Reason in Chains, Learn in Trees: Self-Rectification and Grafting for Multi-turn Agent Policy Optimization

Yu Li **Sizhe Tang** **Tian Lan***
Department of Electrical and Computer Engineering, George Washington University
{yul,tlan}@gwu.edu
*Corresponding author

Abstract

Reinforcement learning for Large Language Model agents is often hindered by sparse rewards in multi-step reasoning tasks. Existing approaches like Group Relative Policy Optimization treat sampled trajectories as independent chains, assigning uniform credit to all steps in each chain and ignoring the existence of critical steps that may disproportionally impact reasoning outcome. In this paper, we propose T-STAR (Tree-structured Self-Taught Agent Rectification), a framework that recovers the latent correlated reward structure across seemingly independent trajectories. Specifically, we consolidate trajectories into a unified Cognitive Tree by identifying and merging functionally similar steps/nodes. It enables an Introspective Valuation mechanism that back-propagates trajectory-level rewards through the tree to obtain a new notion of variance-reduced relative advantage at step-level. Using the Cognitive Tree, we also develop In-Context Thought Grafting to synthesize corrective reasoning by contrasting successful and failed branches at critical divergence points/steps. Our proposed Surgical Policy Optimization then capitalizes on the rich policy gradient information concentrated at these critical points/steps through a Bradley-Terry type of surgical loss. Extensive experiments across embodied, interactive, reasoning, and planning benchmarks demonstrate that T-STAR achieves consistent improvements over strong baselines, with gains most pronounced on tasks requiring extended reasoning chains.

1 Introduction

Reinforcement learning has emerged as a powerful post-training paradigm for Large Language Models, enabling capabilities ranging from planning to complex mathematical reasoning (Wang et al., 2025a; Kumar et al., 2025; Park et al., 2025). Recent advances extend this paradigm to LLM agents with multi-turn reasoning cycles, tackling tasks such as web navigation, embodied control, and tool-augmented question answering (Zeng et al., 2025; Wang et al., 2025b; Chen et al., 2025). These settings typically adopt the ReAct framework (Yao et al., 2022), where agents interleave reasoning thoughts with executable actions across multiple steps, with trajectories (or reasoning chains) often spanning dozens of decisions (Shinn et al., 2023; Yao et al., 2023; Bai et al., 2023). To optimize such agents, methods like Proximal Policy Optimization (PPO) require training separate value networks to estimate advantages, adding computational overhead and potential instability in long-horizon settings (Schulman et al., 2017). Group Relative Policy Optimization (GRPO) addresses this by estimating a notion of relative advantage through comparison within sampled trajectory groups, eliminating the need for value networks while maintaining competitive performance on multi-turn reasoning agents (Guo et al., 2025a). Therefore extracting rich policy gradient information from finite rollouts and sparse rewards is a fundamental challenge (Zheng et al., 2018; Chiappa et al., 2023).

However, existing approaches often treat sampled trajectories as independent chains and thus are oblivious to the latent correlated reward structure across them. We note two key limitations arising from this. First, since sparse rewards are received only at trajectory completion, all steps within each trajectory receive the same credit (despite their functional disparity), while common steps shared by multiple trajectories may receive inconsistent credit (despite their functional equivalence). Existing coarse-grained credit assignment fails to distinguish critical decision points from routine execution steps, making it difficult for agents to learn which specific reasoning choices led to success or failure (Bai et al., 2024; Guo et al., 2025b; Zeng et al., 2025). Second, rich policy-gradient information are likely concentrated at critical decision points/steps that have a disproportional impact on

Findings of the Association for Computational Linguistics: ACL 2026, pages 4673–4686

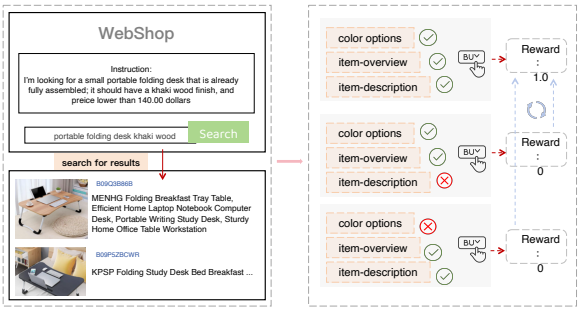

Figure 1: An illustration using WebShop Operation. Three trajectories share identical prefixes but diverge at the attribute verification step, yielding different outcomes. Existing methods with trajectory-level rewards assign inconsistent credit for the prefixes in each trajectory, while critical decision steps fail to be distinguished and utilized for policy optimization.

reasoning outcome which may lead to suboptimal policy update (Kumar et al., 2023; Sun et al., 2023).

To address these limitations, we propose a novel framework, Tree-structured Self-Taught Agent Rectification (T-STAR), that recovers the latent reward structure across seemingly independent rollouts for improved policy optimization. By consolidating trajectories into a unified Cognitive Tree by merging functionally similar steps/nodes, T-STAR enables an Introspective Valuation mechanism that back-propagates trajectory-level rewards through the tree and constructs a variance-reduced relative advantage at each step/node. To capture critical decision points, T-STAR develops In-Context Thought Grafting joining reasoning branches sharing similar prefixes but belonging to different trajectories. It allows T-STAR to contrast successful and failed branches and enables a Surgical Policy Optimization to capitalize on the rich policy gradient information concentrated at these critical divergence steps through a Bradley-Terry type of surgical loss function.

Our proposed T-STAR works without requiring additional rollouts or reward models and can be readily integrated into state-of-the-art methods including GRPO, DAPO, and GiGPO (Guo et al., 2025a; Yu et al., 2025; Feng et al., 2025). When multiple rollouts attempt similar tasks, they frequently traverse functionally similar intermediate states before diverging at critical junctures (Zhang et al., 2025b; Li et al., 2025, 2026a). A failed trajectory may contain a perfectly valid reasoning prefix that, with a single corrected decision at the divergence point, would have succeeded (Huang et al., 2025). Recognizing this latent structure could both reduce gradient variance through aggregation and precisely localize where targeted corrections are needed (Yang et al., 2025). Figure 1 illustrates three diverging trajectories in WebShop, consolidated into a Cognitive Tree. Not only do prefixes receive consistent credit via a variance-reduced relative advantage (by reward backpropagation in the tree), critical divergence point (at the attribute verification step) is identified and is leveraged to produce a new policy gradient component from Bradley-Terry type of surgical loss.

Comprehensive experiments across embodied, interactive, reasoning, and planning tasks demonstrate consistent improvements over GRPO and variants, with gains most pronounced on tasks requiring extended reasoning chains where trajectory overlap is frequent. Our contributions include: (1) a Cognitive Tree construction enabling variance-reduced advantage estimation through trajectory consolidation and reward back-propagation; (2) a self-taught rectification mechanism that synthesizes step-level supervision at divergence points/steps via thought grafting; and (3) surgical policy optimization targeting critical decision steps.

2 Related Work

RL for LLM Agents. Reinforcement learning has become a key paradigm for training LLM agents (Zhang et al., 2025a). PPO (Schulman et al., 2017) requires value networks that are computationally expensive for long-context agent tasks. Group-based methods like GRPO (Guo et al., 2025a) eliminate value networks through in-group advantage estimation, enabling more efficient training. Subsequent variants such as DAPO (Yu et al., 2025) and GiGPO (Feng et al., 2025) further improve training stability. These methods typically treat sampled trajectories as independent chains and suffers from the two limitations mentioned.

Self-Improvement in Reasoning. Sparse rewards in long-horizon tasks pose significant challenges for effective credit assignment (Andrychowicz et al., 2017). While Process Reward Models offer step-level supervision, they rely on expensive human annotation (Lightman et al., 2023), prompting valid alternatives such as automatic supervision via Monte Carlo estimation (Uesato et al., 2022; Wang et al., 2022) or inference-time tree search (Li et al., 2026b). However, these structural approaches primarily optimize inference rather than training dynamics. To enable persistent ca-

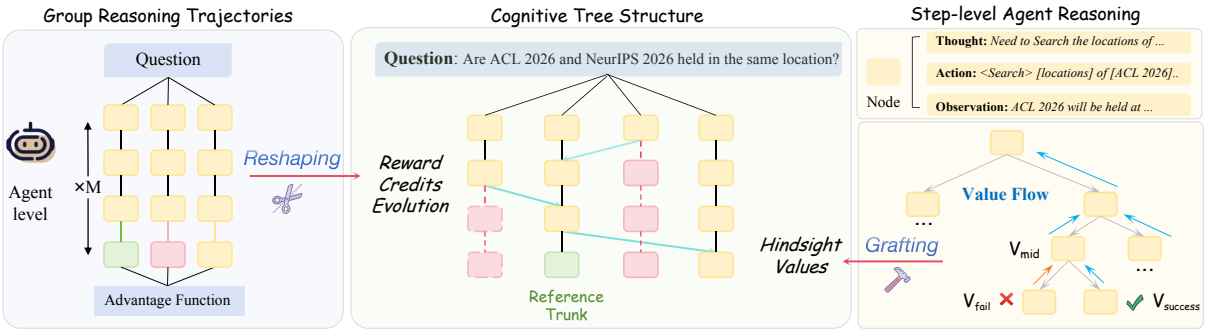

Figure 2: Overview of T-STAR. Given M sampled trajectories per task, T-STAR consolidates them into a Cognitive Tree by merging functionally equivalent nodes, then computes structural values via Bellman backup that propagate downstream success rates to intermediate nodes. This enables trajectory stitching where successful reasoning steps receive appropriate credit even within failed rollouts. At divergence points where children exhibit large value differences, the agent performs self-rectification by generating corrective reasoning that transfers successful logic to failed contexts, producing preference pairs for surgical policy optimization.

pability gains, self-improvement paradigms have emerged, ranging from inference-time verbal feedback in Reflexion (Shinn et al., 2023) to bootstrapping rationales in STaR (Zelikman et al., 2022) and iterative RL updates (Huang et al., 2023; Gulcehre et al., 2023). Despite these advancements, existing methods typically operate either at the coarse trajectory level or strictly during inference (Madaan et al., 2023).

3 Methodology

We introduce T-STAR (Tree-structured Self-Taught Agent Rectification), a framework that addresses sparse supervision in multi-step agent RL through variance-reduced advantage estimation. Rather than modeling sampled trajectories as independent chains, T-STAR consolidates them into a Cognitive Tree that exposes shared decision structure and locates critical divergence steps. At these points, we enable agent self-rectification through thought grafting, where the agent synthesizes corrective reasoning by contrasting successful and failed branches.

The overall pipeline is illustrated in Figure 2. The framework is structured around three core pillars: (1) constructing a Cognitive Tree that reveals shared reasoning structure and identifies divergence points across trajectories, (2) performing Introspective Valuation through Q-tree-based assignment, combined with In-Context Thought Grafting where the agent actively rectifies failed reasoning paths, and (3) applying Surgical Policy Optimization using the synthesized stepwise preferences.

3.1 Constructing the Cognitive Tree

We adopt the ReAct framework (Yao et al., 2022) where the agent engages in multi-turn Thought-Action-Observation cycles. At each step $t = 0, 1, \ldots, T-1$, the LLM generates a thought z_t and a discrete textual action a_t based on context s_t, then obtains observation o_t through tool use. A complete T-step trajectory is:

$$\tau = \{(s_0, z_0, a_0, o_0), \ldots, (s_{T-1}, z_{T-1}, a_{T-1}, o_{T-1})\} \quad (1)$$

GRPO (Guo et al., 2025a) samples M trajectories $\{\tau_i\}_{i=1}^{M}$ per task and estimates advantages through in-group comparison $\hat{A}_{\text{GRPO}}(\tau_i) = (R_i - \bar{R})/\sigma_R$, where $\bar{R} = \frac{1}{M}\sum_{i=1}^{M} R_i$ and σ_R are computed at each trajectory group, with sparse binary reward $R(\tau) \in \{0, 1\}$ received only upon task completion. There are two limitations: $\hat{A}_{\text{GRPO}}(\tau_i)$ remains constant across all steps, failing to distinguish critical decision points from routine steps; and treating trajectories as independent chains discards the opportunity to reduce variance when multiple trajectories share common reasoning prefixes before diverging.

To address these limitations, we consolidate independent chains into a Cognitive Tree $\mathcal{T} = (V, E)$ that serves two purposes: exposing shared decision structure to enable variance reduction, and locating divergence points where agent self-rectification is most valuable. Each node $v \in V$ represents a cognitive state characterized by tuple (z, a, o). The tree is constructed by identifying functionally equivalent nodes across trajectories.

For each trajectory $\tau_i = \{v_0^{(i)}, v_1^{(i)}, \ldots, v_{T_i}^{(i)}\}$, we define node depth as its position in the trajec-

tory, with V_d denoting all nodes at depth d. The merging process is governed by two compatibility predicates. For functional equivalence, we compare policy distributions over next actions via KL divergence:

$$D_{\mathrm{KL}}(v_i \| v_j) = \sum_a \pi_\theta(a|s_i) \log \frac{\pi_\theta(a|s_i)}{\pi_\theta(a|s_j)} \quad (2)$$

Since exact computation over the entire vocabulary is intractable, we estimate this value via Monte Carlo sampling. Two nodes are considered functionally equivalent if the estimated $D_{\mathrm{KL}}(v_i \| v_j) < \epsilon_{\mathrm{kl}}$. For historical compatibility, we extract state-modifying actions $\mathcal{S}(v) = \{a \in \mathcal{H}(v) : \text{modifies\_state}(a)\}$ from node history. Two nodes are historically compatible if $\mathcal{S}(v_i) = \mathcal{S}(v_j)$, ensuring they operate on equivalent knowledge states.

At each depth d, we compute a compatibility graph $G_d = (V_d, E_d)$ where edge (v_i, v_j) exists if both predicates hold, then merge nodes within each connected component. After merging, edge weights capture empirical transition frequencies:

$$w(v \to v') = \frac{|\mathcal{T}(v \to v')|}{|\mathcal{T}(v)|} \quad (3)$$

where $\mathcal{T}(v \to v')$ is the set of trajectory indices traversing this transition and $\mathcal{T}(v)$ is the set of all trajectories passing through v. The resulting tree $\mathcal{T}$ encodes shared decision structure: nodes with multiple children represent divergence points where different trajectories made different reasoning choices, providing the structural foundation for subsequent valuation and rectification.

3.2 Introspective Valuation and Self-Taught Rectification

Q-tree Valuation We define the Q-tree value function $Q_{\text{tree}} : V \to \mathbb{R}$ via Bellman backup:

$$Q_{\text{tree}}(v) = \gamma \sum_{v' \in C(v)} w(v \to v') Q_{\text{tree}}(v') \quad (4)$$

where $R(v) \in \{0, 1\}$ is terminal reward, $C(v)$ the children of v, $w(v \to v')$ the edge weights from Eq. (3), and $Q_{\text{tree}}(v) = R(v)$ if v is the leaf. Based on Q-tree values, we define per-node advantages:

$$\hat{A}_{\text{tree}}(v) = \frac{Q_{\text{tree}}(v) - \bar{R}}{\sigma_R} \quad (5)$$

using the same mean $\bar{R}$ and normalization σ_R as GRPO. We note that $Q_{\text{tree}}(v)$ can be computed

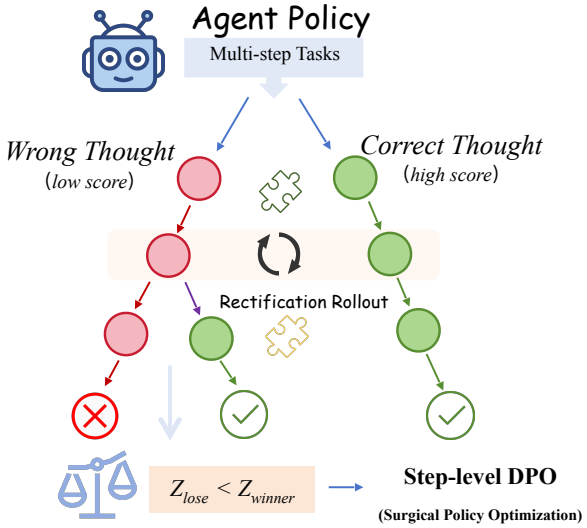

Figure 3: Thought grafting mechanism. At divergence points identified by large value spreads among children, the agent observes contrasting outcomes between successful and failed branches, then generates rectified reasoning that transfers the successful logic to the failed context. The resulting preference pairs provide step-level supervision for surgical policy optimization.

by back-propagating the trajectory-level rewards through the tree as shown in Algorithm 1.

Consider the set of sampled trajectories $\{\tau_i\}_{i=1}^M$. For any node v, let $\mathcal{T}(v)$ denote the set of trajectory indices traversing v (as defined in Eq. 3), with cardinality $k_v = |\mathcal{T}(v)|$. By substituting the Q-tree value definition into Eq. (5), we can express the node-level advantage as $\hat{A}_{\text{tree}}(v) = \frac{1}{k_v} \sum_{i \in \mathcal{T}(v)} \frac{R_i - \bar{R}}{\sigma_R}$. Assuming that trajectory completions are independent conditioned on state v, we derive the following properties relating node-level advantage $\hat{A}_{\text{tree}}(v)$ to GRPO relative advantage:

Lemma 1 (Aggregation and Variance Reduction)**.** *The tree-based node advantage $\hat{A}_{tree}(v)$ satisfies:*

$$\hat{A}_{tree}(v) = \frac{1}{k_v} \sum_{i \in \mathcal{T}(v)} \hat{A}_{GRPO}(\tau_i), \quad (6)$$

and its variance satisfies:

$$Var(\hat{A}_{tree}(v)) = \frac{1}{k_v} Var(\hat{A}_{GRPO}(\tau_i)). \quad (7)$$

Lemma 1 shows that for nodes shared across k trajectories, the tree-based advantage is the average of standard GRPO relative advantage received from different trajectories. Thus, it achieves $1/k$ variance reduction , i.e., $\mathrm{Var}(\hat{A}_{\text{tree}}(v)) = \frac{1}{k}\mathrm{Var}(\hat{A}_{\text{GRPO}})$, while remaining the same as GRPO for nodes belonging to a single trajectory when

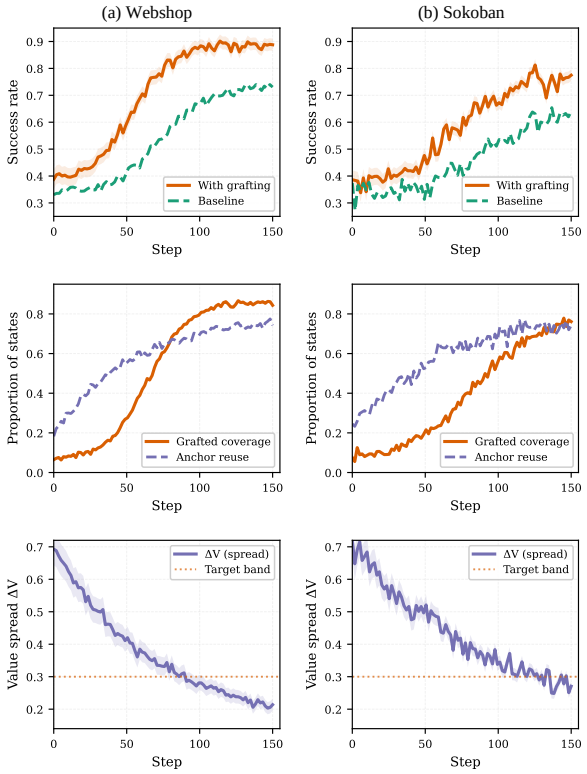

Figure 4: Training dynamics of T-STAR on WebShop and Sokoban, showing success rate, grafting coverage and anchor reuse, and value spread ΔV at divergence points across training iterations.

$k = 1$. Hence, a more stable gradient signal for shared segments is provided. And Q-tree values also identify divergence points that nodes whose children exhibit large value differences indicate where reasoning quality determined outcomes.

Self-Taught Rectification Based on the cognitive tree, we identify divergent nodes as those with children exhibiting large value spreads:

$$\Delta V(v) = \max_{v' \in C(v)} Q_{\text{tree}}(v') - \min_{v'' \in C(v)} Q_{\text{tree}}(v'') > \delta \tag{8}$$

Let V_{div} denote the set of divergent nodes, with v^+, v^- denoting the highest-value and lowest-value children respectively. These nodes represent critical decision points where different reasoning choices led to different outcomes.

At each divergent node $v \in V_{\text{div}}$, the agent performs self-rectification through an additional rollout, as illustrated in Figure 3. Given the shared parent context s and observing that $v^+ = (z^+, a^+, o^+)$ succeeded while $v^- = (z^-, a^-, o^-)$ failed, the agent generates a rectified thought z_{rect} that incorporates the successful reasoning principle from z^+. Since the number of divergent nodes is limited, this requires only one additional rollout pass. This

Algorithm 1 T-STAR Training Procedure

Input: Policy π_θ, tasks $\mathcal{P}$, group size M, thresholds δ, ϵ_{kl}

Initialize $\pi_{\text{ref}} \leftarrow \pi_\theta$, $\mathcal{D}_{\text{graft}} \leftarrow \emptyset$
for iteration $k = 1$ to K **do**
  for task $p \sim \mathcal{P}$ **do**
    Sample $\{\tau_i\}_{i=1}^{M} \sim \pi_\theta(\cdot|p)$
    Build $\mathcal{T}$ via Eq. (2)-(3), compute Q_{tree} via Eq. (4)
    for $v \in \mathcal{T}$ where $\Delta V(v) > \delta$ **do**
      Generate $z_{\text{rect}} \sim \pi_\theta(\cdot|s, v^+, v^-)$
      $\mathcal{D}_{\text{graft}} \leftarrow \mathcal{D}_{\text{graft}} \cup \{(s, z_{\text{rect}}, z^-)\}$
    end for
  end for
  Update θ via Eq. (10), $\pi_{\text{ref}} \leftarrow \alpha\pi_{\text{ref}} + (1-\alpha)\pi_\theta$
end for

creates the grafting dataset:

$$\mathcal{D}_{\text{graft}} = \{(s, z_{\text{rect}}, z^-, t(v)) : v \in V_{\text{div}}\} \tag{9}$$

where $t(v)$ is the timestep index. Each tuple provides a preference pair (z_{rect}, z^-) grounded in actual outcome differences, synthesizing dense step-level supervision from sparse trajectory rewards.

Figure 4 validates these mechanisms empirically. As training progresses, the value spread ΔV at divergence points decreases steadily, indicating that Q-tree valuation identifies meaningful decision boundaries where the policy gradually learns consistent behavior. Meanwhile, the increasing anchor reuse demonstrates that successful reasoning patterns discovered through grafting transfer effectively across similar contexts, confirming that thought grafting synthesizes generalizable corrections rather than instance-specific fixes.

Surgical Policy Optimization The divergent nodes identified in the cognitive tree represent critical decision points where reasoning quality determined outcomes, which lead to the comparative structure. The preference pairs (z_{rect}, z^-) constructed at these points provide targeted supervision for policy improvement. Therefore based on that, we optimize the overall loss function through a hybrid objective combining trajectory-level and step-level learning:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{GRPO}}(\theta) + \lambda \mathcal{L}_{\text{Surgical}}(\theta) \tag{10}$$

where λ controls the relative weight. The surgical loss implements preference learning via the

Table 1: Performance on interactive and embodied tasks. Success rate (%) for ALFWorld subtasks and score/success for WebShop. T-STAR consistently outperforms across all baselines and architectures.

Type	Method	ALFWorld (Success Rate %)							WebShop	
		Pick	Look	Clean	Heat	Cool	Pick2	All	Score	Succ.
Closed-Source Model										
Prompting	GPT-4o	86.8	74.4	79.1	90.9	93.6	83.1	84.6	84.2	61.7
Prompting	Gemini-1.5-Pro	81.6	68.1	72.9	84.8	88.6	77.7	79.0	79.3	54.7
Qwen2.5-3B-Instruct										
Prompting	ReAct	48.2	40.9	44.8	51.7	55.4	38.6	46.6	50.7	25.3
Prompting	Reflexion	61.2	49.6	55.9	65.2	68.4	45.6	57.7	58.8	32.3
RL Training	GRPO	81.1	69.9	75.7	84.7	88.1	73.4	78.8	64.2	36.2
RL Training	+ T-STAR	82.4	**75.1**	77.8	86.7	89.2	**78.9**	**81.7**	**68.2**	**39.6**
RL Training	DAPO	84.3	73.4	79.1	87.1	91.1	76.7	82.0	67.8	39.1
RL Training	+ T-STAR	**86.7**	77.2	**82.8**	89.1	92.6	81.9	**85.0**	**71.7**	**42.4**
RL Training	GiGPO	90.1	78.9	85.6	92.1	95.2	83.8	87.6	73.1	41.6
RL Training	+ T-STAR	91.1	**83.2**	87.8	94.1	96.3	**87.9**	**90.1**	**76.3**	**45.1**
Phi-4-mini-instruct-3.8B										
Prompting	ReAct	46.2	39.2	42.3	49.6	53.2	37.3	44.6	49.8	23.8
Prompting	Reflexion	58.6	47.1	54.2	62.1	66.2	42.9	55.2	56.4	30.7
RL Training	GRPO	79.7	67.3	73.3	82.9	86.8	70.8	76.8	62.3	34.1
RL Training	+ T-STAR	81.3	**73.2**	76.8	84.4	88.1	**76.8**	**80.1**	**65.2**	**38.3**
RL Training	DAPO	82.6	71.9	77.9	85.9	89.9	75.7	80.7	66.1	37.4
RL Training	+ T-STAR	**85.9**	75.6	**81.9**	87.8	91.2	80.6	**83.8**	**69.2**	**41.1**
RL Training	GiGPO	88.7	77.3	84.2	90.6	94.2	82.1	86.2	71.3	41.1
RL Training	+ T-STAR	90.1	**81.7**	87.3	92.7	95.7	**86.8**	**89.0**	**74.2**	**44.3**

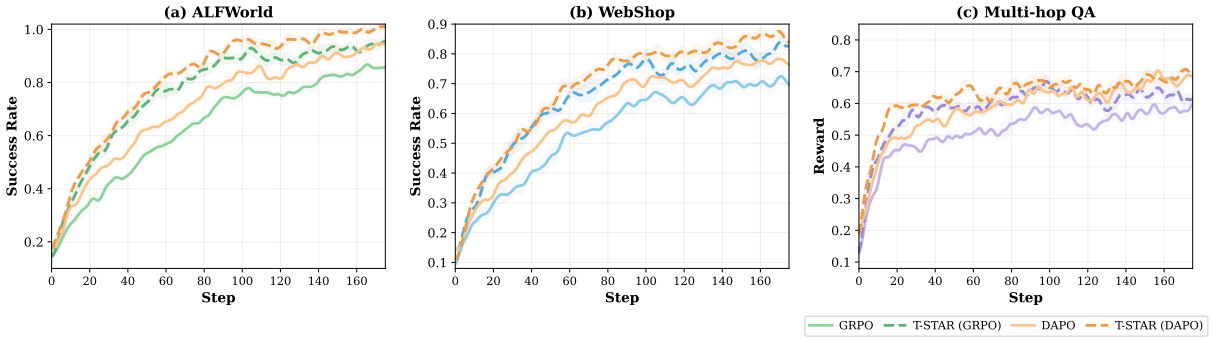

Figure 5: Training dynamics on (a) ALFWorld, (b) WebShop, and (c) Multi-hop QA. T-STAR achieves faster convergence and higher final performance. Shaded regions indicate standard deviation across three seeds.

Bradley-Terry model:

$$\mathcal{L}_{\text{Surgical}}(\theta) = -\mathbb{E}_{(s,z_{\text{rect}},z^-)\sim\mathcal{D}_{\text{graft}}}\left[\log\sigma(\beta\Delta_\theta)\right] \tag{11}$$

with preference margin:

$$\Delta_\theta = \log\frac{\pi_\theta(z_{\text{rect}}|s)}{\pi_{\text{ref}}(z_{\text{rect}}|s)} - \log\frac{\pi_\theta(z^-|s)}{\pi_{\text{ref}}(z^-|s)} \tag{12}$$

To ensure surgical precision, where step-level optimization always plays a supporting role, gradients from $\mathcal{L}_{\text{Surgical}}$ are masked to affect only the divergence timestep t_{div}, while $\mathcal{L}_{\text{GRPO}}$ provides learning signal across all timesteps. The reference policy π_{ref} is updated via exponential moving average to prevent distribution drift.

Algorithm 1 summarizes the complete training procedure. In T-STAR framework, the agent samples trajectories, constructs the cognitive tree to identify divergence points, operates rectified thoughts and then updates the designed policy.

4 Results

4.1 Experiment Setup

The evaluation spans three task categories with diverse reasoning requirements. For embodied and interactive environments, ALFWorld provides

Table 2: Performance on search-augmented QA tasks (Exact Match). T-STAR shows largest gains on multi-hop reasoning. Best performance (highlighted in bold) is always achieved by T-STAR.

Type	Method	Single-Hop QA (EM)				Multi-Hop QA (EM)				Avg.
		NQ	Trivia	PopQA	Avg.	Hotpot	2Wiki	MusiQ	Bamb	
Closed-Source Model										
Prompting	GPT-4o	59.5	73.0	61.5	64.7	54.0	50.5	47.0	44.5	55.8
Prompting	Gemini-1.5-Pro	56.0	69.5	57.5	61.0	50.0	46.5	43.0	40.5	52.0
Qwen2.5-3B-Instruct										
	Direct	23.5	36.0	25.0	28.2	19.0	16.5	13.0	11.5	21.2
Prompting	ReAct	29.0	43.5	30.5	34.3	26.0	22.5	19.5	17.5	26.7
	Search-o1	32.5	46.5	33.5	37.5	28.5	25.5	21.5	19.5	29.6
RL Training	GRPO	41.0	54.5	41.5	45.7	36.5	32.0	28.0	25.5	37.1
RL Training	+ T-STAR	43.5	56.0	43.5	**47.7**	**40.5**	**35.5**	**30.5**	28.0	**39.7**
RL Training	DAPO	43.0	56.5	43.0	47.5	38.5	34.5	30.0	27.5	39.1
RL Training	+ T-STAR	**45.5**	58.0	45.0	**49.5**	**42.0**	**38.0**	**33.0**	**30.5**	**41.8**
RL Training	GiGPO	45.0	58.5	44.5	49.3	41.0	37.0	32.5	30.0	41.1
RL Training	+ T-STAR	47.0	**60.5**	**46.5**	**51.3**	**45.0**	**41.5**	**35.5**	**33.0**	**44.1**
Phi-4-mini-instruct-3.8B										
	Direct	25.5	38.5	27.5	30.5	20.5	17.5	14.0	12.0	22.4
Prompting	ReAct	31.5	46.0	33.0	36.8	27.5	24.0	21.0	19.0	28.6
	Search-o1	35.0	49.0	36.0	40.0	30.0	27.0	23.0	21.0	31.5
RL Training	GRPO	43.5	57.5	44.0	48.3	38.5	34.0	30.0	27.5	39.4
RL Training	+ T-STAR	46.0	59.0	46.0	**50.3**	**42.5**	**37.5**	**33.0**	30.0	**42.0**
RL Training	DAPO	45.5	59.5	46.0	50.3	40.5	36.5	32.0	29.5	41.2
RL Training	+ T-STAR	**48.0**	61.0	48.0	**52.3**	**44.5**	**40.0**	**35.0**	**32.5**	**44.0**
RL Training	GiGPO	47.5	61.5	47.5	52.2	43.5	39.0	34.5	32.0	43.6
RL Training	+ T-STAR	49.5	**63.5**	**49.5**	**54.2**	**48.0**	**43.5**	**37.5**	**35.0**	**46.6**

Table 3: Performance on logical planning tasks. T-STAR enables recovery from dead-ends through thought grafting.

Type	Method	Sokoban (Success %)			Blocksworld	
		Easy	Med.	Hard	Stack	Unstack
Closed-Source Model						
Prompting	GPT-4o	78.3	45.4	17.4	84.6	89.8
Prompting	Gemini-1.5-Pro	72.7	40.4	14.4	79.7	85.4
Qwen2.5-3B-Instruct						
RL Training	PPO	16.9	4.7	0.7	24.1	32.4
RL Training	GRPO	37.4	13.6	2.2	47.1	54.4
RL Training	+ T-STAR	**44.7**	**20.6**	**6.7**	**55.7**	**62.9**
RL Training	DAPO	42.6	18.4	3.9	52.3	58.9
RL Training	+ T-STAR	**49.4**	**25.9**	**8.3**	**61.4**	**67.7**
RL Training	GiGPO	47.2	22.8	6.2	58.3	64.8
RL Training	+ T-STAR	**54.4**	**29.6**	**11.2**	**67.1**	**73.4**
Phi-4-mini-instruct-3.8B						
RL Training	PPO	18.7	5.2	0.4	25.9	33.9
RL Training	GRPO	39.4	14.8	3.3	48.8	56.1
RL Training	+ T-STAR	**46.9**	**22.4**	**6.7**	**58.4**	**64.7**
RL Training	DAPO	43.6	19.1	4.2	53.9	59.8
RL Training	+ T-STAR	**51.8**	**27.4**	**9.4**	**63.4**	**70.3**
RL Training	GiGPO	48.6	23.8	7.3	59.6	66.8
RL Training	+ T-STAR	**56.1**	**32.4**	**12.6**	**69.8**	**76.4**

text-based household tasks (pick, clean, heat, cool) while WebShop requires e-commerce navigation with product search and attribute verification. Search-augmented QA includes single-hop datasets (Natural Questions, TriviaQA, PopQA) and multi-hop benchmarks (HotpotQA, 2WikiMultiHopQA, MusiQue, Bamboogle) requiring compositional reasoning across documents. Logical planning tasks include Sokoban with varying difficulty levels and Blocksworld with stacking operations.

Baselines include three group-based RL methods: GRPO with in-group trajectory comparison, DAPO with dynamic advantage scaling, and GiGPO with group-in-group optimization structures. Prompting methods (ReAct, Reflexion) and closed-source models (GPT-4o, Gemini-1.5-Pro) serve as additional references. T-STAR is applied on top of each RL baseline using Qwen2.5-3B-Instruct and Phi-4-mini-instruct-3.8B, with all methods adopting the ReAct framework and search tool access. Training runs for 160 steps with EMA-updated reference policy ($\alpha = 0.95$). Evaluation uses success rate for ALFWorld and Sokoban, score and success rate for WebShop, and Exact Match for QA tasks, with results averaged over three seeds.

4.2 Main Results

T-STAR demonstrates consistent improvements across all three task categories and baseline methods, as shown in Tables 1–3. On interactive and

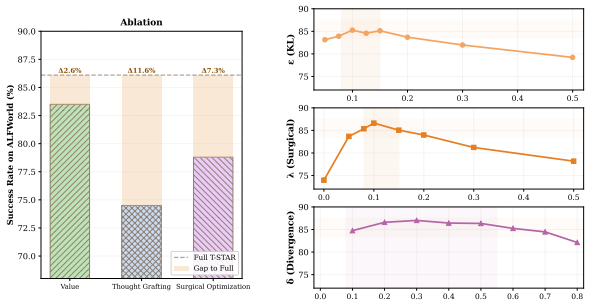

Figure 6: Ablation study on ALFWorld. Left: component contribution. Right: hyperparameter sensitivity (ϵ_{kl}, λ, δ).

embodied tasks, T-STAR achieves 3.0–3.8% gains on ALFWorld and 3.2–5.8% on WebShop. The improvement is particularly pronounced on complex subtasks requiring longer action sequences, such as Pick2 and Clean, where trajectory overlap occurs more frequently and enables greater variance reduction through shared node averaging. On search-augmented QA, T-STAR shows its largest gains on multi-hop reasoning: 2.8–7.5% on HotpotQA, 2.8–6.9% on MultiHopQA, 2.2–4.6% on MusiQue, and 2.8–6.8% on Bamboogle, while single-hop improvements are comparatively modest at 1.9–3.5%. This disparity aligns with our theoretical analysis—multi-hop tasks involve more sequential decisions, creating more opportunities for trajectory sharing at early reasoning steps. On logical planning tasks, T-STAR achieves 5.5–7.5% gains on easy levels, 4.5–8.5% on medium difficulty, and 3.0–4.0% even on hard instances where baseline methods struggle to learn meaningful policies. The training dynamics in Figure 5 also reveal that T-STAR not only achieves higher final performance but also exhibits substantially more stable learning. The stability is evident in multi-hop QA, where sparse rewards cause baseline methods to exhibit high variance and occasional performance drops during training, while our method maintains stable improvement throughout training process.

4.3 Analysis

As illustrated in Figure 6, the ablation experiments on ALFWorld reveal the relative contribution of each component to the overall improvement. Removing thought grafting causes the largest performance drop at 11.6%, confirming that corrective supervision at divergence points is the most critical mechanism. Removing Q-tree valuation reduces performance by 7.3%, demonstrating that variance-reduced credit assignment provides substantial benefit even without explicit rectification. Removing surgical optimization decreases performance by 2.6%, indicating that while step-level preference learning contributes, the primary gains derive from improved credit assignment and corrective data synthesis rather than the specific optimization procedure. The hyperparameter sensitivity analysis in Figure 6 shows robustness across reasonable ranges of ϵ_{kl}, λ, and δ, with graceful degradation outside optimal settings.

The relationship between task complexity and improvement magnitude provides additional validation for our approach. Tasks with longer reasoning chains exceeding 15 steps show 5–8% improvements, while shorter tasks under 10 steps yield 2–4% gains, confirming that tree-based credit assignment provides greatest value when sparse rewards must propagate through many decision points. Manual inspection of 100 grafted thoughts reveals that 83% provide semantically meaningful corrections that address specific reasoning errors, including adding missing search constraints, correcting logical inference steps, and refining action selection based on observed outcomes.

5 Conclusion

In this work, we presented T-STAR, a framework that addresses sparse supervision in multi-step agent reinforcement learning through tree-structured credit assignment and self-taught rectification. By consolidating independent rollouts into a Cognitive Tree, T-STAR enables variance-reduced advantage estimation while localizing critical divergence points where the agent synthesizes step-level corrective supervision through thought grafting and surgical policy optimization, without requiring additional reward models or rollouts.

Experiments across embodied, interactive, reasoning, and planning tasks demonstrate consistent improvements over a wide range of baselines, with gains pronounced on tasks requiring extended reasoning chains. The decreasing value spread at divergence points and increasing anchor reuse confirm that the framework produces genuine policy improvement through generalizable corrections. The core insight that independent rollouts contain latent shared structure exploitable for both variance reduction and targeted correction, suggests broader applications to sequential decision-making settings.

Limitations

The functional equivalence criterion based on KL divergence relies on Monte Carlo approximation, which may introduce noise for states with high-entropy action distributions but can also be addressed by adaptive sampling strategies in future work. Additionally, thought grafting requires the agent to generate meaningful corrections conditioned on contrasting branches, which depends on the base model's ability to identify and articulate reasoning differences. Finally, our evaluation focuses on text-based environments with discrete actions; extending T-STAR to continuous action spaces or multimodal observations would be interesting topics for future exploration.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight experience replay. *Advances in neural information processing systems*, 30.

Hao Bai, Yifei Zhou, Jiayi Pan, Mert Cemri, Alane Suhr, Sergey Levine, and Aviral Kumar. 2024. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. *Advances in Neural Information Processing Systems*, 37:12461–12495.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Wentse Chen, Jiayu Chen, Hao Zhu, and Jeff Schneider. 2025. Context-lite multi-turn reinforcement learning for llm agents. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*.

Alberto Silvio Chiappa, Alessandro Marin Vargas, Ann Huang, and Alexander Mathis. 2023. Latent exploration for reinforcement learning. *Advances in Neural Information Processing Systems*, 36:56508–56530.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2025. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, and 1 others. 2023. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025a. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Yiran Guo, Lijie Xu, Jie Liu, Dan Ye, and Shuang Qiu. 2025b. Segment policy optimization: Effective segment-level credit assignment in rl for large language models. *arXiv preprint arXiv:2505.23564*.

Bingning Huang, Tu Nguyen, and Matthieu Zimmer. 2025. Tree-opo: Off-policy monte carlo tree-guided advantage optimization for multistep reasoning. *arXiv preprint arXiv:2509.09284*.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2023. Large language models can self-improve. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 1051–1068.

Komal Kumar, Tajamul Ashraf, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, Phillip HS Torr, Fahad Shahbaz Khan, and Salman Khan. 2025. Llm post-training: A deep dive into reasoning large language models. *arXiv preprint arXiv:2502.21321*.

Navdeep Kumar, Esther Derman, Matthieu Geist, Kfir Y Levy, and Shie Mannor. 2023. Policy gradient for rectangular robust markov decision processes. *Advances in Neural Information Processing Systems*, 36:59477–59501.

Yu Li, Tian Lan, and Zhengling Qi. 2025. Inspo: Unlocking intrinsic self-reflection for llm preference optimization. *arXiv preprint arXiv:2512.23126*.

Yu Li, Tian Lan, and Zhengling Qi. 2026a. When right meets wrong: Bilateral context conditioning with reward-confidence correction for grpo. *arXiv preprint arXiv:2603.13134*.

Yu Li, Rui Miao, Zhengling Qi, and Tian Lan. 2026b. Arise: Agent reasoning with intrinsic skill evolution in hierarchical reinforcement learning. *arXiv preprint arXiv:2603.16060*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Chanwoo Park, Seungju Han, Xingzhi Guo, Asuman E Ozdaglar, Kaiqing Zhang, and Joo-Kyung Kim. 2025.

Maporl: Multi-agent post-co-training for collaborative large language models with reinforcement learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30215–30248.

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Zexu Sun, Bowei He, Jinxin Liu, Xu Chen, Chen Ma, and Shuai Zhang. 2023. Offline imitation learning with variational counterfactual reasoning. *Advances in Neural Information Processing Systems*, 36:43729–43741.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

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

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, and 1 others. 2025a. Reinforcement learning for reasoning in large language models with one training example. *arXiv preprint arXiv:2504.20571*.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, and 1 others. 2025b. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*.

Zhicheng Yang, Zhijiang Guo, Yinya Huang, Xiaodan Liang, Yiwei Wang, and Jing Tang. 2025. Treerpo: Tree relative policy optimization. *arXiv preprint arXiv:2506.05183*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488.

Siliang Zeng, Quan Wei, William Brown, Oana Frunza, Yuriy Nevmyvaka, Yang Katie Zhao, and Mingyi Hong. 2025. Reinforcing multi-turn reasoning in llm agents via turn-level credit assignment. In *ICML 2025 Workshop on Computer Use Agents*.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, and 1 others. 2025a. The landscape of agentic reinforcement learning for llms: A survey. *arXiv preprint arXiv:2509.02547*.

Kongcheng Zhang, Qi Yao, Baisheng Lai, Jiaxing Huang, Wenkai Fang, Dacheng Tao, Mingli Song, and Shunyu Liu. 2025b. Reasoning with reinforced functional token tuning. *arXiv preprint arXiv:2502.13389*.

Zeyu Zheng, Junhyuk Oh, and Satinder Singh. 2018. On learning intrinsic rewards for policy gradient methods. *Advances in neural information processing systems*, 31.

Appendix

A Further Analysis

A.1 Theoretical Analysis

Lemma 2 (Aggregation and Variance Reduction)**.** *The tree-based node advantage $\hat{A}_{tree}(v)$ satisfies:*

$$\hat{A}_{tree}(v) = \frac{1}{k_v} \sum_{i \in \mathcal{T}(v)} \hat{A}_{GRPO}(\tau_i), \tag{13}$$

and

$$Var(\hat{A}_{tree}(v)) = \frac{1}{k_v} Var(\hat{A}_{GRPO}(\tau_i)). \tag{14}$$

Proof. **1. Derivation of Unbiased Aggregation**

Let $R_i \triangleq R(\tau_i)$ denote the terminal reward for trajectory i. The GRPO advantage is given by $\hat{A}_{\text{GRPO}}(\tau_i) = \sigma_R^{-1}(R_i - \bar{R})$. Substituting the definition of $Q_{\text{tree}}(v)$ into the node advantage formulation:

$$\begin{aligned}
\hat{A}_{\text{tree}}(v) &= \frac{1}{\sigma_R} \left(\left[\frac{1}{k_v} \sum_{i \in \mathcal{T}(v)} R_i \right] - \bar{R} \right) \\
&= \frac{1}{\sigma_R} \left(\frac{1}{k_v} \sum_{i \in \mathcal{T}(v)} R_i - \frac{1}{k_v} \sum_{i \in \mathcal{T}(v)} \bar{R} \right) \\
&= \frac{1}{k_v} \sum_{i \in \mathcal{T}(v)} \frac{R_i - \bar{R}}{\sigma_R} \\
&= \frac{1}{k_v} \sum_{i \in \mathcal{T}(v)} \hat{A}_{\text{GRPO}}(\tau_i).
\end{aligned} \tag{15}$$

This establishes the linear equivalence between the node-level advantage and the mean trajectory-level advantage.

2. Variance Analysis via Conditional Independence

Let X_i be the random variable representing the advantage $\hat{A}_{\text{GRPO}}(\tau_i)$ for a trajectory passing through v. We analyze the variance of the estimator $\hat{A}_{\text{tree}}(v)$ conditioned on the prefix node v. Since the policy π_θ generates rollouts stochastically, for any distinct pair $i, j \in \mathcal{T}(v)$ with $i \neq j$, the completions are independent given v. Thus, the covariance term vanishes:

$$\text{Cov}(X_i, X_j \mid v) = 0, \quad \forall i \neq j. \tag{16}$$

Let $\Sigma^2 \triangleq \text{Var}(X_i)$ be the variance of the single-trajectory advantage. Expanding the variance of the sum:

$$\begin{aligned}
\text{Var}(\hat{A}_{\text{tree}}(v)) &= \text{Var}\left(\frac{1}{k_v} \sum_{i \in \mathcal{T}(v)} X_i \right) \\
&= \frac{1}{k_v^2} \left(\sum_{i \in \mathcal{T}(v)} \text{Var}(X_i) + \sum_{i \neq j} \text{Cov}(X_i, X_j) \right) \\
&= \frac{1}{k_v^2} \left(\sum_{i=1}^{k_v} \Sigma^2 + 0 \right) \\
&= \frac{k_v \Sigma^2}{k_v^2} = \frac{1}{k_v} \text{Var}(\hat{A}_{\text{GRPO}}(\tau_i)).
\end{aligned} \tag{17}$$

Consequently, for any shared node where $k_v > 1$, the variance of the gradient estimation signal is strictly reduced.

A.2 Complexity Analysis

We analyze the computational overhead of T-STAR relative to standard GRPO, decomposing into three components and providing both theoretical bounds and empirical measurements.

A.2.1 Tree Construction

The tree construction phase involves two sub-operations: functional equivalence testing and graph-based merging.

Functional Equivalence Testing. For M trajectories of average length T, we must evaluate pairwise KL divergence at each depth $d \in \{0, 1, \ldots, T-1\}$. At depth d, let n_d denote the number of distinct nodes before merging. Computing exact KL divergence $D_{\text{KL}}(v_i \| v_j) = \sum_a \pi_\theta(a|s_i) \log \frac{\pi_\theta(a|s_i)}{\pi_\theta(a|s_j)}$ requires enumerating the action space $|A|$, which is intractable for large vocabularies.

We adopt Monte Carlo approximation: sample K actions from $\pi_\theta(\cdot|s_i)$ and estimate

$$\hat{D}_{\text{KL}}(v_i \| v_j) = \frac{1}{K} \sum_{k=1}^{K} \log \frac{\pi_\theta(a_k|s_i)}{\pi_\theta(a_k|s_j)}, \quad a_k \sim \pi_\theta(\cdot|s_i). \tag{18}$$

This requires K forward passes per node pair. With $n_d \leq M$ nodes at each depth, the total cost across all depths is $O(M^2 T K)$ forward passes. We set $K = 16$, which provides sufficient accuracy for threshold-based equivalence decisions.

Graph-Based Merging. At each depth, we construct a compatibility graph $G_d = (V_d, E_d)$ where

edges connect functionally equivalent and historically compatible nodes. Finding connected components via union-find requires $O(n_d^2 \cdot \alpha(n_d))$ operations, where α is the inverse Ackermann function. Summing across depths yields $O(M^2T \cdot \alpha(M))$, which is effectively linear in practice.

Historical Compatibility. Checking $S(v_i) = S(v_j)$ requires comparing action histories of length at most d. With efficient hashing of state-modifying action sequences, this adds $O(M^2T^2)$ hash comparisons in the worst case, though average-case complexity is lower due to early rejection.

A.2.2 Q-tree Computation

The Q-tree value computation (Eq. 5) performs a single bottom-up traversal of the constructed tree. Let $|V|$ denote the total number of nodes after merging.

Bellman Backup. Each internal node v computes $Q_{\text{tree}}(v) = \gamma \sum_{v' \in C(v)} w(v \to v') Q_{\text{tree}}(v')$, requiring $O(|C(v)|)$ operations. Summing over all nodes:

$$\sum_{v \in V} |C(v)| = |E| \leq |V| - 1 = O(MT). \quad (19)$$

Thus, the total Bellman backup cost is $O(MT)$ arithmetic operations.

Edge Weight Computation. Edge weights $w(v \to v') = |T(v \to v')|/|T(v)|$ are computed during tree construction by maintaining trajectory membership sets. Using efficient set operations, this adds $O(MT)$ overhead.

Divergence Identification. Computing $\Delta V(v) = \max_{v'} Q_{\text{tree}}(v') - \min_{v''} Q_{\text{tree}}(v'')$ for each internal node requires $O(|C(v)|)$ comparisons per node, totaling $O(MT)$ across the tree.

A.2.3 Thought Grafting

For each divergent node $v \in V_{\text{div}}$, generating a rectified thought requires one forward pass through the policy model.

Number of Divergent Nodes. The divergence threshold δ controls the granularity of rectification. Let p_{div} denote the fraction of internal nodes satisfying $\Delta V(v) > \delta$. With $\delta = 0.3$, we observe $p_{\text{div}} = 0.12$ on ALFWorld and $p_{\text{div}} = 0.09$ on WebShop. With approximately $MT/2$ internal nodes, we have $|V_{\text{div}}| \approx p_{\text{div}} \cdot MT/2$.

Generation Cost. Each rectified thought generation involves conditioning on the shared context s, the successful branch v^+, and the failed branch v^-. The input length is approximately $3L$ where L is the average context length. Generation produces thoughts of average length l_z, requiring $O(l_z)$ autoregressive steps.

A.2.4 Overall Complexity Comparison

Table 4 summarizes the computational complexity of each component.

Component	Time Complexity	Dominant Cost
Standard GRPO	$O(MT \cdot L)$	Forward/backward passes
T-STAR *Additional Overhead*		
KL Estimation	$O(M^2TK)$	Forward passes (inference)
Graph Merging	$O(M^2T)$	CPU operations
Q-tree Backup	$O(MT)$	Arithmetic operations
Thought Grafting	$O(p_{\text{div}} \cdot MT \cdot l_z)$	Forward passes (generation)

Table 4: Computational complexity breakdown. M: trajectories per task, T: average trajectory length, K: MC samples, L: context length, l_z: thought length, p_{div}: divergence fraction.

The dominant additional cost is KL estimation during tree construction. However, this involves inference-only forward passes without gradient computation, which are significantly cheaper than training forward-backward passes.

A.2.5 Empirical Runtime Analysis

We measure actual training time on ALFWorld with Qwen2.5-3B-Instruct.

Method	Time/Iter (s)	Total Time (h)	Relative
GRPO	42.3	1.88	1.00×
GRPO + T-STAR	51.8	2.30	1.22×
T-STAR *Breakdown*			
Tree Construction	5.2	–	–
Q-tree + Divergence	0.8	–	–
Thought Grafting	3.5	–	–

Table 5: Runtime comparison on ALFWorld (160 training iterations).

A.2.6 Scalability Considerations

Scaling with Trajectory Count M. The $O(M^2)$ term in tree construction becomes significant for large M. For $M > 16$, we recommend hierarchical clustering: first group trajectories by coarse features (e.g., first action), then apply full equivalence testing within groups. This reduces the complexity to $O(M^2/g + g \cdot (M/g)^2) = O(M^2/g)$ where g is the number of groups.

Scaling with Trajectory Length T. Longer trajectories increase tree depth but typically exhibit

Model Size	GRPO Time (s)	T-STAR Overhead (s)	Relative Overhead
1.5B	28.4	8.2	28.9%
3B	42.3	9.5	22.5%
7B	78.6	11.3	14.4%

Table 6: Scaling behavior of T-STAR overhead with model size on ALFWorld.

Category	Dataset	Train	Test	Steps
Embodied	ALFWorld	3,321	140	12.4
	WebShop	10,587	500	8.6
Single-hop	NQ	79,168	3,610	3.2
	TriviaQA	78,785	11,313	2.8
	PopQA	–	14,267	2.5
Multi-hop	HotpotQA	90,447	7,405	6.4
	2WikiMH	15,000	12,576	5.8
	MusiQue	19,938	2,417	7.2
	Bamboogle	–	125	4.6
Planning	Sokoban	5,000	1,000	18.5
	Blocksworld	4,000	800	14.2

Table 7: Dataset statistics across four task categories.

Param	Val	Param	Val
KL threshold ϵ_{kl}	0.25	Learning rate	5e-6
MC samples K	16	Batch size	32
Trajectories M	8	Training steps	160
Discount γ	0.99	EMA coef. α	0.95
Divergence δ	0.3	Max seq. len	4096
Surgical wt. λ	0.15	Max steps	20
Temperature β	0.1		

Table 8: Hyperparameter settings for T-STAR.

more sharing at early depths, maintaining reasonable $|V|$. The linear dependence on T in most components ensures scalability. Empirically, we observe that the effective tree size grows sublinearly with T due to increased merging opportunities at shallow depths.

Scaling with Model Size. Forward pass cost scales with model parameters, affecting both KL estimation and thought grafting. For larger models, the relative overhead of T-STAR decreases since the fixed-cost components (graph operations, Q-tree computation) remain constant. Table 6 presents scaling behavior across different model sizes.

B Experimental Details

B.1 Dataset Statistics

We evaluate T-STAR across 11 datasets spanning four task categories. Table 7 summarizes the statistics of all benchmarks. The Avg. Steps column indicates the average number of reasoning steps required per task, which directly correlates with the potential for trajectory sharing in our cognitive tree construction.

B.2 Hyperparameter Settings

Table 8 provides complete hyperparameter settings for T-STAR. The KL threshold $\epsilon_{\text{kl}} = 0.25$ balances merging granularity: smaller values create sparser trees with less sharing, while larger values risk merging semantically different states. The divergence threshold $\delta = 0.3$ controls the selectivity of thought grafting.

B.3 Baseline Implementations

For baseline implementations, we ensure fair comparison by maintaining consistent configurations across all methods.

GRPO. We implement GRPO with trajectory-level advantage normalization using group size $M = 8$. The clipping parameter is set to $\epsilon = 0.2$.

DAPO. DAPO applies dynamic advantage scaling with clip range $[0.8, 1.2]$ and temperature annealing from 1.0 to 0.5 over training.

GiGPO. GiGPO uses outer group size 4 and inner group size 2, maintaining 8 trajectories per task.

Prompting Baselines. ReAct and Reflexion use 3-shot demonstrations. Reflexion maintains a verbal feedback buffer with maximum 3 reflections. All prompting methods use greedy decoding.

B.4 Evaluation Metrics

We adopt task-specific evaluation metrics:

- **ALFWorld**: Success rate (%) within 30 steps.
- **WebShop**: Score (0-100) based on attribute matching, and success rate (%).
- **QA Tasks**: Exact Match (EM) after normalization.
- **Sokoban**: Success rate (%) within step limits (easy: 20, medium: 40, hard: 60).
- **Blocksworld**: Success rate (%) for target configuration.

Method	Pk	Lk	Cl	Ht	Co	Pk2	All
GRPO	79.5	67.0	73.5	83.0	87.0	71.0	76.8
+T-STAR	82.0	70.5	76.0	85.5	89.0	75.5	79.8
DAPO	83.0	71.0	77.0	86.0	90.0	75.0	80.3
+T-STAR	85.5	74.0	80.5	88.5	92.0	79.0	83.3
GiGPO	89.0	77.0	84.0	92.0	95.0	82.0	86.5
+T-STAR	91.5	80.5	87.0	94.5	97.0	85.5	89.3

Table 9: Per-task results on ALFWorld (Qwen2.5-3B). Pk=Pick, Lk=Look, Cl=Clean, Ht=Heat, Co=Cool.

Dataset	Depth	Nodes	Merge	$\lvert V_{\text{div}} \rvert$
ALFWorld	12.4	47.2	0.52	5.8
WebShop	8.6	38.5	0.44	4.2
HotpotQA	6.4	28.3	0.45	3.6
Sokoban-E	12.8	52.1	0.58	6.4
Sokoban-H	24.6	89.4	0.41	8.2

Table 10: Cognitive tree statistics. Merge ratio = 1 - (nodes after / nodes before merging).

B.5 Infrastructure

All experiments are conducted on NVIDIA A100 80GB GPUs with DeepSpeed ZeRO Stage 2. We use vLLM for efficient inference during trajectory sampling.

C Additional Results

C.1 Per-Task Breakdown on ALFWorld

Table 9 shows per-task performance on ALFWorld. The largest gains are on Pick2 (+4.5%) and Look (+3.5%), involving longer action sequences.

C.2 Cognitive Tree Statistics

Table 10 presents cognitive tree statistics. The merge ratio reflects trajectory sharing degree. Higher merge ratios correlate with larger improvements.

C.3 Grafting Quality Analysis

We manually analyze 100 grafting instances per benchmark. Table 11 summarizes results.

D Case Study

We provide examples illustrating how T-STAR identifies divergence points and performs thought grafting.

D.1 WebShop Example

Task: *I'm looking for a small portable folding desk that is already fully assembled; khaki wood finish, price < $140.*

Dataset	Valid	Error	Success
ALFWorld	91%	83%	72%
WebShop	88%	79%	68%
HotpotQA	85%	76%	65%
Sokoban	82%	71%	58%

Table 11: Grafting quality. Valid: semantically correct; Error: identifies mistake; Success: leads to completion.

Shared Prefix (Merged Nodes):

1. `search[portable folding desk khaki]`
2. `click[B09Q3B86B]` → Product page
3. `click[item-description]`

Divergence Point ($\Delta V = 0.72$):

Successful Branch (v^+)**:**
Thought: Description says "assembly required", doesn't match "fully assembled". Should check other products.
Action: `click[back to search]`
Outcome: Finds pre-assembled desk. **R=1.0**

Failed Branch (v^-)**:**
Thought: Desk looks good, right color, $89.99 under budget.
Action: `click[khaki wood][buy now]`
Outcome: Buys unassembled desk. **R=0.0**

Grafted Thought (z_{rect})**:**
Description mentions "assembly required". User requested "fully assembled". This doesn't meet requirements. Should return to search for pre-assembled option.

D.2 HotpotQA Example

Question: *Were the directors of "Jaws" and "E.T." born in the same decade?*

Divergence Point ($\Delta V = 0.65$) after first search:

Successful (v^+)**:**
Thought: Found Spielberg directed Jaws. Need to verify E.T. director and birth year.
Action: `search[E.T. director]`
Outcome: Confirms director, finds 1946. **R=1.0**

Failed (v^-)**:**
Thought: Spielberg born 1946. Same era movies, probably same decade.
Action: `finish[Yes]`
Outcome: Premature answer. **R=0.0**

Grafted (z_{rect})**:**
Confirmed Spielberg directed Jaws (born 1946). Question asks about both films. Should not assume same director without verification. Search E.T. director first.