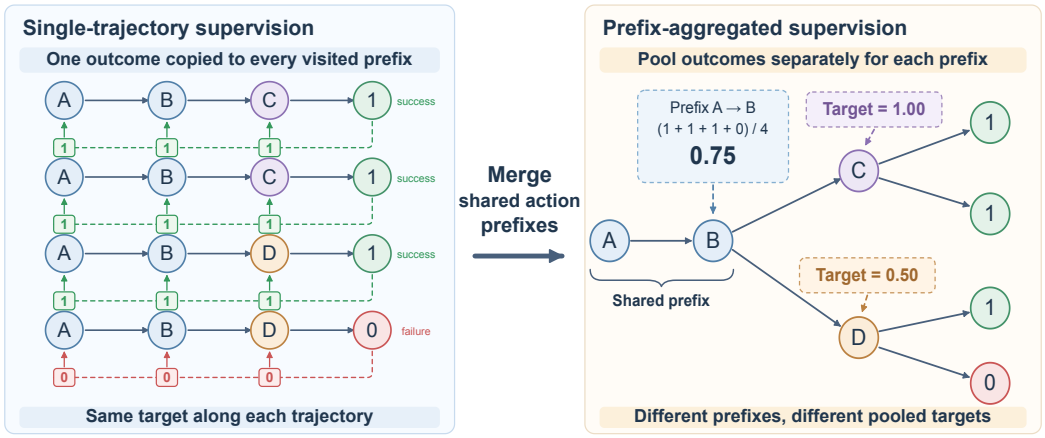


Figure 1: **(a)** Single-trajectory supervision repeats one terminal target across turns. **(b)** APVT aggregates continuation outcomes at shared prefixes to provide differentiated value supervision.

estimates, their targets may still rely on the same final reward, as in the PPO setting analyzed by SPPO (Wang et al., 2026b), without adding outcome evidence for individual intermediate states. When the value target for an intermediate state is derived from a single sampled continuation, the resulting supervision can vary substantially across possible continuations, potentially reducing the reliability of both value and advantage estimates. These limitations suggest that finer-grained credit assignment alone may not be sufficient. The key challenge is therefore to derive reliable learning signals for intermediate decisions from sparse terminal rewards under a limited sampling budget.

This challenge motivates us to revisit the relationship between rollout structure and value learning in multi-turn interactions. Our approach builds on two connected observations. First, although a single trajectory yields only one terminal outcome, multiple rollouts for the same task can share an action prefix before branching into different continuations. Aggregating the terminal returns of these continuations provides multiple outcome samples for the shared prefix, yielding richer supervision than a single-trajectory return (see Figure 1). This observation motivates grouping returns by shared action prefixes and conditioning the critic on those prefixes. Second, a critic trained with aggregated prefix-level supervision can also inform rollout allocation. With terminal rewards, the value estimate of an action prefix is proportional to the probability that the current policy will eventually solve the task from that prefix, thereby providing a policy-dependent measure of task difficulty. We use this signal to prioritize continuation rollouts to moderately difficult tasks, which balance challenge and attainability. This strategy is analogous to PCL's prompt-level curriculum, where an online value model prioritizes prompts of intermediate difficulty during RL post-training (Gao et al., 2025). Together, these observations suggest a feedback loop in which shared-prefix rollouts provide aggregated supervision for value learning, while the learned critic guides subsequent rollout allocation.

Building on this feedback loop, we propose **TreePPO**, an on-policy actor-critic framework that couples action-prefix value learning with critic-guided rollout allocation through a tree-structured rollout process. TreePPO merges shared action prefixes across tree-search trajectories into an Action-Prefix Value Tree (APVT). Each APVT node corresponds to an action prefix and aggregates terminal rewards across its continuations, providing an empirical value target for a critic conditioned on the task prompt and action prefix. We warm up the critic using APVT supervision. As training proceeds, new on-policy rollouts continually expand the tree. Their terminal rewards are combined with previously collected rewards at shared prefixes, enriching critic supervision without additional sampling. The critic supports turn-level advantage estimation and guides rollout allocation. For each task, we average its predictions across trunk action prefixes to estimate difficulty. TreePPO then assigns more continuation rollouts to moderately difficult tasks. APVT-based return aggregation and critic-guided rollout allocation close the feedback loop between value learning and exploration.

Extensive experiments with Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct show that TreePPO achieves the best overall performance among the evaluated baselines across Multi-hop QA, ALFWorld, and WebShop. The two actors achieve average exact match (EM) scores of 30.90% and 36.60% across

four Multi-hop QA benchmarks and ALFWorld success rates of 83.33% and 86.07%, respectively. Ablation studies further support the effectiveness of its key components.

Our contributions are as follows:

- We propose TreePPO, a framework that establishes a closed feedback loop between critic learning and rollout exploration: aggregated outcomes from multiple continuations enrich critic supervision, while the critic guides adaptive rollout allocation toward moderately difficult tasks.
- We introduce the Action-Prefix Value Tree (APVT), which aggregates terminal rewards over shared ordered action prefixes to construct empirical value targets for intermediate decisions. These targets support both critic initialization and subsequent updates using on-policy rollouts.
- Experiments on multi-hop QA, ALFWorld, and WebShop show that TreePPO achieves the highest average QA EM and environment-task success rates among the evaluated baselines at both actor scales. Ablation studies further demonstrate the effectiveness of the key components of TreePPO.

2 Related Work

Credit Assignment and Value Supervision. Outcome-based RL enables LLMs to learn search and multi-turn interaction from environment feedback (Jin et al., 2025; Wang et al., 2025). Methods for credit assignment under sparse terminal rewards fall into two broad categories. The first estimates advantages from sampled returns without a learned critic (Shao et al., 2024; Ahmadian et al., 2024). Trajectory-level estimates assign the same advantage to all turns. Finer-grained estimates group repeated states or shared prefixes (Feng et al., 2025; Tran et al., 2025), or sample alternative continuations (Ji et al., 2025; Yang et al., 2025; Kazemnejad et al., 2025). Under limited rollout budgets, however, each state or prefix may have too few return samples for reliable estimation. The second learns a critic for value and advantage estimation (Schulman et al., 2017; 2015). For multi-turn agents, hierarchical and action-level methods align value learning with interaction turns or complete actions (Zhou et al., 2024; Li et al., 2026; Wang et al., 2026a). Nevertheless, value targets at different positions may share a single terminal return, as in the PPO setting analyzed by SPPO (Wang et al., 2026b). Per-turn value prediction therefore does not itself provide additional return samples, and targets from a single continuation may remain noisy. Related work improves value pretraining and advantage estimation (Yuan et al., 2025; Yue et al., 2025), while ATPO constructs critic targets from returns collected through repeated tree expansion (Cao et al., 2026). Beyond collecting additional rollouts, an important question is how to reuse available returns for critic supervision under a fixed sampling budget. TreePPO addresses this by aggregating returns from matching ordered action prefixes across rollout trees. Each shared prefix thus receives supervision from multiple continuations rather than a single trajectory, reducing the need for fresh rollouts to estimate its value.

Curriculum Learning and Rollout Allocation. Prior work uses reward histories, capability estimates, or value predictions to select training prompts and allocate rollout budgets (Zheng et al., 2025b; Yao et al., 2026; Gao et al., 2025). PCL prioritizes prompts of intermediate predicted difficulty (Gao et al., 2025). TRACE trains a prefix-value predictor on aggregated tree returns to allocate rollouts at prompt roots and intermediate prefixes (Zou et al., 2026). Its experimental instantiations use TreeRPO or Tree-GRPO for policy optimization, with advantages derived from group-relative return comparisons; the learned predictor guides allocation but does not provide the policy's advantage baseline. TreePPO instead adopts a PPO-based actor–critic formulation in which the same critic provides turn-level value baselines and guides task-level rollout allocation. APVT aggregates returns at matching ordered action prefixes across trajectories and training iterations to supervise this critic. Its predictions prioritize moderately difficult tasks for additional continuations, whose returns update APVT statistics for subsequent critic learning. The distinction lies in how learned values enter policy optimization, together with the persistent reuse of action-prefix outcomes for critic supervision.

3 TreePPO

3.1 Problem Formulation

For a task q with initial observation o_0, an interaction trajectory is $\tau = (q, o_0, (c_t, a_t, o_t)_{t=1}^{T})$, where c_t denotes reasoning, a_t an environment action with its arguments, and o_t the resulting

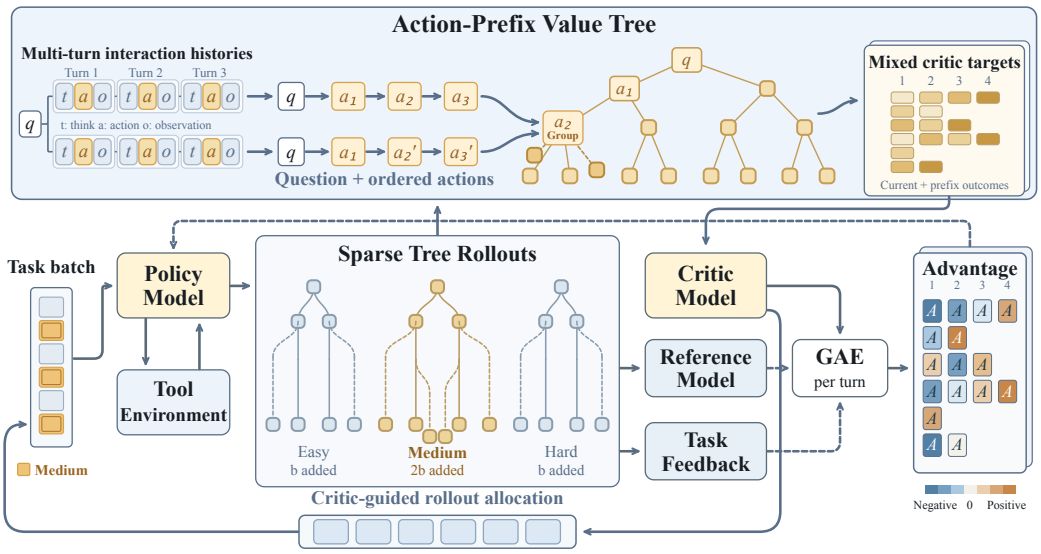


Figure 2: **Overview of the TreePPO training pipeline.** Within each iteration, a fixed policy samples trunk trajectories and generates continuations from uniformly selected branch points. The critic evaluates trunk prefixes to allocate more branch rollouts to medium-difficulty tasks. An Action-Prefix Value Tree aggregates terminal outcomes at shared prefixes; these statistics are combined with current outcomes to supervise the critic. The critic also provides turn-level baselines for policy optimization. New rollouts update the tree and continue the feedback loop.

observation. At turn t, the actor policy π generates (c_t, a_t) conditioned on the interaction history $h_t = (q, o_0, (c_i, a_i, o_i)_{i<t})$. The objective is to learn a policy that maximizes the expected terminal return over the task distribution.

The environment provides only a terminal reward $R(\tau) \in \mathbb{R}$, with $r_t = 0$ for $t < T$ and $r_T = R(\tau)$. The undiscounted return and the value of history h_t under policy π are

$$\widehat{G}_t = \sum_{j=t}^{T} r_j = R(\tau), \qquad V^{\pi}(h_t) = \mathbb{E}_{\pi}[R(\tau) \mid h_t]. \tag{1}$$

In PPO-based LLM RL with sparse terminal rewards, a common setting is $\gamma = \lambda_A = 1$, as discussed in SPPO (Wang et al., 2026b). Under this setting, the critic's GAE-based return target reduces to the Monte Carlo return:

$$\widehat{G}_t^{\lambda} = \bar{V}(h_t) + \sum_{j=t}^{T} \left[r_j + \bar{V}(h_{j+1}) - \bar{V}(h_j) \right] = \sum_{j=t}^{T} r_j = R(\tau). \tag{2}$$

where $\bar{V}$ denotes the pre-update critic and $\bar{V}(h_{T+1}) = 0$. Thus, under this outcome-based supervision, every visited history receives the same terminal reward as its regression target, despite potentially having different expected returns.

3.2 Overview

TreePPO establishes a bidirectional feedback loop between tree-based rollout exploration and critic learning. An Action-Prefix Value Tree (APVT) aggregates terminal returns at shared action prefixes to supervise the critic. The critic evaluates trunk trajectories and uses its value predictions to dynamically allocate branch rollout budgets across tasks. It also provides value baselines for turn-level policy optimization. Returns from these rollouts, in turn, enrich APVT supervision for subsequent critic updates. Figure 2 illustrates the complete training procedure of TreePPO.

Following Tree-GRPO (Ji et al., 2025), we collect rollouts in two stages: initialization and a single round of expansion. During initialization, we sample trunk trajectories in parallel and record their

interaction histories. The critic then evaluates the action prefixes along these trunk trajectories. Based on its value predictions, we dynamically assign each task a branch rollout budget B_q for the subsequent expansion stage (Section 3.4). During expansion, we use random branching: for each task, we sample eligible branch points uniformly at random from its trunk trajectories according to the assigned budget B_q, restore their histories and environment states, and generate the additional continuations in parallel. The sampling policy π_k remains fixed throughout both stages. Unlike iterative tree search that repeatedly selects or evaluates nodes for further expansion (Zhou et al., 2023; Cao et al., 2026), this procedure does not expand newly generated branches further, avoiding dependencies between successive expansion rounds. The resulting trajectories share action prefixes but explore different continuations, providing multiple terminal outcomes for prefix-level value supervision. APVT organizes the collected trajectories by action prefix. While the actor conditions on the full interaction history h_t, the critic estimates the expected terminal return conditioned on the task and preceding actions:

$$x_t = g(h_t) = (q, a_1, \ldots, a_{t-1}), \qquad \widehat{v}_t = V_\phi(x_t). \tag{3}$$

where $x_1 = (q)$ denotes the empty prefix. Histories with matching action prefixes may differ in reasoning and observations, limiting outcome sharing when grouped by full history. Using action prefixes allows us to pool these outcomes while preserving the ordered action sequence, and conditioning the critic on the same representation aligns its predictions with the aggregated supervision. The next section explains how APVT aggregates returns into critic targets.

3.3 Critic Learning with APVT

Action-prefix aggregation. For each task, an APVT organizes sampled trajectories by their ordered action prefixes. The root represents the empty prefix, and each edge records an environment action with its arguments. Inserting a trajectory reuses matched children under the same parent and creates nodes for unmatched actions. QA search actions allow semantic matching after an exact miss; other actions require normalized exact matches. Thus, trajectories with matched prefixes share nodes while different preceding paths remain separate.

Each node u accumulates an outcome count N_u and return sum S_u from trajectories passing through it, giving the empirical value target

$$\widehat{V}_u = S_u / N_u, \qquad N_u > 0. \tag{4}$$

This pools multiple continuation outcomes for the same matched prefix.

Critic warm-up and online learning. We first warm up the critic while keeping the actor fixed, collecting rollouts to construct the APVT and training the critic on its node values. During online training, we retain node counts and return sums across iterations, so that supervision at matched prefixes is not limited to the outcomes collected in the current batch. New on-policy rollouts update these statistics and add nodes for newly encountered prefixes. Because the accumulated outcomes may come from different histories and earlier policies, we combine the aggregated estimates with the current trajectory's outcome. Let $u_{\tau,t}$ denote the node corresponding to $x_{\tau,t}$, and let $\mu_{-\tau}(u)$ be its mean after excluding the current execution's contribution. The online target is

$$y_{\tau,t} = \begin{cases} \alpha R(\tau) + (1-\alpha)\mu_{-\tau}(u_{\tau,t}), & N_{-\tau}(u_{\tau,t}) > 0. \\ R(\tau), & N_{-\tau}(u_{\tau,t}) = 0. \end{cases} \tag{5}$$

where $N_{-\tau}$ counts the remaining outcomes. Self-exclusion avoids including the same outcome in both components. After inserting the current batch, we freeze the targets. The base regression objective is

$$\mathcal{L}_V(\phi) = \frac{1}{2|\mathcal{S}_k|} \sum_{(x,y)\in\mathcal{S}_k} \big(V_\phi(x) - y\big)^2. \tag{6}$$

where $\mathcal{S}_k$ contains one input–target pair per interaction turn in the batch.

3.4 Critic-Guided Rollout Allocation

The critic also guides exploration by estimating task difficulty under the current policy. At each training iteration, we use the pre-update critic to predict values for the action-prefix nodes along each

task's trunk trajectories. We average these predictions to score each task, then rank the tasks by their scores. We partition them into approximately equal-sized easy, medium, and hard groups, denoted by $\mathcal{Q}_{\text{easy}}$, $\mathcal{Q}_{\text{medium}}$, and $\mathcal{Q}_{\text{hard}}$; higher scores indicate easier tasks. Recomputing these groups allows rollout allocation to adapt as the policy improves.

Motivated by PCL's finding that prompts with success probabilities near 50% provide sample-efficient learning signals during RL post-training (Gao et al., 2025), we prioritize medium-difficulty tasks to focus exploration where the current policy faces a challenge but can still succeed. Specifically, we assign twice as many additional branch rollouts to medium-difficulty tasks as to the other two groups:

$$B_q = \begin{cases} 2b, & q \in \mathcal{Q}_{\text{medium}}. \\ b, & q \in \mathcal{Q}_{\text{easy}} \cup \mathcal{Q}_{\text{hard}}. \end{cases} \tag{7}$$

Here b denotes the base branch budget, and B_q excludes the initial trunk rollouts. The budget values and task-specific rollout counts are given in Section 4.1.

3.5 Turn-level Policy Optimization

The critic links exploration to policy optimization: its predictions guide branch budgets and provide the turn-level baselines used to update the actor.

Turn-level advantages. The APVT targets supervise the critic; actor advantages use the reward sequence of the current on-policy batch. Following turn-level actor–critic learning (Li et al., 2026), tokens generated within a turn share an advantage. Let $r^{\pi}_{\tau,t}$ denote the reward used for actor optimization. With the pre-update critic $\bar{V}_k$, GAE (Schulman et al., 2015) is computed as

$$\begin{aligned} \delta_{\tau,t} &= r^{\pi}_{\tau,t} + \gamma \bar{V}_k(x_{\tau,t+1}) - \bar{V}_k(x_{\tau,t}), \\ \widehat{A}_{\tau,t} &= \sum_{\ell=0}^{T_\tau - t} (\gamma\lambda_A)^{\ell} \delta_{\tau,t+\ell}, \qquad \bar{V}_k(x_{\tau,T_\tau+1}) = 0. \end{aligned} \tag{8}$$

Here, γ is the discount factor and λ_A is the GAE parameter. We set $\gamma = \lambda_A = 1$ in all experiments. The terminal value is zero. The APVT mixed targets do not replace $r^{\pi}_{\tau,t}$ in GAE. Advantages remain fixed while both models are updated.

Policy optimization. Following GSPO (Zheng et al., 2025a), we use a length-normalized importance ratio over each turn's generated response. Let $z_{\tau,t}$ denote the reasoning and action tokens generated from history $h_{\tau,t}$. We define

$$\rho_{\tau,t}(\theta) = \exp\left[\frac{1}{|z_{\tau,t}|} \sum_{j=1}^{|z_{\tau,t}|} \log \frac{\pi_\theta(z_{\tau,t,j} \mid h_{\tau,t}, z_{\tau,t,<j})}{\pi_k(z_{\tau,t,j} \mid h_{\tau,t}, z_{\tau,t,<j})} \right]. \tag{9}$$

We combine this geometric importance ratio with turn-level advantages to maximize the clipped policy surrogate:

$$J_{\text{clip}}(\theta) = \mathbb{E}_{\tau\sim\mathcal{B}_k}\left[\frac{c_{q(\tau)}}{|\mathcal{I}_\tau|} \sum_{j\in\mathcal{I}_\tau} \min\left\{ \widetilde{\rho}_{\tau,j}\widehat{A}_{\tau,t(j)}, \text{clip}(\widetilde{\rho}_{\tau,j}, 1-\epsilon_l, 1+\epsilon_h)\widehat{A}_{\tau,t(j)} \right\} \right]. \tag{10}$$

Here, τ is sampled uniformly from the current batch $\mathcal{B}_k$, $\mathcal{I}_\tau$ contains valid actor-token positions, and $t(j)$ identifies the turn containing token j. The implemented ratio $\widetilde{\rho}_{\tau,j}$ uses the shared turn-level ratio while preserving token-specific gradients. We use $c_q = N/(QN_q)$ to balance task contributions, where N is the total trajectory count, Q the number of tasks, and N_q the trajectory count for task q.

Together, these components form a feedback loop: shared-prefix rollouts provide aggregated supervision for the critic, whose estimates support policy optimization and guide subsequent exploration. As the policy evolves, new rollouts expand APVT coverage and update outcome statistics at shared prefixes, providing additional evidence for subsequent critic learning. Further details on the proposed framework, its implementation, and the experimental setup are provided in Appendices A and B.

Table 1: Multi-hop QA EM and ALFWorld and WebShop success rates (%, ↑). Multi-hop QA Avg. averages only the four Multi-hop QA benchmarks. ALFWorld and WebShop scores are averaged across three random seeds. Bold indicates the best results, while dashes indicate unavailable results.

	Multi-hop QA (EM, %)					**ALFWorld**	**WebShop**
Method	HotpotQA	2Wiki	MuSiQue	Bamboogle	Multi-hop QA Avg.	Success (%)	Success (%)
Qwen2.5-3B-Instruct							
Search-R1 *(reference only)*	30.40	29.30	12.00	24.00	23.93	—	—
StepSearch *(reference only)*	34.50	32.00	**17.40**	**34.40**	29.58	—	—
PPO	31.20	31.30	9.75	26.40	24.66	30.85	64.53
GRPO	30.80	30.30	9.65	24.80	23.89	75.87	66.87
Tree-GRPO	30.20	29.10	8.95	29.10	24.34	61.94	60.20
Turn-PPO	34.60	33.70	12.20	28.00	27.13	74.38	75.80
TreePPO (ours)	**38.70**	**37.10**	14.20	33.60	**30.90**	**83.33**	**77.37**
Qwen2.5-7B-Instruct							
Search-R1 *(reference only)*	39.40	31.20	18.10	38.40	31.78	—	—
StepSearch *(reference only)*	38.60	36.60	**22.60**	40.00	34.45	—	—
PPO	40.40	37.50	17.50	39.20	33.65	56.22	69.50
GRPO	41.00	39.00	16.70	38.40	33.78	79.10	78.57
Tree-GRPO	34.25	33.90	13.45	33.60	28.80	66.67	70.73
Turn-PPO	39.00	36.20	13.50	42.00	32.68	80.10	79.73
TreePPO (ours)	**43.00**	**39.70**	18.90	**44.80**	**36.60**	**86.07**	**80.53**

4 Experiments

4.1 Experimental Setup

Datasets. We train on HotpotQA (Yang et al., 2018) and evaluate on its held-out split, 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023), alongside ALFWorld (Shridhar et al., 2021) and WebShop (Yao et al., 2022).

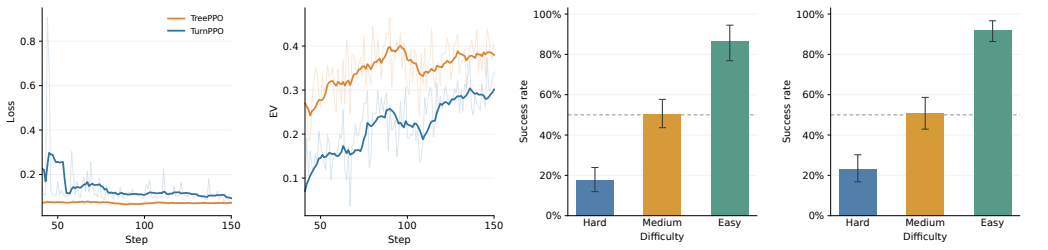
Baselines. We compare with actor–critic methods PPO (Schulman et al., 2017) and Turn-PPO (Li et al., 2026), and group-relative methods GRPO (Shao et al., 2024) and Tree-GRPO (Ji et al., 2025). Published Search-R1 (Jin et al., 2025) and StepSearch (Zheng et al., 2025c) results serve as separate references because their training data and supervision differ.

Implementation Details. We use Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct as actors (Yang et al., 2024a). We select $\alpha = 0.8$. Multi-hop QA builds on Search-R1 (Jin et al., 2025) with an E5-based Wikipedia retriever (Wang et al., 2022). Initial steps hold the actor fixed to construct APVT and initialize the critic; these count toward the same total step budget as the baselines. GRPO and Turn-PPO use six independent rollouts per Multi-hop QA question and eight per ALFWorld or WebShop task. Tree-GRPO uses two trunks plus four branches for Multi-hop QA, and four trunks plus four branches for ALFWorld and WebShop. TreePPO uses the same trunk counts but dynamically allocates an average of four branches per task, with base budget $b = 3$. Its total rollout counts therefore average six for Multi-hop QA and eight for ALFWorld and WebShop.

Metrics. We report Multi-hop QA exact match (EM) and its equal-weight benchmark average; terminal EM supplies the binary training feedback. ALFWorld and WebShop success rates are averaged across three random seeds. Further details are provided in Appendices A and B.

4.2 Main Results

Multi-hop QA. TreePPO achieves the highest Multi-hop QA average among the directly comparable methods at both scales: 30.90% with the 3B actor and 36.60% with the 7B actor. Relative to Turn-PPO, the gains are 3.78 and 3.93 percentage points, respectively, and TreePPO improves on Turn-PPO on all four benchmarks at both scales. At 7B, TreePPO also exceeds PPO, GRPO, and Tree-GRPO on every benchmark, improving over the strongest of these baselines, GRPO, by 2.83 points on average. The StepSearch numbers require a different interpretation: StepSearch uses fine-grained MuSiQue supervision, which gives it a task-specific advantage on MuSiQue and leads to the



(a) Critic value loss (b) Explained variance (c) Earlier checkpoint (d) Later checkpoint

Figure 3: Critic diagnostics and difficulty grouping (7B actors). (a) Value loss of TreePPO and Turn-PPO against their respective targets. (b) Explained variance against those targets. (c) Multi-hop QA success by predicted difficulty at step 100. (d) The same at step 150. Groups use mean trunk-prefix critic predictions; success rates use 64 rollouts per question. The dashed line marks 50% success.

Table 2: Progressive ablations of TreePPO with a 3B actor. Each row removes one additional component from the preceding configuration. EM (%, ↑); Avg. averages the four benchmarks. Bold denotes column maxima, including ties.

Variant	HotpotQA	2Wiki	MuSiQue	Bamboogle	Avg.
TreePPO	**38.70**	**37.10**	14.20	**33.60**	**30.90**
w/o critic-guided allocation	38.20	35.90	14.20	32.00	30.08
w/o critic-guided allocation & tree rollout	**38.70**	34.80	**14.70**	31.20	29.85
further w/o APVT-based critic learning	34.60	33.70	12.20	28.00	27.13

highest MuSiQue scores in the table. TreePPO nevertheless outperforms StepSearch on HotpotQA and 2WikiMultiHopQA at both scales and on Bamboogle at 7B; StepSearch remains higher on Bamboogle at 3B. This pattern indicates that TreePPO's gains extend beyond the dataset receiving specialized supervision and are strongest on transfer-oriented evaluations.

ALFWorld and WebShop. TreePPO achieves the highest mean success rates across three random seeds in both environments at both actor scales. On ALFWorld, the 3B and 7B actors reach 83.33% and 86.07%, exceeding the strongest baselines, GRPO and Turn-PPO respectively, by 7.46 and 5.97 percentage points. On WebShop, the corresponding success rates are 77.37% and 80.53%, improving over Turn-PPO, the strongest baseline at both scales, by 1.57 and 0.80 percentage points.

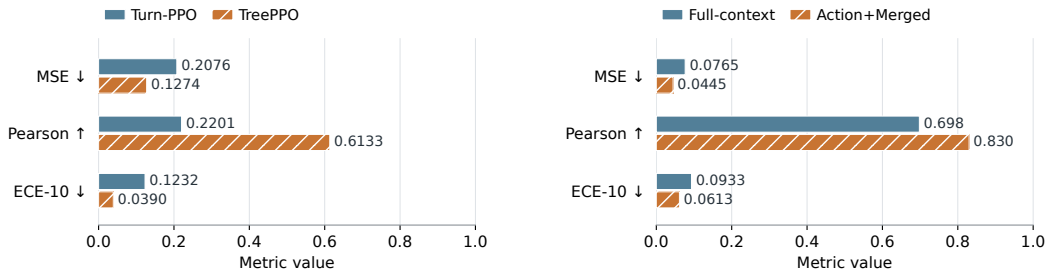
Online critic learning. With 7B actors, TreePPO records mean value loss 0.0717 and explained variance (EV) 0.3676, versus 0.1092 and 0.2687 for Turn-PPO (Figure 3). Its stable loss and higher EV indicate sustained fitting of APVT targets as the actor evolves, supporting the value-learning feedback loop. These training diagnostics are not a shared-target accuracy comparison because targets and measurement procedures differ.

Critic-guided allocation. As a post hoc check of the difficulty grouping, we rank Multi-hop QA questions by the mean critic prediction over their trunk prefix positions at two training checkpoints. We estimate each question's success rate from 64 rollouts in two evaluation batches using the corresponding actor. Success rates rise across the three groups, and the medium groups reach 50.6% and 50.8% (Figure 3). This pattern is consistent with PCL's finding that prioritizing prompts near 50% success can improve RL post-training (Gao et al., 2025).

4.3 Ablation Studies

We further conduct ablation experiments to assess the contributions of critic-guided rollout allocation, tree-structured sampling, and APVT-based critic learning. As shown in Table 2, we observe:

First, replacing critic-guided allocation with random allocation reduces average EM from 30.90% to 30.08%. This supports the critic's role beyond advantage estimation: its predictions provide a useful signal for allocating exploration resources across tasks. **Second,** further replacing tree rollouts with independent trajectories reduces average EM to 29.85%. Explicit branching therefore provides an additional average benefit, although its effect varies across benchmarks. Removing



(a) Online critic checkpoints

(b) Offline critics checkpoints

Figure 4: Critic prediction quality on HotpotQA against MC64 continuation estimates. Lower MSE and ECE-10 and higher Pearson correlation are better.

tree rollouts changes how trajectories are collected but preserves the APVT, which continues to accumulate outcomes at matching action prefixes across trajectories and training iterations. **Third,** with independent rollouts, removing APVT-based critic learning further reduces average EM from 29.85% to 27.13%. This supports the effectiveness of APVT-based value learning even without critic-guided allocation or explicit branching. During the first 25 training steps, APVT variants keep the actor fixed to build the tree and initialize the critic. When APVT is removed, the control variant updates both the actor and critic throughout this period.

4.4 Further Analysis

Historical outcome reuse. In the 3B HotpotQA run, historical-prefix hit rates rise from roughly 30% to 50%, with about five past returns per matched prefix. These statistics exclude current-step outcomes and demonstrate repeated reuse of historical supervision (Figure 5, Appendix C.4).

Value estimation and calibration. We assess online and offline critics against Monte Carlo success rates estimated from 64 continuations (MC64). Online, we evaluate frozen TreePPO and Turn-PPO critics on the same HotpotQA states. TreePPO achieves lower MSE (0.1274 vs. 0.2076) and ECE (0.0390 vs. 0.1232), and higher Pearson correlation (0.6133 vs. 0.2201) against the MC64 references (Figure 4a). In the offline evaluation, Action+Merged uses action-prefix inputs and aggregated targets, whereas Full-context uses full interaction histories and individual trajectory outcomes. The same fixed actor generates the original trajectories and MC64 continuations. APVT training statistics use only training trajectories; held-out trajectory outcomes and MC64 outcomes are excluded from critic training. After one training epoch per configuration, both critics are evaluated against shared MC64 references at the semantic-node level. Action+Merged achieves lower MSE (0.0445 vs. 0.0765) and ECE (0.0613 vs. 0.0933), along with higher Pearson correlation (0.830 vs. 0.698) and within-question Pearson correlation (0.657 vs. 0.461) (Figure 4b). These results support more accurate node-level value prediction and better within-question differentiation on held-out trajectories from questions seen during critic training. Full protocols and metrics are provided in Appendices C.1–C.2.

5 Conclusion

We present TreePPO, an on-policy actor–critic framework coupling action-prefix value learning and critic-guided rollout allocation for multi-turn LLM agents with sparse terminal rewards. Its Action-Prefix Value Tree (APVT) aggregates terminal rewards across shared action prefixes to supervise the critic, whose estimates support turn-level advantages and allocate more rollouts to intermediate-difficulty tasks. Evaluations on four multi-hop QA benchmarks, ALFWorld, and WebShop demonstrate TreePPO's effectiveness across reasoning and interactive decision-making tasks. Critic evaluations show that action-prefix inputs and aggregated supervision better match Monte Carlo value estimates than a full-context critic trained on individual outcomes. Ablations confirm the contributions of action-prefix aggregation and critic-guided allocation. Together, these results support joint design of rollout collection and value supervision. Branching requires restorable states, while accumulated prefix statistics can reflect earlier policies. Future work could address limited state restoration and policy-aware reuse of historical outcomes.

AI USE STATEMENT

We used generative AI tools in two supporting roles. First, as writing assistants, they were used to polish language, improve clarity, and check the grammar of text drafted by the authors. Second, as coding assistants, they were used for code completion, refactoring, and debugging in the data-synthesis pipeline, training scripts, and plotting code. All AI-assisted code was reviewed and checked by the authors. Generative AI tools were not used for research ideation, method design, experimental design, or result interpretation. All claims, analyses, and conclusions were formulated by the authors. The authors reviewed and edited all AI-assisted text, reviewed all AI-assisted code, and validated the code by executing it in the same sandbox environments used for the experiments. The authors take full responsibility for the final content of this work, including its text, claims, analyses, and artifacts.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024. doi: 10.18653/v1/2024.acl-long.662. URL https://aclanthology.org/2024.acl-long.662/.

Ruike Cao, Shaojie Bai, Fugen Yao, Liang Dong, Jian Xu, and Li Xiao. ATPO: Adaptive tree policy optimization for multi-turn medical dialogue. In *International Conference on Learning Representations*, pp. 31272–31292, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/34b3a40ec9752c1ae48fe85fef8fe8dc-Paper-Conference.pdf.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. *arXiv preprint arXiv:2505.10978*, 2025. URL https://arxiv.org/abs/2505.10978.

Zhaolin Gao, Joongwon Kim, Wen Sun, Thorsten Joachims, Sid Wang, Richard Yuanzhe Pang, and Liang Tan. Prompt curriculum learning for efficient LLM post-training. *arXiv preprint arXiv:2510.01135*, 2025. URL https://arxiv.org/abs/2510.01135.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. ToRA: A tool-integrated reasoning agent for mathematical problem solving. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.17452.

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 COLING*, pp. 6609–6625, 2020.

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for LLM agent reinforcement learning. *arXiv preprint arXiv:2509.21240*, 2025.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025. URL https://arxiv.org/abs/2503.09516.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 29557–29590. PMLR, 2025. URL https://proceedings.mlr.press/v267/kazemnejad25a.html.

Junbo Li, Peng Zhou, Rui Meng, Meet P. Vadera, Lihong Li, and Yang Li. Turn-PPO: Turn-level advantage estimation with PPO for improved multi-turn RL in agentic LLMs. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 6227–6243, 2026. doi: 10.18653/v1/2026.findings-eacl.328. URL https://aclanthology.org/2026.findings-eacl.328/.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of EMNLP*, pp. 5687–5711, 2023.

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

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Hieu Tran, Zonghai Yao, and Hong Yu. Exploiting tree structure for credit assignment in RL training of LLMs. *arXiv preprint arXiv:2509.18314*, 2025. URL https://arxiv.org/abs/2509.18314.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Daoyu Wang, Qingchuan Li, Mingyue Cheng, Jie Ouyang, Shuo Yu, Chunli Liu, Shijin Wang, Qi Liu, and Enhong Chen. CAPO: Critic-guided action-aligned policy optimization for advancing LLM agent capabilities. *arXiv preprint arXiv:2604.18401*, 2026a.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Tianyi Wang, Yixia Li, Long Li, Yibiao Chen, Shaohan Huang, Yun Chen, Peng Li, Yang Liu, and Guanhua Chen. Sppo: Sequence-level ppo for long-horizon reasoning tasks, 2026b. URL https://arxiv.org/abs/2604.08865.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. 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.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024a.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*, 2024b. URL https://arxiv.org/abs/2405.15793.

Zhicheng Yang, Zhijiang Guo, Yinya Huang, Xiaodan Liang, Yiwei Wang, and Jing Tang. TreeRPO: Tree relative policy optimization. *arXiv preprint arXiv:2506.05183*, 2025. 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 EMNLP*, pp. 2369–2380, 2018.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

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://arxiv.org/abs/2210.03629.

Zhiyuan Yao, Yi-Kai Zhang, Yuxin Chen, Yueqing Sun, Zishan Xu, Yu Yang, Tianhao Hu, Qi Gu, Hui Su, and Xunliang Cai. CoBA-RL: Capability-oriented budget allocation for reinforcement learning in LLMs. *arXiv preprint arXiv:2602.03048*, 2026. URL https://arxiv.org/abs/2602.03048.

Yufeng Yuan, Yu Yue, Ruofei Zhu, Tiantian Fan, and Lin Yan. What's behind PPO's collapse in long-CoT? value optimization holds the secret. *arXiv preprint arXiv:2503.01491*, 2025. URL https://arxiv.org/abs/2503.01491.

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

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

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

Xuhui Zheng, Kang An, Ziliang Wang, Yuhang Wang, and Yichao Wu. StepSearch: Igniting LLMs search ability via step-wise proximal policy optimization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21805–21830, 2025c. URL https://aclanthology.org/2025.emnlp-main.1106v1.pdf.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023. URL https://arxiv.org/abs/2310.04406.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn RL. *arXiv preprint arXiv:2402.19446*, 2024. URL https://arxiv.org/abs/2402.19446.

Heming Zou, Qi Wang, Yun Qu, Yuhang Jiang, Lizhou Cai, Yixiu Mao, Ru Peng, Xin Xu, Weijie Liu, Kai Yang, Saiyong Yang, and Xiangyang Ji. TRACE: A unified rollout budget allocation framework for efficient agentic reinforcement learning. *arXiv preprint arXiv:2606.11119*, 2026. URL https://arxiv.org/abs/2606.11119.

A Experimental Details

A.1 Evaluation Sets and Metrics

Evaluation covers the four benchmarks in Tree-GRPO's Multi-hop QA setting (Ji et al., 2025). Table A1 summarizes the public split sizes and documented 3B evaluation subset sizes. HotpotQA evaluation samples come from a held-out split; the other three benchmarks measure cross-dataset transfer.

Table A1: Public Multi-hop QA split sizes and documented 3B evaluation subset sizes.

Benchmark	Public split	3B evaluation	Relation to RL data
HotpotQA	7,405	2,000	Held-out, same dataset
2WikiMultiHopQA	12,576	2,000	Cross-dataset
MuSiQue	2,417	2,000	Cross-dataset
Bamboogle	125	125	Cross-dataset

Reinforcement learning uses questions from the HotpotQA training split. The online budget rule in Section 3.4 recomputes easy, medium, and hard groups from the current critic predictions. These online groups are distinct from any fixed grouping used to construct the training dataset.

The evaluation metric is answer EM, reported as percentages. Avg. is the equal-weight arithmetic mean across four benchmarks. Means and differences are computed from unrounded scores and then rounded to two decimal places. Improvements are expressed in EM percentage points.

A.2 Unified Experimental Settings

Models and interaction. All three task settings use Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct actors. TreePPO uses a 3B critic at both actor scales. Table A2 summarizes interaction settings. Multi-hop QA queries Wikipedia through an E5 retriever, returning three passages per search (Wang et al., 2022); unfinished trajectories receive a final generation with retrieval disabled. Its observation and critic input limits are 500 and 1,024 tokens. ALFWorld and WebShop use ReAct with one action per turn and the latest two turns as context. Their training temperature and top-p are both 1.0; ALFWorld evaluation uses temperature 0.4.

Training and evaluation. Multi-hop QA uses the HotpotQA training split and the evaluation sets in Appendix A.1. ALFWorld and WebShop use 1,000/1,024 training tasks and 134/256 test tasks, respectively, with 32 tasks per update, evaluation every five updates, and total budgets of 150/120 updates. Their success rates are averaged across three random seeds. TreePPO fixes the actor for initialization during the first 25 steps in Multi-hop QA and 30 steps in both interactive environments, constructing APVT and training the critic before joint updates. These steps are included in the total budgets.

Optimization. Table A3 compares TreePPO's optimization settings across tasks. Its critic backbone and scalar value head are trained jointly; the head outputs a real-valued prediction. In ALFWorld and WebShop, all methods use actor learning rate 10^{-6}, KL coefficient 0.01, and no entropy bonus; PPO and Turn-PPO use critic learning rate 10^{-5}. The actor ratio-clipping interval is $[0.98, 1.03]$ except for Turn-PPO, which uses $[0.8, 1.2]$ as described in Appendix A.3. PPO, Turn-PPO, and TreePPO use $\gamma = \lambda_A = 1$ in these environments. In tree rollouts, inherited trunk and new continuation tokens contribute to actor loss, while observation tokens are excluded. Multi-hop QA TreePPO uses one actor PPO epoch per update.

Rewards and supervision targets. For Multi-hop QA, the APVT stores answer EM $R(\tau)$, and the current online critic uses the mixed target in Equation 5. In the revised allocation configuration, the actor's terminal reward applies a format-validity gate:

$$r_T^\pi = F(\tau)R(\tau), \qquad F(\tau) \in \{0, 1\}, \qquad r_t^\pi = 0 \quad (t < T). \tag{11}$$

Table A2: Interaction settings across tasks. Multi-hop QA permits four regular interaction cycles followed by a final answer if needed.

Setting	Multi-hop QA	ALFWorld	WebShop
Initial input limit (tokens)	2,048	2,048	4,096
Context window	4,096 tokens	2 turns	2 turns
Generation limit (tokens/turn)	500	512	512
Interaction turns	4 + final answer	40	15

Table A3: TreePPO optimization settings across tasks. Initialization is included in the total training steps.

Setting	Multi-hop QA	ALFWorld	WebShop
Actor learning rate	10^{-6}	10^{-6}	10^{-6}
Critic learning rate	5×10^{-6}	2×10^{-6}	2×10^{-6}
KL coefficient	0.001	0.01	0.01
Actor ratio clipping	$[0.98, 1.03]$	$[0.98, 1.03]$	$[0.98, 1.03]$
GAE (γ, λ_A)	$(1, 1)$	$(1, 1)$	$(1, 1)$
Initialization steps	25	30	30
Current-outcome weight α	0.8	0.8	0.8
Critic-target multiplier	1	10	1

Table A4: Rollouts per task and advantage estimators for the five methods. TreePPO reports average counts.

Method	Multi-hop QA	ALFWorld	WebShop	Advantage estimation
PPO	1	1	1	Token-level GAE
GRPO	6	8	8	Group-relative
Turn-PPO	6	8	8	Turn-level GAE
Tree-GRPO	6	8	8	Group-relative
TreePPO	6 (avg.)	8 (avg.)	8 (avg.)	Turn-level GAE

Here $F(\tau)$ indicates whether the trajectory has valid formatting, and the per-turn format penalty is zero. The actor uses r_t^π for GAE. KL regularization enters the actor loss as a separate term, as defined in Equation 16. The current critic target combines current answer EM with the node mean after excluding any contribution from that execution, using the weights in Equation 5. Evaluation reports answer EM.

For ALFWorld and WebShop, APVT stores binary task success. The same mixed-target rule applies, with critic targets multiplied by ten in ALFWorld and retained on the $[0, 1]$ scale in WebShop. WebShop uses continuous product-match scores for actor rewards. APVT statistics remain on the binary-success scale in both environments.

A.3 Method Configurations and Comparison Protocol

PPO. We use standard token-level PPO (Schulman et al., 2017), sampling one rollout per question in each training iteration. Its critic learns from individual trajectory outcomes and supplies token-level advantages.

GRPO. GRPO (Shao et al., 2024) samples six independent rollouts per Multi-hop QA question and eight per ALFWorld or WebShop task. It computes group-relative advantages from their outcomes, without a learned critic.

Turn-PPO. We follow the turn-level advantage estimation and policy objective of Turn-PPO (Li et al., 2026), with clipping parameter $\epsilon = 0.2$, corresponding to an importance-ratio interval of $[0.8, 1.2]$. To align trajectory counts with TreePPO, we adjust the number of independent rollouts per question to six for Multi-hop QA and eight for ALFWorld and WebShop. Its actor and critic condition on full interaction histories.

Table A5: ALFWorld and WebShop success rates (%, ↑). Scores are averaged across three random seeds for both environments. Bold denotes the best result in each column.

	ALFWorld		WebShop	
Method	3B	7B	3B	7B
PPO	30.85	56.22	64.53	69.50
GRPO	75.87	79.10	66.87	78.57
Turn-PPO	74.38	80.10	75.80	79.73
Tree-GRPO	61.94	66.67	60.20	70.73
TreePPO	**83.33**	**86.07**	**77.37**	**80.53**

Tree-GRPO. We follow Tree-GRPO's tree-sampling procedure (Ji et al., 2025) and compute group-relative advantages from the resulting trajectories. For Multi-hop QA, two trunk trajectories each receive two additional branches, giving six rollouts in total. ALFWorld and WebShop use four initial trunks and four branch continuations, giving eight rollouts in total.

TreePPO. TreePPO adopts Tree-GRPO's initialize-then-expand sampling structure, with critic-guided allocation and APVT-based value learning. It uses two trunks for Multi-hop QA and four for ALFWorld and WebShop, plus four additional branches on average. Its total trajectory counts therefore match Turn-PPO on average (Table A4). The actor conditions on full histories, while the critic uses question–action-prefix inputs. Allocation details appear in Appendix A.4.

Trajectory counts and token budgets. Matching trajectory counts does not equalize token costs because tree branches reuse their trunk prefixes. In our Multi-hop QA setting, generating two trunks and four additional branches consumes approximately as many rollout tokens as four independent chain rollouts. We nevertheless retain six independent rollouts for Turn-PPO, rather than reducing its count to match this approximate token budget. Turn-PPO consequently uses more rollout-generation tokens and larger critic inputs than TreePPO. We retain this modest resource advantage for the baseline rather than enforce equal token budgets.

Training diagnostics and result reporting. For the 7B-actor comparison in Figure 3(a,b), mean value loss and explained variance are computed over steps 101–150. Panels (c,d) evaluate difficulty groups at steps 100 and 150, ranking tasks by the mean critic prediction over their trunk prefix positions. On Multi-hop QA with Qwen2.5-Instruct actors, GRPO and Tree-GRPO are unstable: their training rewards decline sharply after approximately 100 updates (Figure 6). We therefore report their best observed evaluation checkpoints rather than their final checkpoints. The other methods use their final checkpoints. Table 1 reports the current results, with unavailable entries shown as dashes and Multi-hop QA averages computed over its four benchmarks.

Literature references. Search-R1 and StepSearch scores are taken from Table 2 of StepSearch (Zheng et al., 2025c) as external references. Their reported training data differ: Search-R1 uses approximately 170k NQ and HotpotQA examples, whereas StepSearch uses approximately 19k MuSiQue examples with additional process supervision. These entries retain the original training and evaluation settings.

A.4 Cross-Question Budget Configuration

The current cross-question allocation follows Section 3.4 for Multi-hop QA, ALFWorld, and WebShop. For each task, the pre-update critic's predictions at all evaluated trunk prefix positions are averaged to obtain its score. Tasks are ranked by this score and divided into approximately equal-sized hard, medium, and easy groups. The base branch-budget unit is $b = 3$ in all three settings. Each medium-difficulty task receives six additional branch rollouts, while each easy or hard task receives three. These groups are recomputed as training proceeds.

Here $b = 3$ is the base branch budget in Equation 7. The total assigned branch budget is $b(|\mathcal{Q}_{\text{easy}}| + 2|\mathcal{Q}_{\text{medium}}| + |\mathcal{Q}_{\text{hard}}|)$ and therefore depends on the group sizes. Within a task, the tree-rollout procedure samples eligible branch origins as described in Appendix B.3. Equal trajectory counts need not imply equal token or environment-interaction costs.

Table A6: Additional branch budgets under the three-group allocation rule. Initial trunk rollouts are excluded.

Current difficulty group	Additional branch budget
Easy	$b = 3$
Medium	$2b = 6$
Hard	$b = 3$

Trunk and average branch counts. TreePPO samples two initial trunk trajectories per question for Multi-hop QA and four per task for ALFWorld and WebShop. Each setting uses an average of four additional branch rollouts, giving average total budgets of six for Multi-hop QA and eight for ALFWorld and WebShop. Under the $3/6/3$ rule, the average additional budget is $3 + 3f_{\text{medium}}$, where f_{medium} is the fraction of tasks in the medium group. An average of four therefore corresponds to a medium-group fraction of one third. For Multi-hop QA, easy and hard questions receive five total trajectories and medium questions receive eight; for ALFWorld and WebShop, the corresponding totals are seven and ten. The average branch count is distinct from the base unit b.

Configuration and result mapping. The TreePPO rows in Table 1 use the current mixed-target configuration: the current trajectory outcome has weight 0.8, and the matched-node mean after excluding that trajectory has weight 0.2. The 3B Multi-hop QA row is the full TreePPO (A) configuration in Section 4.3. Earlier Randgen scores and diagnostic histories are not the current main-table evaluation.

A.5 Prompt Templates

We provide the prompt templates for Multi-hop QA, ALFWorld, and WebShop below. Placeholders in braces are filled with task-specific information at each interaction step. For ALFWorld and WebShop, we include both variants with and without interaction history. Colors and line wrapping are for presentation only.

Multi-hop QA Prompt Template

Answer the given question. You must conduct reasoning inside <think> and </think> first every time you get new information. After reasoning, if you find you lack some knowledge, you can call a search engine by <search> query </search> and it will return the top searched results between <information> and </information>. You can search as many times as your want. If you find no further external knowledge needed, you can directly provide the answer inside <answer> and </answer>, without detailed illustrations. For example, <answer> Beijing </answer>.Question:

ALFWorld Prompt Template (Without History)

You are an expert agent operating in the ALFRED Embodied Environment. Your current observation is: {current_observation} Your admissible actions of the current situation are: [{admissible_actions}].
Now it's your turn to take an action. You should first reason step-by-step about the current situation. This reasoning process MUST be enclosed within <think> </think> tags. Once you've finished your reasoning, you should choose an admissible action for current step and present it within <action> </action> tags.

ALFWorld Prompt Template (With History)

You are an expert agent operating in the ALFRED Embodied Environment. Your task is to: {task_description} Prior to this step, you have already taken {step_count} step(s). Below are the most recent {history_length} observations and the corresponding actions you took: {action_history} You are now at step {current_step} and your current ob-

ALFWorld Prompt Template (With History) (continued)

servation is: {current_observation} Your admissible actions of the current situation are: [{admissible_actions}].
Now it's your turn to take an action. You should first reason step-by-step about the current situation. This reasoning process MUST be enclosed within <think> </think> tags. Once you've finished your reasoning, you should choose an admissible action for current step and present it within <action> </action> tags.

WebShop Prompt Template (Without History)

You are an expert autonomous agent operating in the WebShop e-commerce environment. Your task is to: {task_description}. Your current observation is: {current_observation}. Your admissible actions of the current situation are: [{available_actions}].
Now it's your turn to take one action for the current step. You should first reason step-by-step about the current situation, then think carefully which admissible action best advances the shopping goal. This reasoning process MUST be enclosed within <think> </think> tags. Once you've finished your reasoning, you should choose an admissible action for current step and present it within <action> </action> tags.

WebShop Prompt Template (With History)

You are an expert autonomous agent operating in the WebShop e-commerce environment. Your task is to: {task_description}. Prior to this step, you have already taken {step_count} step(s). Below are the most recent {history_length} observations and the corresponding actions you took: {action_history} You are now at step {current_step} and your current observation is: {current_observation}. Your admissible actions of the current situation are: [{available_actions}].
Now it's your turn to take one action for the current step. You should first reason step-by-step about the current situation, then think carefully which admissible action best advances the shopping goal. This reasoning process MUST be enclosed within <think> </think> tags. Once you've finished your reasoning, you should choose an admissible action for current step and present it within <action> </action> tags.

B Method Implementation Details

We use the notation of Section 3, with k indexing training iterations.

B.1 Action Representation and Incremental Matching

Action representation. Actions are normalized by lowercasing, trimming, and collapsing whitespace for node lookup; critic inputs retain their original text. Each node is identified by its full parent path, and the critic input at turn t maps to the node before action a_t. Multi-hop QA search prefixes are serialized as the question followed by the ordered search queries. Semantic matching is restricted to Multi-hop QA search actions; other actions use normalized exact matching.

For Multi-hop QA search actions, semantic matching uses E5-base-v2. Queries receive the `query:` prefix and are encoded with a 64-token window; masked mean pooling and L2 normalization produce unit vectors. The cosine-similarity threshold is $\eta = 0.95$. A node's representative query and vector are set at creation; later merges add aliases while retaining that vector. The same query under different parent paths is looked up separately within each parent.

Algorithm 1 specifies the parent-constrained lookup and insertion procedure.

Tree storage. APVT statistics are maintained separately from model parameters. A retained store accumulates outcomes across iterations, whereas a rebuilt store contains only current-batch outcomes. Interaction records used for expansion are described in Appendix B.3.

Algorithm 1 Parent-constrained prefix matching

Require: Actions $a_{1:m}$, APVT $\mathcal{T}_q$, threshold η
$u \leftarrow \mathrm{root}(\mathcal{T}_q)$; $P \leftarrow (u)$
for $a \in a_{1:m}$ **do**
  $v \leftarrow \mathrm{ExactChild}(u, \mathrm{norm}(a))$
  if $v = \varnothing$ **and** a is a Multi-hop QA search action **then**
    $v \leftarrow \mathrm{BestSemanticChild}(u, a)$
    if $v = \varnothing$ **or** $\mathrm{sim}(a, v) < \eta$ **then**
      $v \leftarrow \mathrm{NewChild}(u, a)$
    end if
  end if
  if $v = \varnothing$ **then**
    $v \leftarrow \mathrm{NewChild}(u, a)$
  end if
  $P \leftarrow P \oplus v$; $u \leftarrow v$
end for
Ensure: Matched path P

Matching and storage complexity. Exact lookup in a parent's alias table takes amortized $O(1)$ time. On an exact miss, semantic fallback compares an action only with the current parent's children. Let u_{i-1} be the parent before inserting action i. For m actions with d-dimensional embeddings, semantic lookup requires

$$O\left(d \sum_{i=1}^{m} |\,\mathrm{Ch}(u_{i-1})|\right) \tag{12}$$

time, followed by $O(m)$ statistic updates. Storage grows linearly with the retained nodes, aliases, and representative embeddings.

B.2 Outcome Statistics and Critic Samples

Outcome counting and updates. Let $\mathcal{W}(\tau)$ contain all nodes on the inserted path of execution τ, including its inherited prefix and new suffix. Each execution contributes once per node in this set. For the current executions $\mathcal{B}_{k,q}$, superscripts $k,-$ and $k,+$ denote statistics before and after insertion:

$$\begin{aligned} N_u^{k,+} &= N_u^{k,-} + \sum_{\tau \in \mathcal{B}_{k,q}} \mathbb{I}[u \in \mathcal{W}(\tau)], \\ S_u^{k,+} &= S_u^{k,-} + \sum_{\tau \in \mathcal{B}_{k,q}} \mathbb{I}[u \in \mathcal{W}(\tau)] R(\tau), \\ \mu(u) &= S_u^{k,+} / N_u^{k,+}. \end{aligned} \tag{13}$$

New nodes start with zero counts and sums; existing statistics follow the retention setting above. Counts represent inserted executions, not independent root samples, and node means are computed from outcome sums rather than averages of child means. Critic initialization uses Equation 6 with prefix–node-mean pairs from the preconstructed APVT.

Excluding the current execution's contribution. After inserting the complete batch, fix the node statistics N_u, S_u and define

$$\begin{aligned} o_{\tau,u} &= \mathbb{I}[u \in \mathcal{W}(\tau)], \qquad N_{-\tau}(u) = N_u - o_{\tau,u}, \\ \mu_{-\tau}(u) &= \frac{S_u - o_{\tau,u} R(\tau)}{N_{-\tau}(u)}, \qquad N_{-\tau}(u) > 0. \end{aligned} \tag{14}$$

This estimate is used in Equation 5, including its fallback when $N_{-\tau}(u) = 0$. Statistics and targets remain fixed during critic optimization.

Answer samples. For a trajectory with a parseable answer a^{ans}, we concatenate its final valid search prefix and answer text into input x^{ans}, labeled with $R(\tau)$. This sample joins the prefix samples in $\mathcal{S}_k$. APVT matching paths remain sequences of search queries. Answer samples support value regression; the terminal bootstrap value in GAE is separately set to zero.

Repeated samples. The training set is constructed from turn occurrences and retains repeated prefixes across trajectories. The base configuration shares node-mean labels. The mixed configuration shares statistics but incorporates each trajectory's outcome, allowing targets at the same node to differ. Regression averages over the resulting training items. Task-specific target-scale conversions are specified in Appendix A.2.

B.3 Within-Question Branch Selection and State Restoration

Branch budgets and random selection follow Section 3.2 and Appendix A.4.

Restoring original interaction states. An eligible branch origin must be nonterminal, have a restorable interaction record, and have remaining interaction budget. Each candidate retains its actual full context and environment restoration state. Restoration resets the environment and replays the recorded actions, checking that the resulting observations match the saved record before sampling a new continuation. Terminal endpoints contribute return statistics but are not expansion candidates. Candidates matching the same APVT node retain distinct original identities. Each suffix is concatenated with its inherited context to form a complete training trajectory; root rollouts are complete by construction. Policy losses use the valid actor-token mask, and outcome insertion follows Appendix B.2.

B.4 Turn-Level Probability Ratios and Policy Loss

Advantages and task weights follow Section 3.5. The task coefficient applies only to the policy surrogate, leaving the KL reduction unchanged; earlier base configurations use $c_q = 1$. Under trajectory averaging, the allocation configuration assigns each question aggregate coefficient $N_q c_q / N = 1/Q$.

Token-specific gradients. Let $m_{\tau,t} = \log \rho_{\tau,t}$ for the turn-level ratio in Equation 9, and let $\ell_{\theta,\tau,t,j}$ be the log-probability of actor token j conditioned on its full preceding context. The implemented ratio is

$$\widetilde{\rho}_{\tau,t,j} = \exp\left(\min\left\{\ell_{\theta,\tau,t,j} - \text{sg}[\ell_{\theta,\tau,t,j}] + \text{sg}[m_{\tau,t}],\, 10\right\}\right). \tag{15}$$

Here sg denotes stop-gradient. When the numerical cap is inactive, the forward value equals $\rho_{\tau,t}$, and gradients propagate through the current token's $\ell_{\theta,\tau,t,j}$. Valid tokens in a turn thus share a forward probability ratio while retaining their individual token gradients.

Actor-token mask and reduction. Equation 10 averages over valid actor tokens within each trajectory and then over trajectories. Across all three tasks, the mask includes both inherited prefix tokens and newly generated continuation tokens; a shared trunk therefore contributes again through each branch. Environment observations are excluded. Probability-ratio segments are contiguous valid-token spans, while advantages are assigned by parsed turn; parsing and mask rules handle malformed text.

Complete actor objective. The actor maximizes the clipped surrogate in Equation 10 with a KL penalty:

$$J_\pi(\theta) = J_{\text{clip}}(\theta) - \beta_{\text{KL}} \mathcal{L}_{\text{KL}}(\theta). \tag{16}$$

Equivalently, the implementation minimizes $-J_\pi(\theta)$.

B.5 Complete Training Procedure

Algorithm 2 summarizes the order of data collection, target construction, and parameter updates.

Algorithm 2 One online training iteration of TreePPO

Require: Question batch, sampling policy, pre-update critic $\bar{V}$, retained APVTs, base budget $b = 3$, scoring and grouping configuration

1: Fix the policy and critic; sample initial trunk trajectories in parallel (two per question for QA; four per task for ALFWorld and WebShop).
2: Extract original restoration candidates and evaluate their action prefixes with $\bar{V}$.
3: Aggregate prefix scores and partition tasks into $\mathcal{Q}_{\text{easy}}$, $\mathcal{Q}_{\text{medium}}$, and $\mathcal{Q}_{\text{hard}}$.
4: Assign $B_q = 2b = 6$ to medium tasks and $B_q = b = 3$ to easy and hard tasks, excluding initial trunk rollouts.
5: Sample eligible branch origins uniformly within each task and restore their original histories and environment states.
6: Generate continuations under the fixed policy and assemble complete training trajectories.
7: Merge action paths into APVTs. Count each execution once at each node in its write range $\mathcal{W}(\tau)$.
8: Construct fixed mixed targets using Equation 5, excluding only actual contributions; apply the environment target-scale mapping in Appendix A.2 when applicable, and append the answer samples described in Appendix B.2.
9: Compute turn-level GAE with $\gamma = \lambda_A = 1$, fixed $\bar{V}$, and each trajectory's actor rewards.
10: Scale advantages by $N/(QN_q)$ in the policy surrogate as in Section 3.5.
11: Optimize the single critic regression loss and the actor objective separately; retain updated APVTs.

Ensure: Updated actor, critic, and APVTs for the next iteration

C Offline Critic Evaluation and Ablation Protocols

C.1 Critic Training and MC64 Evaluation

Online critic evaluation. We use the step-150 TreePPO and TurnPPO critic checkpoints from the main experiments in Section 4.2, with both critics based on Qwen2.5-3B-Instruct. We fix a set of 256 questions from the HotpotQA training split and generate four trunk trajectories per question. From each root and the search states along the trunks, we sample 64 complete continuations and use the fraction of correct final answers as the value reference. Both frozen critics predict values for the same states, evaluated using MSE, Pearson correlation, and ten-bin ECE.

Offline critic comparison. We use a fixed actor to collect 32 trajectories for each of 2,048 HotpotQA questions and retain the 1,362 questions with both successful and unsuccessful outcomes. For each retained question, we use 24 trajectories for critic training and reserve the remaining 8 for evaluation. Training APVT return counts and sums are computed exclusively from the 24 training trajectories; neither the held-out trajectory outcomes nor the MC64 evaluation outcomes enter these statistics or the critic-training targets. This is a trajectory-level split within each question: training and evaluation share questions and may share action prefixes, but evaluation outcomes are excluded from training.

Both critics initialize from Qwen2.5-3B-Instruct with identical scalar-head initialization. Action+Merged learns aggregated node values from question–action-prefix inputs and also uses additional terminal-outcome samples from the training trajectories. Full-context learns individual trajectory outcomes from full interaction histories. We compare the critics after one training epoch each, matching passes over their respective training sets rather than optimization-update counts.

Offline MC64 references. The 8 held-out trajectories per question provide the test prefixes. From each test prefix, we independently sample 64 complete continuations using the same fixed actor that generated the original trajectories. The fraction of correct final answers serves as the shared Monte Carlo value reference for both critics. Thus, training trajectories and evaluation continuations are generated under a common fixed policy, and the reference values are estimated from newly sampled continuations rather than the original held-out trajectories' terminal outcomes.

Evaluation is performed at the semantic-node level defined by APVT. For each node u, we pool the success counts and continuation counts of its corresponding physical test prefixes:

$$\widehat{V}_u^{\mathrm{MC64}} = \frac{\sum_{p \in \mathcal{P}_u} s_p}{\sum_{p \in \mathcal{P}_u} n_p}, \qquad n_p = 64, \tag{17}$$

where $\mathcal{P}_u$ is the set of test prefixes mapped to node u, and s_p is the number of successful continuations from prefix p. Predictions from each critic are also aggregated within semantic nodes and compared with these MC64 references. The resulting metrics assess node-level value prediction under the fixed actor on questions seen during critic training. MC64 provides a finite-sample estimate of value.

C.2 Results and Metrics

Table A7: Offline critic evaluation against MC64 references after one training epoch per approach. Lower MSE and ECE-10 are better; higher correlations are better.

Critic	Pearson	Within-Q Pearson	Macro Pearson	Macro Spearman	MSE	ECE-10
Action+Merged	**0.830**	**0.657**	**0.488**	**0.394**	**0.0445**	**0.0613**
Full-context	0.698	0.461	0.197	0.201	0.0765	0.0933

Metrics. Pearson measures correlation across semantic nodes. Within-question Pearson removes question-specific means before computing correlation. We additionally report macro Pearson and Spearman correlations, MSE, and ten-bin expected calibration error (ECE-10). Macro Spearman averages per-question rank correlations with equal question weights. All metrics compare critic predictions directly with the semantic-node MC64 references.

C.3 Component Ablation Settings and Results

The 3B ablations in Section 4.3 compare four configurations: (A) full TreePPO; (B) random allocation with tree rollouts and APVT; (C) independent rollouts with APVT; and (D) Turn-PPO. Removing rollout branching in C changes trajectory collection but preserves the persistent APVT, including prefix matching, outcome accumulation across iterations, and critic supervision. Configurations A–C hold the actor fixed during the first 25 steps to construct the initial APVT and warm up the critic, then jointly update the actor and critic. Turn-PPO jointly updates both models from the first step. The startup phase counts toward the same total training-step budget; C versus D therefore compares complete critic-learning procedures, not return aggregation in isolation. Full TreePPO and the random-allocation variant with tree rollout use the same $0.8/0.2$ mixed targets. The unrounded macro-average EM scores, in table order, are 30.900, 30.075, 29.850, and 27.125. Full TreePPO supplies the current 3B main-table result.

C.4 Historical Outcome Reuse Statistics

Figure 5 analyzes historical outcome reuse in the main-experiment 3B TreePPO run. Hit rate is the fraction of non-empty search-action prefix occurrences in the current trunk trajectories that have historical support. Return counts average over matched APVT prefixes, counting repeated matches to the same prefix once within each training step. Both statistics exclude returns from trajectories added in the current step.

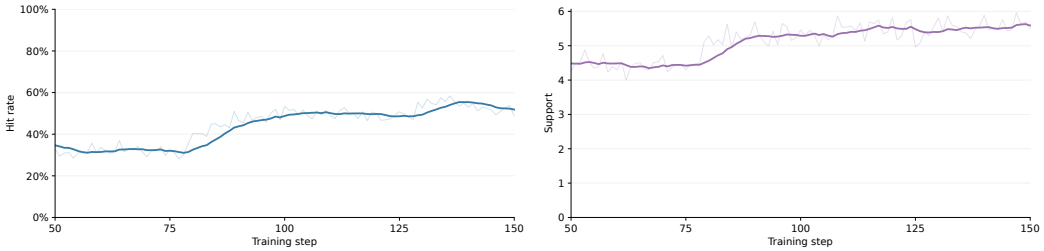


Figure 5: Historical outcome reuse: hit rate (left) and returns per matched prefix (right) during training. Current-step returns are excluded.

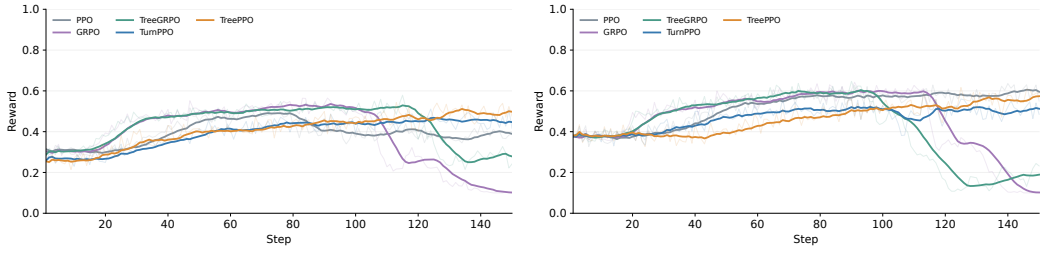


Figure 6: Multi-hop QA training rewards with Qwen2.5-3B-Instruct (left) and Qwen2.5-7B-Instruct (right). GRPO and Tree-GRPO show marked declines late in training.