arXiv:2610.14737v1 [cs.LG] 10 Oct 2026

PTGPO: PREFIX TREE GROUP POLICY OPTIMIZATION

Ruolin Chen[1,2,3⋆], *Yang Li*[3⋆], *Jihang Wang*[1,2⋆], *Zhuoya Zhao*[4⋆], *Yuchen Liu*[3], *Jiaming Xu*[3†], *Qian Zhang*[1†]

[1] Brain-inspired Cognitive AI Lab, Institute of Automation, Chinese Academy of Sciences
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences
[3] HyperAI Team, Xiaomi Corporation
[4] University of Ottawa

ABSTRACT

Training language agents on long-horizon interactive tasks is challenging because sparse terminal rewards provide weak credit signals for intermediate actions. Existing group-based methods either assign a shared trajectory-level advantage or estimate step-level advantages from state groups, which may conflate the current action with subsequent decisions. We propose Prefix Tree Group Policy Optimization (PTGPO), a critic-free method that constructs prefix trees over trajectory suffixes within anchor-state groups. PTGPO assigns each tree edge a credit given by the difference between its child and parent empirical mean returns, capturing the conditional return change revealed by each additional action. These prefix-tree credits provide fine-grained learning signals without requiring a critic or additional rollouts. Experiments on interactive benchmarks ALFWorld, WebShop and Sokoban show that PTGPO consistently improves over RL baselines, demonstrating its effectiveness.

***Index Terms*—** LLM agent, reinforcement learning, credit assignment, policy optimization

1. INTRODUCTION

Recent advances [1, 2] in large language models (LLMs) have extended their capabilities beyond static text generation toward interactive reasoning and decision-making. LLM-powered agents can perceive, reason, and act across diverse environments, including embodied tasks [3, 4], web navigation [5, 6], and interactive games [7, 8]. Unlike single-turn generation, these applications require long-horizon planning and coherent sequences of actions, where intermediate decisions shape subsequent observations and task completion. Reinforcement learning (RL) provides a natural post-training framework for improving such agents through interaction and task-level feedback [9, 10, 11]. A central challenge is translating this feedback into effective learning signals for the individual decisions made throughout an episode.

Group-based reinforcement learning methods such as GRPO [12] have become an effective critic-free paradigm for LLM training, where multiple responses can be sampled for the same prompt and compared using verifiable outcome rewards. These methods have demonstrated strong performance on single-turn tasks, particularly mathematical reasoning. However, extending these methods to long-horizon multi-turn agentic tasks remains challenging [11]. In such environments, an episode may involve many interaction steps, while rewards are often sparse, so assigning a single trajectory-level advantage to every action obscures which decisions actually caused success or failure [13, 14]. Recent methods such as GiGPO [15] mitigate this issue by grouping steps with identical environment states. However, computing step-level advantages from trajectory outcomes within such groups may conflate the grouped action with later decisions, creating a credit-assignment mismatch.

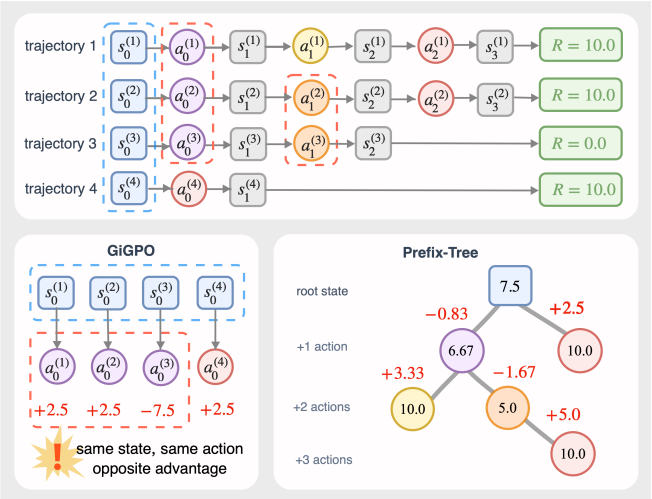

Fig. 1. Comparison of state-group and prefix-tree credit assignment under the undiscounted setting ($\gamma = 1$). Identical actions may receive conflicting advantages when their subsequent actions lead to different returns. PTGPO recursively groups trajectory suffixes by action prefix, where node values denote empirical mean returns and edge labels denote child-parent mean differences. Colors indicate identical actions.

As illustrated in Fig. 1, even when two samples share the same current state and take the same action, their terminal rewards may differ, or even have opposite signs, because their subsequent actions diverge. This difference should not be attributed to the current action, but rather to the subsequent action that actually determines the outcome. However, these decisive actions often occur after the trajectories have entered sparse or non-identical states, where reliable state-level groups can no longer be formed. The unresolved return variation should therefore be carried forward along the shared action prefix and progressively decomposed as subsequent actions are revealed. When the trajectories branch into groups with different conditional mean returns, credit can be assigned to the newly revealed branching action.

In this work, we propose Prefix Tree Group Policy Optimization

⋆Equal contribution. †Corresponding author (xujiaming1@xiaomi.com, q.zhang@ia.ac.cn).

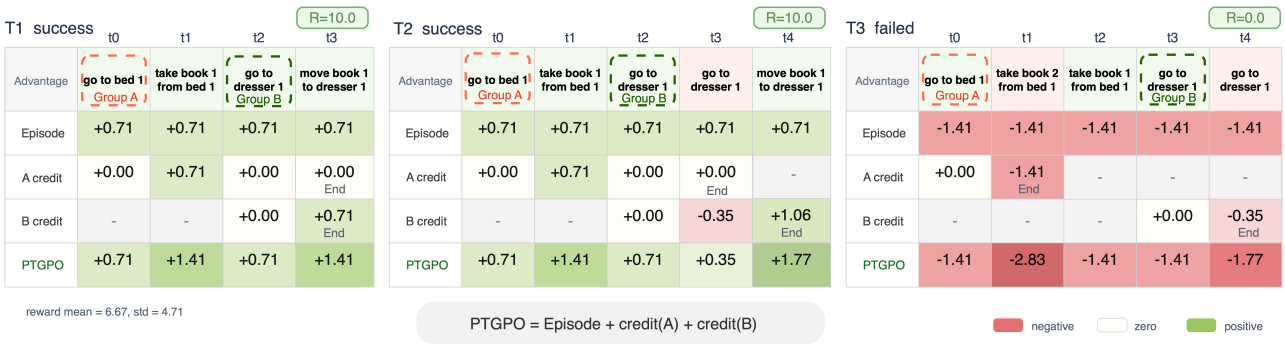

Fig. 2. Illustration of PTGPO advantage decomposition under the undiscounted setting ($\gamma = 1$). Episode-level advantages are shared by all actions in the same trajectory, while prefix-tree credits redistribute return residuals to branching decisions that change the conditional mean return. "End" marks the termination of a prefix decomposition.

(PTGPO), a critic-free method for fine-grained credit assignment in long-horizon tasks. PTGPO constructs a prefix tree for each anchor-state group, recursively grouping trajectory suffixes by shared action prefixes. For each parent-child edge in the tree, PTGPO assigns credit equal to the difference between the child and parent empirical mean returns. This quantity measures the conditional return increment introduced by the newly revealed action. Across three challenging agentic benchmarks ALFWorld, WebShop and Sokoban, PTGPO consistently outperforms existing prompting and RL baselines. Our analysis further shows that the proposed prefix-tree credit assignment provides effective step-level learning signals, making PTGPO a practical solution for long-horizon agentic tasks with sparse rewards.

2. METHOD

For each sampled trajectory i, let its length be T_i. For each sampled step (i, t), we define the discounted return as

$$\hat{R}_t^{(i)} = \sum_{u=t}^{T_i} \gamma^{u-t} r_u^{(i)}, \tag{1}$$

where $\gamma \in [0, 1]$ is the discount factor. In sparse terminal-reward tasks, intermediate rewards are zero.

Following GiGPO, we use the same anchor-state matching strategy to construct step-level root groups over sampled rollouts. Let $\tilde{s}$ denote an anchor state, and let

$$\mathcal{G}(\tilde{s}) = \{(i, t) \mid s_t^{(i)} = \tilde{s}\} \tag{2}$$

be the set of all step occurrences that share this anchor state.

For each group, PTGPO builds a prefix tree. Let $\mathbf{a}_d$ be a sequence of d actions. The corresponding node is the set

$$p(\tilde{s}, \mathbf{a}_d) = \left\{(i, t) \in \mathcal{G}(\tilde{s}) \;\middle|\; (a_t^{(i)}, \ldots, a_{t+d-1}^{(i)}) = \mathbf{a}_d\right\}. \tag{3}$$

Samples with the same action sequence of length d are included. The root is $p(\tilde{s}, \varnothing) = \mathcal{G}(\tilde{s})$. If $\mathbf{a}_{d+1}$ extends $\mathbf{a}_d$ by one action a_d, then the node $p(\tilde{s}, \mathbf{a}_{d+1})$ is a child of $p(\tilde{s}, \mathbf{a}_d)$, and we have

$$p(\tilde{s}, \mathbf{a}_{d+1}) \subseteq p(\tilde{s}, \mathbf{a}_d). \tag{4}$$

Thus, starting from the root, PTGPO recursively splits each node according to the next action, merging sampled steps with the same future action prefix into the same branch.

For any nonempty node, its empirical mean return is

$$\hat{\mu}\big(p(\tilde{s}, \mathbf{a}_d)\big) = \frac{1}{|p(\tilde{s}, \mathbf{a}_d)|} \sum_{(i,t) \in p(\tilde{s}, \mathbf{a}_d)} \hat{R}_t^{(i)}. \tag{5}$$

For a sampled step (i, t) with $s_t^{(i)} = \tilde{s}$, consider its prefix-tree path, where $\mathbf{a}_d = (a_t^{(i)}, \ldots, a_{t+d-1}^{(i)})$ and $\mathbf{a}_0 = \varnothing$. The unnormalized state-group step advantage [15] then admits the following decomposition along a path expanded to depth D:

$$\begin{aligned} \hat{R}_t^{(i)} - \hat{\mu}\big(p(\tilde{s}, \varnothing)\big) = &\sum_{d=0}^{D-1} \Delta\big(p(\tilde{s}, \mathbf{a}_d), p(\tilde{s}, \mathbf{a}_{d+1})\big) \\ &+ \left[\hat{R}_t^{(i)} - \hat{\mu}\big(p(\tilde{s}, \mathbf{a}_D)\big)\right], \end{aligned} \tag{6}$$

where

$$\Delta\big(p(\tilde{s}, \mathbf{a}_d), p(\tilde{s}, \mathbf{a}_{d+1})\big) = \hat{\mu}\big(p(\tilde{s}, \mathbf{a}_{d+1})\big) - \hat{\mu}\big(p(\tilde{s}, \mathbf{a}_d)\big) \tag{7}$$

is the credit assigned to the newly added action $a_{t+d}^{(i)}$ for each $(i, t) \in p(\tilde{s}, \mathbf{a}_{d+1})$, reflecting the change in mean return from parent to child, while differences due to later actions remain within $p(\tilde{s}, \mathbf{a}_{d+1})$ for further decomposition. A node can be expanded only if $|p(\tilde{s}, \mathbf{a}_d)| \geq \tau_{\text{expand}}$.

PTGPO normalizes each credit by the standard deviation of returns in its root group:

$$\hat{\sigma}\big(p(\tilde{s}, \varnothing)\big) = \mathrm{Std}\left(\left\{\hat{R}_t^{(i)} \mid (i, t) \in p(\tilde{s}, \varnothing)\right\}\right). \tag{8}$$

For action $a_k^{(i)}$, PTGPO sums the normalized edge credits from starting steps $t = 0, \ldots, k$:

$$A_k^{\mathrm{PT},(i)} = \sum_{t=0}^{k} \frac{\Delta\Big(p\big(s_t^{(i)}, \mathbf{a}_{k-t}\big), p\big(s_t^{(i)}, \mathbf{a}_{k-t+1}\big)\Big)}{\hat{\sigma}\big(p(s_t^{(i)}, \varnothing)\big)}. \tag{9}$$

Each edge credit is normalized by the standard deviation of the root group associated with its starting step t, with each term capturing a distinct conditional-return innovation under a different anchor-conditioned prefix decomposition. In practice, the accumulated magnitude remains well controlled because the maximum expansion depth of each prefix tree is limited and valid root groups become increasingly sparse along the trajectory.

Table 1. Performance on ALFWorld and WebShop. We train each RL method with 3 random seeds and report the mean and standard deviation. The best performances are highlighted in bold.

Type	Method	ALFWorld							WebShop	
		Pick	Look	Clean	Heat	Cool	Pick2	All	Score	Succ.
Closed-Source Model										
Prompting	GPT-4o	75.3	60.8	31.2	56.7	21.6	49.8	48.0	31.8	23.7
Prompting	Gemini-2.5-Pro	92.8	63.3	62.1	69.0	26.6	58.7	60.3	42.5	35.9
Qwen2.5-1.5B-Instruct										
Prompting	Direct Prompt	5.9	5.5	3.3	9.7	4.2	0.0	4.1	23.1	5.2
Prompting	ReAct	17.4	20.5	15.7	6.2	7.7	2.0	12.8	40.1	11.3
Prompting	Reflexion	35.3	22.2	21.7	13.6	19.4	3.7	21.8	55.8	21.9
RL Training	PPO	$64.8_{\pm 3.5}$	$40.5_{\pm 6.9}$	$57.1_{\pm 4.9}$	$60.6_{\pm 6.6}$	$46.4_{\pm 4.0}$	$47.4_{\pm 1.9}$	$54.4_{\pm 3.1}$	$73.8_{\pm 3.0}$	$51.5_{\pm 2.9}$
RL Training	RLOO	$88.3_{\pm 3.0}$	$52.8_{\pm 8.6}$	$71.0_{\pm 5.9}$	$62.8_{\pm 8.7}$	$66.4_{\pm 5.5}$	$56.9_{\pm 4.7}$	$69.7_{\pm 2.5}$	$73.9_{\pm 5.6}$	$52.1_{\pm 6.7}$
RL Training	GRPO	$85.3_{\pm 1.5}$	$53.7_{\pm 8.0}$	$84.5_{\pm 6.8}$	$78.2_{\pm 7.9}$	$59.7_{\pm 5.0}$	$53.5_{\pm 5.6}$	$72.8_{\pm 3.6}$	$75.8_{\pm 3.5}$	$56.8_{\pm 3.8}$
RL Training	GiGPO	$93.6_{\pm 6.7}$	$82.1_{\pm 8.0}$	$90.3_{\pm 6.9}$	$89.7_{\pm 5.5}$	$77.5_{\pm 4.0}$	$73.3_{\pm 11.4}$	$84.8_{\pm 5.5}$	$81.6_{\pm 4.6}$	$67.5_{\pm 4.8}$
RL Training	**PTGPO (Ours)**	$\mathbf{96.6}_{\pm 3.6}$	$\mathbf{87.6}_{\pm 6.9}$	$\mathbf{97.4}_{\pm 2.3}$	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{86.0}_{\pm 6.6}$	$\mathbf{89.3}_{\pm 8.2}$	$\mathbf{92.6}_{\pm 2.5}$	$\mathbf{87.1}_{\pm 0.4}$	$\mathbf{74.3}_{\pm 1.5}$
Qwen2.5-7B-Instruct										
Prompting	Direct Prompt	33.4	21.6	19.3	6.9	2.8	3.2	14.8	26.4	7.8
Prompting	ReAct	48.5	35.4	34.3	13.2	18.2	17.6	31.2	46.2	19.5
Prompting	Reflexion	62.0	41.6	44.9	30.9	36.3	23.8	42.7	58.1	28.8
RL Training	PPO	$92.3_{\pm 4.0}$	$64.0_{\pm 8.4}$	$92.5_{\pm 2.4}$	$89.5_{\pm 7.0}$	$80.3_{\pm 2.0}$	$68.8_{\pm 8.3}$	$80.4_{\pm 2.7}$	$81.4_{\pm 3.1}$	$68.7_{\pm 5.1}$
RL Training	RLOO	$87.6_{\pm 4.3}$	$78.2_{\pm 8.3}$	$87.3_{\pm 5.8}$	$81.3_{\pm 7.6}$	$71.9_{\pm 5.2}$	$48.9_{\pm 8.4}$	$75.5_{\pm 4.6}$	$80.3_{\pm 3.2}$	$65.7_{\pm 4.0}$
RL Training	GRPO	$90.8_{\pm 5.1}$	$66.1_{\pm 6.7}$	$89.3_{\pm 5.4}$	$74.7_{\pm 6.9}$	$72.5_{\pm 5.4}$	$64.7_{\pm 7.3}$	$77.6_{\pm 5.2}$	$79.3_{\pm 2.8}$	$66.1_{\pm 3.7}$
RL Training	GiGPO	$95.2_{\pm 5.9}$	$86.7_{\pm 4.3}$	$96.4_{\pm 3.4}$	$93.7_{\pm 5.9}$	$84.1_{\pm 5.9}$	$92.8_{\pm 2.4}$	$91.9_{\pm 1.6}$	$82.5_{\pm 1.3}$	$70.5_{\pm 1.1}$
RL Training	**PTGPO (Ours)**	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{96.0}_{\pm 3.5}$	$\mathbf{97.3}_{\pm 2.4}$	$\mathbf{97.6}_{\pm 4.1}$	$\mathbf{86.9}_{\pm 5.0}$	$\mathbf{97.2}_{\pm 2.4}$	$\mathbf{96.0}_{\pm 1.1}$	$\mathbf{90.7}_{\pm 2.0}$	$\mathbf{82.0}_{\pm 1.6}$

Table 2. Performance on Sokoban. We train each RL method with 3 random seeds and report the mean and standard deviation. The best performances are highlighted in bold.

Type	Method	Sokoban [6×6]
Prompting	Direct Prompt	11.7
RL Training	GRPO	$73.9_{\pm 1.6}$
RL Training	GiGPO	$77.6_{\pm 2.5}$
RL Training	**PTGPO**	$\mathbf{85.2}_{\pm 1.6}$

Finally, PTGPO is combined with the base episode advantage:

$$A^{\text{episode},(i)} = \frac{\hat{R}^{(i)}_{T_i} - \mu_R}{\sigma_R}, \tag{10}$$

where μ_R and σ_R are the mean and standard deviation of terminal rewards across the group.

$$A^{(i)}_k = A^{\text{episode},(i)} + A^{\text{PT},(i)}_k. \tag{11}$$

Here $A^{\text{episode},(i)}$ is sequence-level and shared by all actions in the response, while $A^{\text{PT},(i)}_k$ provides step-level credit by decomposing the suffix-dependent return along the prefix tree. Fig. 2 gives an intuitive example of the resulting advantage decomposition. PTGPO shifts credit from uniformly rewarding or penalizing every step to emphasizing the key decision points that separate successful and failed trajectories.

Then the clipped policy optimization objective of PTGPO is:

$$\mathcal{J}(\theta) = \mathbb{E}\Big[\min\Big(\rho^{(i)}_k(\theta)A^{(i)}_k, \text{clip}\left(\rho^{(i)}_k(\theta), 1-\epsilon, 1+\epsilon\right)A^{(i)}_k\Big)\Big]. \tag{12}$$

where

$$\rho^{(i)}_k(\theta) = \frac{\pi_\theta(a^{(i)}_k \mid s^{(i)}_k)}{\pi_{\theta_{\text{old}}}(a^{(i)}_k \mid s^{(i)}_k)}. \tag{13}$$

3. EXPERIMENTS

3.1. Setup

Benchmarks. We train agents on three challenging multi-turn benchmarks: ALFWorld [3], WebShop [5] and Sokoban [7]. ALFWorld is a text-based embodied environment where agents complete household tasks through multi-step interaction. WebShop is a web-based interactive benchmark where LLM agents navigate a simulated shopping website to purchase items matching user instructions. Sokoban is a visual grid-based game benchmark where agents move boxes onto target locations through multi-step actions.

Baselines. We compare PTGPO against baselines across paradigms: (1) Closed-source models: GPT-4o [16] and Gemini-2.5-Pro [17]. (2) Prompting methods: ReAct [18] and Reflexion [19]. (3) RL training methods: PPO [20], RLOO [21], GRPO [12] and GiGPO [15], covering both critic-based and group-based policy optimization methods. For VLMs, we focus on comparisons with group-based RL methods.

Implementation. We use Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct [22] as the base models for ALFWorld and WebShop, and Qwen2.5-VL-3B-Instruct [23] for Sokoban. For fair comparison, all RL methods use identical training configurations. We train each method for 160 training iterations with a discount factor of $\gamma = 0.95$ and a learning rate of 1×10^{-6}. In each iteration, we sample 16 training instances and generate a group of 8 trajectories for each instance, resulting in 128 rollouts per iteration. Each episode allows up to 50 environment steps in ALFWorld and up to 15 environment steps in WebShop and Sokoban. We use 8 NVIDIA H20 96GB GPUs for training.

Table 3. Ablation study of PTGPO on ALFWorld and WebShop using Qwen2.5-1.5B-Instruct, and on Sokoban using Qwen2.5-VL-3B-Instruct. The best performances are highlighted in bold.

Variant	ALFWorld	WebShop	Sokoban
PTGPO	**92.6**$_{\pm 2.5}$	74.3$_{\pm 1.5}$	**85.2**$_{\pm 1.6}$
Advantage Composition			
w/o Episode-level Adv	92.4$_{\pm 1.1}$	71.8$_{\pm 3.9}$	81.5$_{\pm 4.4}$
w/o Prefix-tree Adv	72.8$_{\pm 3.6}$	56.8$_{\pm 3.8}$	73.9$_{\pm 1.6}$
Aggregation Strategy			
Mean	84.3$_{\pm 1.6}$	71.4$_{\pm 2.1}$	82.3$_{\pm 2.3}$
Nearest Anchor	91.5$_{\pm 2.3}$	73.2$_{\pm 1.6}$	84.1$_{\pm 1.0}$
Prefix Depth			
$d = 2$	87.6$_{\pm 4.4}$	70.0$_{\pm 0.9}$	78.6$_{\pm 5.1}$
$d = 4$	88.3$_{\pm 2.2}$	72.9$_{\pm 1.0}$	83.3$_{\pm 6.6}$
$d = 8$	92.1$_{\pm 1.2}$	**74.5**$_{\pm 1.9}$	83.6$_{\pm 1.6}$
Expansion Threshold			
$\tau_{\text{expand}} = 4$	90.0$_{\pm 3.6}$	72.0$_{\pm 3.7}$	83.6$_{\pm 3.4}$
$\tau_{\text{expand}} = 6$	85.5$_{\pm 1.1}$	71.1$_{\pm 2.2}$	81.5$_{\pm 1.2}$
$\tau_{\text{expand}} = 8$	84.3$_{\pm 2.9}$	73.3$_{\pm 3.8}$	81.8$_{\pm 1.7}$

3.2. Experiment Results

3.2.1. Main Results

Table 1 shows that PTGPO consistently outperforms prompting methods and RL baselines across multi-turn agentic tasks. On Qwen2.5-1.5B, PTGPO achieves 92.6 on ALFWorld and 74.3% success on WebShop, improving over GRPO by 19.8 and 17.5 points, respectively. With Qwen2.5-7B, PTGPO further improves to 96.0 on ALFWorld and 82.0% success rate on WebShop. On the visual Sokoban benchmark, PTGPO reaches 85.2 as shown in Table 2, outperforming GRPO and GiGPO, demonstrating its effectiveness for both LLM and VLM agents. To further examine the training dynamics, Figure 3(a) reports the episode success rate on ALFWorld as a representative example. PTGPO reaches around 0.8 success rate after about 75 steps, while GiGPO reaches a similar level after about 105 steps and GRPO fails to do so within 160 steps, showing better training efficiency.

3.2.2. Ablation Studies

Table 3 evaluates the main components and design choices of PTGPO. Unless otherwise specified, the full method uses unrestricted prefix-tree expansion with $\tau_{\text{expand}} = 2$ and sums all valid prefix credits associated with each action. Removing the prefix-tree advantage reduces performance by 19.8, 17.5, and 11.3 points on ALFWorld, WebShop, and Sokoban, respectively, confirming that fine-grained prefix credit is the primary source of improvement. In comparison, removing the episode-level advantage has little effect on ALFWorld but causes larger drops on WebShop and Sokoban, indicating that the trajectory-level signal provides complementary global supervision.

We next compare different strategies for combining the credits assigned to the same action by multiple anchor states. Mean aggregation consistently underperforms the default summation strategy, with a particularly large drop on ALFWorld. Using only the nearest valid anchor performs substantially better than mean aggregation, but remains below credit summation on all three benchmarks. These results suggest that different anchor states provide complementary credit signals: restricting learning to the nearest anchor discards useful longer-range evidence, whereas averaging can dilute informative signals when only a small number of valid anchors are available. Summation preserves and accumulates these signals. Moreover, because an action can only receive credit from trees that reach it, the number of accumulated terms is bounded by the tree depth and is further limited by sparse state matching and the expansion threshold.

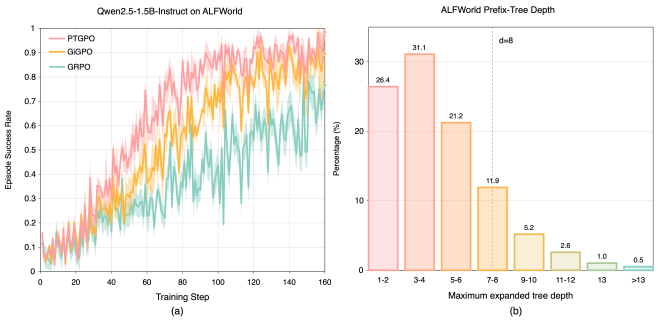

Fig. 3. (a) Episode Success Rate during Training on ALFWorld. (b) Distribution of maximum expanded tree depth at epoch 160 on ALFWorld.

Increasing the maximum prefix depth generally improves performance. Expanding the depth from 2 to 8 improves ALFWorld from 87.6 to 92.1 and WebShop from 70.0 to 74.5, while Sokoban improves from 78.6 to 83.6. Depth 8 is already close to unrestricted expansion, consistent with Fig. 3(b), which shows that most trees terminate within eight levels. Finally, increasing τ_{expand} generally degrades performance. Although a larger threshold yields more reliable node-level return estimates, it prevents many sparsely supported branches from being expanded and consequently removes useful fine-grained credits. Overall, $\tau_{\text{expand}} = 2$ provides the best balance between estimation reliability and credit coverage.

3.2.3. Computational Overhead

We measure the method-specific advantage computation cost on ALFWorld with Qwen2.5-1.5B-Instruct at epoch 160. PTGPO shares the same rollout, probability computation, and policy update pipeline as other group-based methods, with a common cost of about 210.968s per iteration, and introduces additional computation only for prefix-tree construction and prefix-wise advantage estimation. The method-specific advantage computation costs are 0.192s for GRPO, 0.559s for GiGPO, and 1.713s for PTGPO. Despite the additional overhead introduced by prefix-tree construction, PTGPO's extra computation accounts for only 0.812% of the common training cost and has a negligible impact on overall training efficiency.

4. CONCLUSION

In this paper, we propose PTGPO, a novel reinforcement learning method for long-horizon interactive agents with sparse rewards. PTGPO mitigates the credit assignment problem by decomposing trajectory returns through prefix-wise conditional mean increments, assigning step-level credit to actions according to how future action prefixes change expected returns. Experiments on interactive benchmarks show that PTGPO consistently improves over existing RL baselines. Further analyses demonstrate the importance of prefix-tree advantages and show that PTGPO introduces only minor computational overhead without requiring extra rollouts or critic models.

5. REFERENCES

[1] Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al., "Deepseek-v3. 2: Pushing the frontier of open large language models," *arXiv preprint arXiv:2512.02556*, 2025.

[2] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al., "Qwen3 technical report," *arXiv preprint arXiv:2505.09388*, 2025.

[3] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht, "Alfworld: Aligning text and embodied environments for interactive learning," *arXiv preprint arXiv:2010.03768*, 2020.

[4] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar, "Voyager: An open-ended embodied agent with large language models," *arXiv preprint arXiv:2305.16291*, 2023.

[5] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan, "Webshop: Towards scalable real-world web interaction with grounded language agents," *Advances in Neural Information Processing Systems*, vol. 35, pp. 20744–20757, 2022.

[6] Boyu Gou, Demi Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su, "Navigating the digital world as humans do: Universal visual grounding for gui agents," in *International Conference on Learning Representations*, 2025, vol. 2025, pp. 30851–30883.

[7] Max-Philipp B. Schrader, "gym-sokoban," `https://github.com/mpSchrader/gym-sokoban`, 2018.

[8] Zelai Xu, Chao Yu, Fei Fang, Yu Wang, and Yi Wu, "Language agents with reinforcement learning for strategic play in the werewolf game," *arXiv preprint arXiv:2310.18940*, 2023.

[9] Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, et al., "Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning," in *International Conference on Learning Representations*, 2025, vol. 2025, pp. 79791–79821.

[10] 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.

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

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

[13] Zixuan Wang, Yuchen Yan, Hongxing Li, Teng Pan, Dingming Li, Ruiqing Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen, "Milestone-guided policy learning for long-horizon language agents," *arXiv preprint arXiv:2605.06078*, 2026.

[14] Hui-Ze Tan, Xiao-Wen Yang, Hao Chen, Jie-Jing Shao, Yi Wen, Yuteng Shen, Weihong Luo, Xiku Du, Lan-Zhe Guo, and Yu-Feng Li, "Hindsight credit assignment for long-horizon llm agents," *arXiv preprint arXiv:2603.08754*, 2026.

[15] Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An, "Group-in-group policy optimization for llm agent training," *Advances in Neural Information Processing Systems*, vol. 38, pp. 46375–46408, 2026.

[16] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al., "Gpt-4o system card," *arXiv preprint arXiv:2410.21276*, 2024.

[17] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al., "Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities," *arXiv preprint arXiv:2507.06261*, 2025.

[18] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao, "React: Synergizing reasoning and acting in language models," *arXiv preprint arXiv:2210.03629*, 2022.

[19] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao, "Reflexion: Language agents with verbal reinforcement learning," *Advances in neural information processing systems*, vol. 36, pp. 8634–8652, 2023.

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

[21] 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)*, 2024, pp. 12248–12267.

[22] 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*, 2024.

[23] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al., "Qwen2.5-vl technical report," *arXiv preprint arXiv:2502.13923*, 2025.