HIERARCHY-OF-GROUPS POLICY OPTIMIZATION FOR LONG-HORIZON AGENTIC TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Group-based reinforcement learning (RL), such as GRPO, has advanced the capabilities of large language models on long-horizon agentic tasks. To enable more fine-grained policy updates, recent research has increasingly shifted toward *stepwise* group-based policy optimization, which treats each step in a rollout trajectory independently while using a memory module to retain historical context. However, we find a key issue in estimating stepwise relative advantages, namely *context inconsistency*, where steps within the same group may differ in their historical contexts. Empirically, we reveal that this issue can lead to severely biased advantage estimation, thereby degrading policy optimization significantly. To address the issue, in this paper, we propose Hierarchical-of-Groups Policy Optimization (HGPO) for long-horizon agentic tasks. Specifically, within a group of rollout trajectories, HGPO assigns each step to multiple hierarchical groups according to the *consistency* of historic contexts. Then, for each step, HGPO computes distinct advantages within each group and aggregates them with an adaptive weighting scheme. In this way, HGPO can achieve a favorable bias-variance trade-off in stepwise advantage estimation, without extra models or rollouts. Evaluations on two challenging agentic tasks, ALFWorld and WebShop with Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct, show that HGPO significantly outperforms existing agentic RL methods under the same computational constraints.

1 INTRODUCTION

Versatile agents powered by Large Language Models (LLMs) can perceive, reason, and act in complex, open-ended environments (Achiam et al., 2023; Team et al., 2023; Yang et al., 2024; Liu et al., 2024). Representative applications include embodied assistants navigating simulated homes (Shridhar et al., 2021; Li et al., 2024), web navigators completing browsing tasks (Furuta et al., 2024; Zheng et al., 2024; Gou et al., 2025), and autonomous explorers in interactive computer games (Wang et al., 2024a;b). Beyond language and vision understanding, such agents are expected to perform long-horizon planning and robust decision-making.

Deep reinforcement learning (RL) (Sutton & Barto, 2018) has emerged as a key paradigm for enhancing agent performance in the post-training stage (OpenAI, 2024; Guo et al., 2025). In particular, group-based RL methods such as RLOO (Kool et al., 2019; Ahmadian et al., 2024), GRPO (Shao et al., 2024), DAPO (Yu et al., 2025c), Clip-Cov (Cui et al., 2025), and GSPO (Zheng et al., 2025) have demonstrated strong performance in large-scale RL training while requiring fewer computational resources. These methods have proven effective in single-turn tasks such as mathematical reasoning (Liu et al., 2025; Yu et al., 2025c) and code generation (Wei et al., 2025a). To extend this paradigm to multi-turn settings, approaches such as RAGEN (Wang et al., 2025d) and Search-R1 (Jin et al., 2025a) adopt a *trajectory-wise* policy optimization framework, which concatenates environment states and model outputs across turns to enable multi-turn rollouts. However, this framework suffers from a major limitation: the effective context length grows rapidly with the number of interaction turns, leading to severe context explosion.

To address this, recent research has shifted toward the *stepwise* policy optimization framework (Feng et al., 2025b; Luo et al., 2025c; Chen et al., 2025b; Team, 2025; Yu et al., 2025b; Wang et al., 2025c), which treats each step within a rollout trajectory independently while leveraging a memory module to retain historical context. This design allows for flexible context management and highly scalable

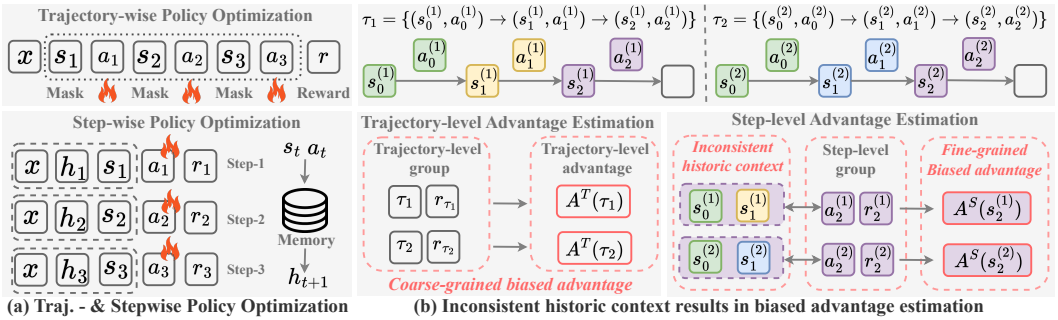

Figure 1: Figure (a) compares trajectory-wise and stepwise policy optimization frameworks. Given two example group trajectories, Figure (b) illustrates trajectory-level and step-level grouping with their corresponding advantage estimations. Best viewed in color.

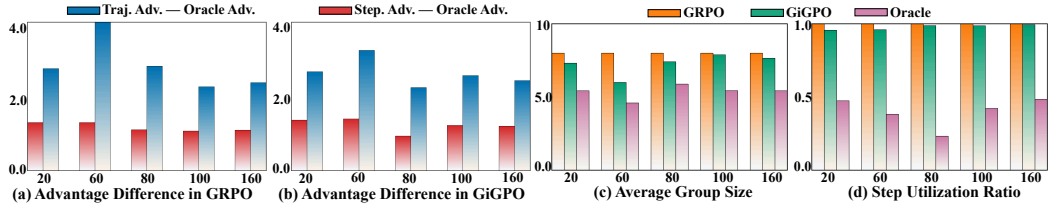

Figure 2: Statistics of GRPO and GiGPO. Figures (a) and (b) present the advantage differences relative to Oracle advantages for GRPO and GiGPO, respectively. Figures (c) and (d) report the average group size and the proportion of Oracle steps, respectively.

RL training. A comparison of the two frameworks is illustrated in Figure 1. (a). Building on the stepwise framework, group-based RL methods such as GRPO (Shao et al., 2024) can be adapted into stepwise group-based variants for long-horizon agentic tasks. Furthermore, to enable finer-grained credit assignment, GiGPO (Feng et al., 2025b) extends GRPO by estimating additional step-level advantages within groups where all steps share the same current state.

However, we find a key issue in estimating stepwise relative advantages, namely *context inconsistency*. This issue arises when steps in the same group differ in their historical contexts. As illustrated in Figure 1. (b), given two group trajectories τ_1 and τ_2, the step-level group of state s_2 (purple) contains steps with *inconsistent* historic contexts. In this case, the estimated advantage within the step-level group becomes *biased*, failing to reflect the true effect of current states and actions conditioned on prior context. To further examine the impact, we conduct a pilot empirical study. We introduce the notion of *Oracle* groups, where all steps share not only the same current state but also identical historic contexts. During GRPO and GiGPO training, we track group sizes, step counts, and estimated advantages for these Oracle groups, alongside trajectory-level and step-level advantages for comparison. As shown in Figures 2. (a) and (b), both trajectory-level and step-level advantages exhibit considerable estimation bias, with trajectory-level bias being substantially larger, which highlights that context inconsistency severely distorts advantage estimation.

A straightforward solution is using only Oracle steps for policy optimization. However, as shown in Figures 2. (c) and (d), Oracle steps are generally scarce within trajectories (i.e., their ratio is low), making this approach inefficient. Moreover, the average group size of Oracle steps is small, which increases the variance of estimated advantages and undermines the stability of RL training.

To address the above challenges, in this paper, we propose Hierarchy-of-Groups Policy Optimization (HGPO), a novel RL training algorithm that introduces a better advantage estimator capable of low-bias and balanced-variance. Specifically, HGPO is built on two key components: context-aware hierarchical grouping and adaptive weighting advantage estimation. First, within each rollout, HGPO groups steps that share the same current state and further assigns them to multiple hierarchical groups according to their historical contexts. This hierarchical structure captures advantages at different context depths, improving data utilization and reducing variance. Second, HGPO aggregates the group advantages using an adaptive weighting scheme: groups with more consistent

historical contexts are assigned larger weights, thereby lowering estimation bias. In this way, HGPO produces more reliable stepwise advantage estimates for policy optimization. We evaluate HGPO on two challenging agentic benchmarks, ALFWorld and WebShop, using Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct. Results show that HGPO consistently outperforms existing baselines while maintaining the same GPU memory usage, using identical LLM rollouts, and incurring minimal additional time cost. Our main contributions are summarized as follows:

- ***Revealing context inconsistency.*** We reveal the issue of context inconsistency in stepwise group-based RL and empirically demonstrate that it introduces significant bias in advantage estimation, thereby degrading policy optimization.
- ***Proposing a novel policy optimization algorithm.*** We introduce Hierarchy-of-Groups Policy Optimization, which constructs hierarchical groups for each step based on historical context and adaptively aggregates their advantages.
- ***Achieving strong empirical performance.*** HGPO achieves state-of-the-art results on two challenging agentic benchmarks, significantly outperforming existing baselines under the same computational constraints.

2 Related work

LLM-based decision-making agents. Large language models (LLMs) have been widely adopted as autonomous agents across diverse domains, including device control (Zhang & Zhang, 2024; Hong et al., 2024; Gur et al., 2024; Hu et al., 2024), code generation (Zhang et al., 2024b), game interaction (Wang et al., 2024a; Tan et al., 2025), and robotics (Zitkovich et al., 2023). Early approaches often relied on fixed pre-trained models guided by structured prompting, such as ReAct (Yao et al., 2023) and Reflexion (Shinn et al., 2024), augmented with memory and retrieval mechanisms (Wang et al., 2024b; Tan et al., 2024) or tool integration (Schick et al., 2023; Xie et al., 2024; Zhang et al., 2024a). While such methods are simple and require no additional training, they remain limited in applicability to domain-specific tasks, largely due to the lack of specialized knowledge in the pre-training of the base models.

Reinforcement learning for LLM-based agents. Reinforcement learning (RL) (Sutton & Barto, 2018) has been central to adapting large language model (LLM) agents to dynamic and open-ended environments. Early work applied classic algorithms such as DQN (Mnih et al., 2015) to text games (Narasimhan et al., 2015), followed by value-based methods like PPO (Schulman et al., 2017) and AWR (Peng et al., 2019) in interactive domains including mobile control (Rawles et al., 2024), embodied tasks in ALFWorld (Shridhar et al., 2021), and card games (Brockman, 2016). More recent research has extended RL to web and application environments (Qian et al., 2025; Sun et al., 2025), with methods such as ArCHer (Zhou et al., 2024b), AgentQ (Putta et al., 2024), CoSo (Feng et al., 2025a), and LOOP (Chen et al., 2025a). In parallel, RL has also become integral to LLM training itself, with RLHF (Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022; Rafailov et al., 2024) aligning models with human preferences, and group-based RL algorithms emerging as scalable and efficient alternatives to PPO. Approaches such as GRPO (Shao et al., 2024), Dr. GRPO (Liu et al., 2025), Clip-Cov (Cui et al., 2025), GSPO (Zheng et al., 2025), and DAPO (Yu et al., 2025c) avoid value networks by estimating advantages over groups of samples. However, most of these methods are designed for single-turn interactions and thus struggle with context consistency in long-horizon agentic tasks.

Long-horizon agentic reinforcement learning. Long-horizon agentic RL (Laban et al., 2025; Zhang et al., 2025; Zhou et al., 2025a; Luo et al., 2025d; Wang et al., 2025a) extends LLMs from single-turn generation to multi-turn decision-making, where RL equips them with planning (Hao et al., 2023; Zhou et al., 2024a; Song et al., 2024), reasoning (Chu et al., 2025), and memory (Jin et al., 2024; Chhikara et al., 2025; Zhou et al., 2025b) capabilities for sustained interaction in dynamic environments. Applications span code generation (Jiang et al., 2024; Gehring et al., 2025; Jain et al., 2025; Chen et al., 2025c; Jin et al., 2025b), software engineering (Wei et al., 2025b; Luo et al., 2025a; Shen et al., 2025; Wang et al., 2024c; Lin et al., 2025), and GUI interaction (Wei et al., 2025c; Lu et al., 2025; Luo et al., 2025b; Qin et al., 2025). Recent advances include long-horizon policy optimization frameworks (Wang et al., 2025d; Jin et al., 2025a) that optimize over multi-turn rollouts, and stepwise policy optimization methods (Feng et al., 2025b; Luo et al., 2025c; Chen et al., 2025b; Team, 2025) that treat each step inde-

pendently while retaining history through memory modules. Yet, stepwise methods often suffer from context inconsistency across long horizons, limiting their effectiveness in complex agentic tasks.

3 PRELIMINARIES

Problem setup of long-horizon agentic tasks. Unlike single-turn tasks, long-horizon agentic tasks require an LLM agent to interact with the environment across multiple turns to accomplish a goal. Formally, given a task example $\boldsymbol{x} \in p(X)$, which typically includes a fixed task-related description, an LLM-based agent π_θ parameterized by θ observes an environment state $\boldsymbol{s}_t \in \mathcal{S}$ at each turn t and generates a textual action $\boldsymbol{a}_t \in \mathcal{V}^n$, where $\mathcal{V}$ denotes the token vocabulary and n is the maximum generation length. Here $t = (1, 2, \ldots, T)$, with T being the maximum number of interaction turns. In this paper, we focus on the sparse delayed reward setting, where the environment provides a scalar reward $r_t \in \mathcal{R}$ only at the final step of a trajectory $\tau = \{(\boldsymbol{s}_1, \boldsymbol{a}_1), \ldots, (\boldsymbol{s}_T, \boldsymbol{a}_T)\}$.

Trajectory-wise vs. stepwise policy optimization. Conventional trajectory-wise policy optimization frameworks (Wang et al., 2025d; Jin et al., 2025a; Wang et al., 2025b; Yu et al., 2025a) typically concatenate the full interaction history of a rollout trajectory τ for policy optimization, i.e., $\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_{0:t}, \boldsymbol{x})$. However, as the number of turns T grows, the context length increases rapidly, which limits the scalability and feasibility of long-horizon RL training. In contrast, stepwise policy optimization frameworks (Feng et al., 2025b; Luo et al., 2025c; Chen et al., 2025b; Team, 2025) decouple the trajectory into individual steps while leveraging a memory module that maintains $K \ll T$ historical contexts. This memory module is updated with the latest K interactions, keeping the prompt length relatively stable and enabling more scalable RL training.

Group-based reinforcement learning. Unlike PPO (Schulman et al., 2017), which estimates advantages using an additional value function, group-based reinforcement learning (RL) algorithms such as GRPO (Shao et al., 2024) compute advantages directly from the statistics of a sampled group of trajectories G_τ. Specifically, GRPO was originally designed for single-turn tasks under a trajectory-wise policy optimization framework. To extend it to long-horizon tasks, we adapt it to the stepwise setting and calculate the trajectory-level advantage as:

$$A^T(\tau_i) = \left(R(\tau_i) - 1/|G_\tau| \sum\nolimits_{j \in G_\tau} R(\tau_i)\right) / \sigma_{G_\tau}, \tag{1}$$

where σ_{G_τ} denotes the standard deviation of rewards within the group G_τ. This trajectory-level computation assigns the same advantage value to every step in trajectory τ_i, thereby overlooking the finer credit assignment required within a trajectory. To address this limitation, one can instead adopt a step-level group relative advantage estimator (Feng et al., 2025b). Here, steps with identical current states $\tilde{\boldsymbol{s}}_i$ across all group trajectories are clustered into step-level groups $G_{\tilde{\boldsymbol{s}}_i}$, and their advantages are computed as:

$$A^S(\tilde{\boldsymbol{s}}_i) = \left(R(\tilde{\boldsymbol{s}}_i) - 1/|G_{\tilde{\boldsymbol{s}}_i}| \sum\nolimits_{j \in G_{\tilde{\boldsymbol{s}}_i}} R(\tilde{\boldsymbol{s}}_j)\right) / \sigma_{G_{\tilde{\boldsymbol{s}}_i}}. \tag{2}$$

Compared to Eq. (1), the step-level estimator in Eq. (2) provides more fine-grained and effective credit assignment across steps within the same trajectory.

4 TRAINING AGENTS WITH HGPO FOR LONG-HORIZON AGENTIC TASKS

In this section, we will first reveal the issue of context inconsistency, introduce our motivation, and propose Hierarchy-of-Group Policy Optimization (HGPO).

4.1 THE ISSUE OF CONTEXT INCONSISTENCY

As discussed, stepwise policy optimization introduces per-step context management for long-horizon RL. However, we find a key issue: context inconsistency. Specifically, as illustrated in Figure 1, steps within a step-level anchor group that share the same current state may have *distinct historical contexts* in their memory modules, resulting in biased advantage estimates. This issue also arises in trajectory-level grouping and advantage computation. Empirically, as shown in Figure 2, we find that both trajectory-level and step-level estimates exhibit substantial bias, with

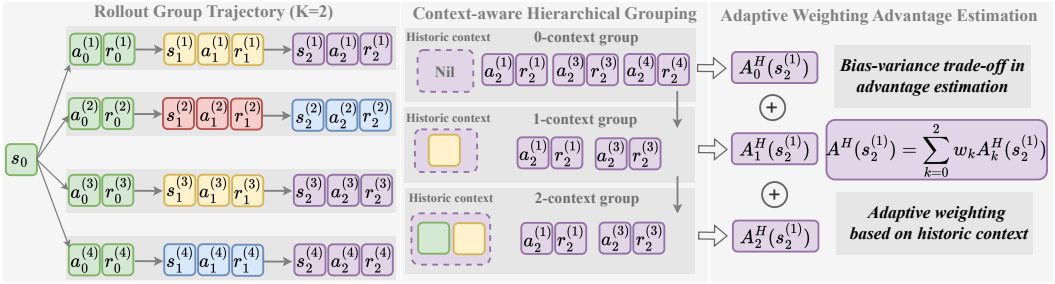

Figure 3: Overview of HGPO. The LLM-based agent interacts with a set of environments initialized from the same state s_0, producing four group trajectories (states with the same color are identical). HGPO comprises two key components: context-aware hierarchical grouping and adaptive weighted advantage computation. For illustration, consider the state s_2 (purple). First, HGPO assigns s_2 into three hierarchical groups according to its historical contexts. Then, it computes the final advantage estimate by adaptively aggregating the weighted advantages from these groups.

trajectory-level bias being more pronounced. This observation confirms that context inconsistency can severely distort advantage estimation, thereby degrading policy optimization. A naive solution is to use only Oracle steps for policy optimization. However, as Figure 2 shows, Oracle steps are scarce in trajectories, making such an approach highly inefficient because most steps are ignored. Additionally, the small average group size of Oracle steps can lead to high variance in advantage estimates, further destabilizing RL training. Motivated by these challenges, we propose leveraging a hierarchy-of-groups structure, which enables more accurate advantage estimation to reduce bias while maintaining low variance.

4.2 Hierarchy-of-Groups Policy Optimization

In this subsection, we introduce HGPO as shown in Figure 3, consisting of context-aware hierarchical grouping and adaptive weighting advantage estimation.

Context-aware hierarchical grouping. We begin by introducing *context-aware hierarchical grouping*, which organizes steps into multi-level groups according to their historical contexts. The key intuition is that the advantage of each step should be evaluated relative to different historical contexts to obtain more accurate estimates. Specifically, we first group together steps that share the same current state, and then, within each group, we construct multiple hierarchical groups based on the consistency of their historical contexts. Steps with longer common histories are assigned to higher-level hierarchical groups. This hierarchy-of-groups structure enables more fine-grained comparisons and brings two main benefits: (i) it improves step utilization for advantage estimation, and (ii) it reduces the variance of estimated advantages.

Formally, let the i-th trajectory be $\tau_i = \{(\boldsymbol{s}_1^{(i)}, \boldsymbol{a}_1^{(i)}), (\boldsymbol{s}_2^{(i)}, \boldsymbol{a}_2^{(i)}), \ldots, (\boldsymbol{s}_T^{(i)}, \boldsymbol{a}_T^{(i)})\}$, and let K denote the maximum context length. We define a k-step context operator for the t-th step as:

$$\mathcal{C}_k(\boldsymbol{s}_t^{(i)}) = \begin{cases} \left(\boldsymbol{s}_{t-k}^{(i)}, \boldsymbol{s}_{t-k+1}^{(i)}, \cdots, \boldsymbol{s}_t^{(i)}\right), t \geq k, \\ \left(\boldsymbol{s}_0^{(i)}, \boldsymbol{s}_1^{(i)}, \cdots, \boldsymbol{s}_t^{(i)}\right), t < k, \end{cases} \tag{3}$$

where $k \in [0, K]$. This operator returns the k historical states preceding the current state. Based on this operator, we define the k-th hierarchical group for the t-th step as:

$$G_k^H(\boldsymbol{s}_t^{(i)}) \;=\; \left\{(j,n) \in \mathcal{I} : \mathcal{C}_k(\boldsymbol{s}_t^{(i)}) = \mathcal{C}_k(\boldsymbol{s}_n^{(j)})\right\}, \tag{4}$$

where the index set $\mathcal{I} = \{(i,t) \mid 1 \leq i \leq N, , 1 \leq t \leq T\}$. Considering all hierarchical groups, the resulting hierarchy-of-groups structure satisfies:

$$G_0^H(\boldsymbol{s}_t^{(i)}) \supseteq G_1^H(\boldsymbol{s}_t^{(i)}) \supseteq \cdots \supseteq G_K^H(\boldsymbol{s}_t^{(i)}), \qquad \left|G_0^H(\boldsymbol{s}_t^{(i)})\right| \geq \cdots \geq \left|G_K^H(\boldsymbol{s}_t^{(i)})\right|. \tag{5}$$

When $K = 0$, the hierarchy-of-groups degenerates to the step-level grouping $G_0^H(\boldsymbol{s}_t^{(i)})$ used in (Feng et al., 2025b). Importantly, the entire context-aware hierarchical grouping procedure operates fully offline: it requires only hashmap lookups over existing rollouts, without relying on additional models or extra data collection.

Adaptive weighting advantage estimation. Intuitively, higher-level hierarchical groups yield more accurate advantage comparisons since they incorporate richer historical context. Building on this insight, we introduce an adaptive weighting scheme that integrates information across all hierarchical groups with appropriately assigned weights, thereby enabling stable and efficient estimation of group-relative advantages. Formally, the advantage estimation for the k-th hierarchical group is defined as:

$$A_k^H(\boldsymbol{s}_t^{(i)}) = \left(R(\boldsymbol{s}_t^{(i)}) - 1/|G_k^H| \sum\nolimits_{j \in G_k^H} R(\boldsymbol{s}_t^{(i)}) \right) / \sigma_{G_k^H}. \tag{6}$$

Finally, the advantage aggregated from K hierarchical groups is denoted by:

$$A^H(\boldsymbol{s}_j^{(i)}) = \sum\nolimits_{k=0}^{K} \boldsymbol{w}_k A_k^H(\boldsymbol{s}_j^{(i)}), \tag{7}$$

where the adaptive weight $\boldsymbol{w}_k = \frac{(k+1)^\alpha}{\sum_k (k+1)^\alpha}$ ($\alpha \geq 0$). It is worth noting that Eq. (7) fuses advantage information along the hierarchy-of-groups in Eq. (5): higher-level groups are preferred due to stronger context consistency. Besides, for each step $(\boldsymbol{s}_t^{(i)}, \boldsymbol{a}_t^{(i)})$ we compute its stepwise reward $r_t^{(i)} = \sum_{j=t}^{T} \gamma^{j-t} r_j^{(i)}$ (Feng et al., 2025b), where $\gamma \in (0, 1]$ is the discount factor. In this way, we can obtain a stepwise reward for each step in the trajectory.

The objective for policy optimization. The policy optimization objective of HGPO is:

$$\begin{aligned} \mathcal{J}_{\mathrm{HGPO}}(\theta) = \mathbb{E}\Bigg[\frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} \min\Big(\rho_\theta(\boldsymbol{a}_t^{(i)}) A^H(\boldsymbol{s}_t^{(i)}),\ \mathrm{clip}\big(\rho_\theta(\boldsymbol{a}_t^{(i)}), 1 \pm \epsilon\big) A^H(\boldsymbol{s}_t^{(i)}) \Big) \Bigg] \\ - \beta \mathbb{D}_{\mathrm{KL}}\big(\pi_\theta(\cdot \mid x) \,\|\, \pi_{\mathrm{ref}}(\cdot \mid x)\big), \end{aligned} \tag{8}$$

where $\rho_\theta(\boldsymbol{a}_t^{(i)}) = \frac{\pi_\theta(\boldsymbol{a}_t^{(i)} | \boldsymbol{s}_t^{(i)}, x)}{\pi_{\theta_{\mathrm{old}}}(\boldsymbol{a}_t^{(i)} | \boldsymbol{s}_t^{(i)}, x)}$ is the importance sampling ratio, β controls the strength of the KL penalty. The pseudo-code is shown in Algorithm 1 of Appendix A.

Proposition 4.1 (Bias-variance trade-off in HGPO) *Let b_k and v_k denote the bias and variance of the estimated advantage A_k^H within the k-th group G_k^H. Based on the following conditions: (1)* Bias decreases monotonically*, i.e., $B_T \geq b_0 \geq b_1 \cdots \geq b_K \geq 0$; (2)* Variance increases monotonically and independently*, i.e., $v_0 \leq v_1 \leq \cdots \leq v_K \leq V_T$, the bias and variance of the estimator A^H are*

$$\begin{aligned} \textit{Bias}[A^H] &= \textit{Bias}\left[\sum\nolimits_{k=0}^{K} w_k A_k^H \right] = \sum\nolimits_{k=0}^{K} w_k b_k, \\ \textit{Var}[A^H] &= \textit{Var}\left[\sum\nolimits_{k=0}^{K} w_k A_k^H \right] = \sum\nolimits_{k=0}^{K} w_k^2 \textit{Var}[A_k^H] = \sum\nolimits_{k=0}^{K} w_k^2 v_k. \end{aligned}$$

Furthermore, the bias and variance satisfy that

$$\begin{aligned} b_K = \sum\nolimits_{k=0}^{K} w_k b_K \leq & \textit{Bias}[A^H] \leq \sum\nolimits_{k=0}^{K} w_k b_0 = b_0 \leq B_T, \\ \frac{1}{K(K+1)^{2\alpha}} v_0 \leq \sum_{k=0}^{K} w_k^2 v_0 \leq & \textit{Var}[A^H] \leq \sum_{k=0}^{K} w_k^2 v_K \leq \frac{(K+1)^{2\alpha}}{K} v_K, \end{aligned}$$

where B_T, b_0, b_K and v_T, V_0, v_K denote the bias and variance of the trajectory-level, step-level, and Oracle advantage, respectively. *Overall, the bias of the HGPO advantage estimator is lower than that of both trajectory- and step-level estimators, while its variance trades off against the step-level and Oracle estimators depending on the number of hierarchical groups K and the weight parameter α.* Proof and more details are provided in Appendix B.

5 Experiments

5.1 Experiment Setup

Agentic benchmarks. We train the LLM agents on two challenging benchmarks: ALFWorld (Shridhar et al., 2021) and WebShop (Yao et al., 2022), which are designed to assess the ability of LLM agents to perform multi-step decision-making. The details are shown in Appendix C.2.

Table 1: Performance comparison on ALFWorld and WebShop. For ALFWorld, we report the overall success rate (↑) for both *in-distribution* (In-Success) and *out-of-distribution* tasks (Out-Success). For WebShop, we report the average task score (↑) and the average task success rate (↑). Most results are averaged over 3 random seeds during testing. The best results are highlighted in bold.

Model	Type	Method	ALFWorld		WebShop	
			In-Success	Out-Success	Task Scores	Task Success Rates
Closed	Prompting	GPT-4o	48.0	46.0	31.8	23.7
	Prompting	Gemini-2.5-Pro	60.3	50.5	42.5	35.9
Qwen2.5-1.5B-Instruct	Prompting	Qwen2.5	4.1	-	23.1	5.2
	Prompting	ReAct	12.8	-	40.1	11.3
	Prompting	Reflexion	21.8	-	55.8	21.9
	RL Training	PPO (with critic)	$54.4_{\pm 3.1}$	-	$73.8_{\pm 3.0}$	$51.5_{\pm 2.9}$
	RL Training	RLOO	$69.7_{\pm 2.5}$	$68.7_{\pm 10.7}$	$73.9_{\pm 5.6}$	$52.1_{\pm 6.7}$
	RL Training	GRPO	$72.8_{\pm 3.6}$	$70.1_{\pm 2.5}$	$75.8_{\pm 3.5}$	$56.8_{\pm 3.8}$
	RL Training	GiGPO (K=2)	$85.42_{\pm 1.32}$	$80.72_{\pm 1.62}$	$84.52_{\pm 0.98}$	$69.79_{\pm 0.59}$
	RL Training	**HGPO** (K=2)	$\mathbf{89.58}_{\pm 0.45}$	$\mathbf{80.73}_{\pm 2.38}$	$\mathbf{87.53}_{\pm 0.77}$	$\mathbf{72.66}_{\pm 1.78}$
	RL Training	GiGPO (K=4)	$85.15_{\pm 2.81}$	$80.98_{\pm 0.45}$	$88.5_{\pm 0.49}$	$74.08_{\pm 0.98}$
	RL Training	**HGPO** (K=4)	$\mathbf{92.45}_{\pm 0.81}$	$\mathbf{89.06}_{\pm 2.34}$	$\mathbf{88.90}_{\pm 0.90}$	$\mathbf{75.91}_{\pm 1.19}$
Qwen2.5-7B-Instruct	Prompting	Qwen2.5	14.8	-	26.4	7.8
	Prompting	ReAct	31.2	-	46.2	19.5
	Prompting	Reflexion	42.7	-	58.1	28.8
	RL Training	PPO (with critic)	$77.08_{\pm 1.12}$	$76.23_{\pm 1.46}$	$81.4_{\pm 3.1}$	$68.7_{\pm 5.1}$
	RL Training	RLOO	$77.86_{\pm 0.03}$	$73.95_{\pm 0.05}$	$80.3_{\pm 3.2}$	$65.7_{\pm 4.0}$
	RL Training	GRPO	$78.64_{\pm 0.73}$	$76.82_{\pm 1.47}$	$79.3_{\pm 2.8}$	$66.1_{\pm 3.7}$
	RL Training	GiGPO (K=2)	$89.84_{\pm 2.20}$	$82.81_{\pm 5.46}$	$86.23_{\pm 1.43}$	$75.13_{\pm 1.37}$
	RL Training	**HGPO** (K=2)	$\mathbf{91.15}_{\pm 1.19}$	$\mathbf{84.89}_{\pm 4.30}$	$\mathbf{88.93}_{\pm 0.84}$	$\mathbf{76.43}_{\pm 1.47}$
	RL Training	GiGPO (K=4)	$90.88_{\pm 0.90}$	$87.76_{\pm 0.45}$	$87.25_{\pm 1.02}$	$76.18_{\pm 1.25}$
	RL Training	**HGPO** (K=4)	$\mathbf{94.79}_{\pm 0.90}$	$\mathbf{93.22}_{\pm 1.62}$	$\mathbf{87.88}_{\pm 0.41}$	$\mathbf{77.21}_{\pm 0.22}$

Comparing methods. We compare HGPO with many competitive baselines: (1) Closed-source LLMs: GPT-4o (Achiam et al., 2023) and Gemini-2.5-Pro (Team et al., 2023). (2) Prompting agents: ReAct (Yao et al., 2023) and Reflexion (Shinn et al., 2024). (3) RL training methods: PPO (Schulman et al., 2017), RLOO (Kool et al., 2019; Ahmadian et al., 2024), GRPO (Shao et al., 2024), and GiGPO (Feng et al., 2025b). The details are shown in Appendix C.1.

Implementation details. Following prior work (Feng et al., 2025b), we adopt Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct (Yang et al., 2024) as our base models. For fairness, all RL training methods share the same hyperparameter configurations. Specifically, the rollout group size N in group-based RL methods is set to 8. Each LLM agent is prompted to first generate a chain-of-thought (Wei et al., 2022) enclosed within <think> </think> tags, followed by the action enclosed within <action> </action> tags. The weighting coefficient α in Eq. (7) is set to 1. For evaluation, both GiGPO and HGPO are tested with three random seeds, and we report the mean and standard deviation of their performance. Full training setups and hyperparameter details are provided in Appendix C.3. *The anonymous code is attached in the supplemental material.*

5.2 Experimental results

HGPO achieves overall superior performance. As shown in Table 1, all RL training methods significantly outperform prompting-based methods, highlighting the substantial gains in agentic reasoning enabled by RL training. Among the RL-based approaches, our proposed HGPO consistently achieves the best performance across all settings.

HGPO achieves greater performance gains with larger K. We observe that HGPO exhibits a more pronounced performance improvement compared with GiGPO as K increases from 2 to 4. Specifically, GiGPO shows only a modest improvement (from 89.84 to 90.88), whereas HGPO demonstrates a substantial gain (from 91.15 to 94.79). This behavior arises because larger K values

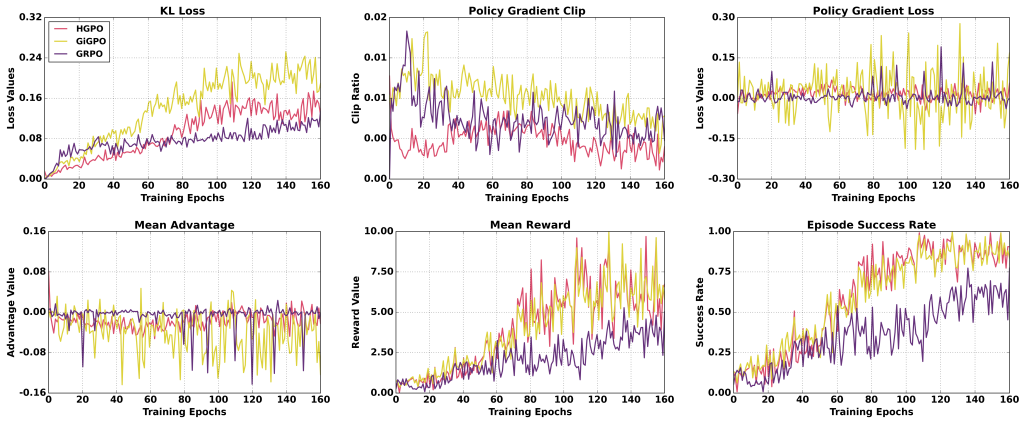

Figure 4: Training dynamics of HGPO (Red), GiGPO (Yellow), and GRPO (Purple) on ALFWorld using Qwen2.5-1.5B-Instruct. The details of these metrics are shown in Appendix D.3.

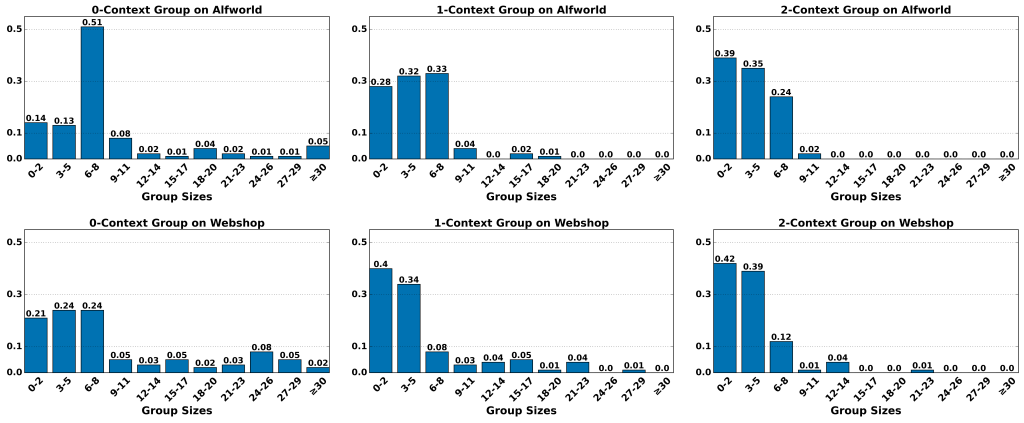

Figure 5: The distributions of hierarchical group sizes ($K = 2$) on ALFWorld and WebShop using Qwen2.5-1.5B-Instruct. The Y-axis denotes the ratio.

exacerbate prompt inconsistency due to the inclusion of more historical contexts, causing step-level advantage estimates in GiGPO to become increasingly biased and limiting performance gains. In contrast, HGPO mitigates prompt inconsistency through hierarchical group advantage computation, emphasizing steps with consistent prompts and thereby reducing estimation bias. This result underscores the effectiveness of our proposed method.

HGPO exhibits better generalization on out-of-distribution ALFWorld tasks. All baseline methods experience significant performance degradation on out-of-distribution tasks in ALFWorld. Notably, HGPO maintains superior performance with less degradation compared to GiGPO. This finding suggests that context inconsistency can severely impair policy optimization and generalization, while HGPO's hierarchical grouping mechanism provides robust and stable advantage estimation, enabling improved generalization to unseen tasks.

5.3 Further Analysis

Training dynamics. Figures 4 and 8 (Appendix D.3) illustrate the training dynamics of GRPO, GiGPO, and HGPO across six metrics: mean advantages, policy gradient loss, KL loss, policy gradient clip fraction, mean reward, and episode success rate. Detailed definitions of these metrics are provided in Appendix C.4. Overall, our method achieves more stable and efficient policy optimization. In particular, for the policy gradient clip fraction, HGPO (red curve) maintains a moderate level, suggesting stable training, whereas GiGPO and GRPO display higher fractions, reflecting in-

stability and constraint. For the KL loss, GRPO's curve is too low, indicating slow learning, while GiGPO's curve is relatively high, reflecting an overly aggressive learning process. By contrast, HGPO achieves a balanced trajectory, demonstrating steady and stable policy learning.

Distribution of hierarchical group sizes. Figure 5 shows the distribution of hierarchical groups in ALFWorld and WebShop using Qwen2.5-1.5B-Instruct at the final epoch (160). The 0/1/2-context groups denote steps sharing 0/1/2 identical historical contexts. We observe that 0-context groups tend to have a higher proportion of large group sizes compared to 1-context and 2-context groups, as they ignore historical context. As K increases, the proportion of large groups decreases while smaller groups become more frequent. This suggests that Oracle steps with identical historical contexts typically form smaller groups, which may increase the variance of advantage estimation. Additional results for $K = 4$ are reported in Appendix D.4.

Step utilization ratio. Table 5 (Appendix D.2) reports the average proportion of steps allocated to different context groups per rollout in ALFWorld and WebShop using Qwen2.5-1.5B-Instruct. The results show that nearly all steps fall into 0-context groups, except for a small fraction corresponding to unique states (appearing only once in a group). As the number of historical contexts increases, the utilization ratio steadily decreases, since fewer steps can be aggregated into higher-level groups. This finding highlights the challenge posed by the scarcity of Oracle steps.

Parameter analysis. Finally, we analyze the effect of the parameter α in Eq. (7), which controls the sharpness of the weight distribution. As shown in Table 4 of Appendix D.1, we observe that setting $\alpha = 0$ results in worse performance, while $\alpha = 1$ yields the best performance. This finding suggests that extensive parameter tuning is not required and that our proposed method HGPO is scalable across different agentic tasks.

5.4 Ablation study

Table 2: Ablation study against HGPO ($K = 2$) on ALFWorld and WebShop.

Ablation	ALFWorld(%)	WebShop(%)
HGPO	**89.58**$_{\pm 0.45}$	**72.66**$_{\pm 1.78}$
W/o HoG-1	13.50$_{\pm 0.58}$	10.13$_{\pm 1.42}$
W/o HoG-2	86.47$_{\pm 1.89}$	57.94$_{\pm 1.02}$
W/o Ada. Weighting	87.23$_{\pm 1.80}$	68.48$_{\pm 0.45}$

In this section, we conduct an ablation study to evaluate the effectiveness of our proposed method. As shown in Table 2, "w/o HoG-1" denotes the setting where hierarchical grouping is removed and only Oracle steps are used to compute advantages for policy optimization. This configuration results in failed policy learning because Oracle steps constitute only a small fraction of all steps, directly undermining policy optimization. "w/o HoG-2" indicates that Oracle advantages are computed for Oracle steps, while step-level advantages for other steps are calculated via Eq. (2). This leads to a significant performance drop, since the small group sizes of Oracle steps cause high variance in advantage estimation, thereby degrading optimization. These two observations collectively validate the necessity of hierarchical grouping. In addition, "w/o Ada. Weighting" refers to replacing adaptive weighting with uniform weights, which is equivalent to fixing α in Eq. (7). Without adaptive weighting, i.e., by aggregating advantages from hierarchical groups using mean weights, performance declines. This is because uniform weighting discards the more accurate advantage information from higher-level hierarchical groups, introducing greater bias in estimation. In contrast, adaptive weighting emphasizes higher-level groups by assigning them larger weights, thus achieving superior performance. Moreover, this mechanism requires no complex hyperparameter tuning and remains scalable across different lengths of historical contexts.

6 Conclusion

In this paper, we propose HGPO, a novel group-based RL algorithm designed to mitigate context inconsistency in long-horizon LLM agent training. HGPO introduces context-aware hierarchical advantage estimation, which enables fine-grained per-step credit assignment while preserving the efficiency and stability of group-based RL. Empirical results on two complex environments, ALFWorld and WebShop, show that HGPO substantially outperforms both prompt-based agents and prior RL approaches. In the future, an interesting direction is to explore alternative strategies for handling context inconsistency, e.g., conditionally controlling trajectories during the rollout stage.

Ethics Statement

This work does not present any potential ethical concerns.

Reproducibility Statement

We provide the anonymous code in the supplementary material to support reproducibility and report all training parameters and settings within the paper.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *ArXiv preprint arXiv:2303.08774*, 2023.

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 Annual Meeting of the Association for Computational Linguistics*, pp. 12248–12267, 2024.

G Brockman. OpenAI Gym. *ArXiv preprint arXiv:1606.01540*, 2016.

Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. Reinforcement learning for long-horizon interactive llm agents. *ArXiv preprint arXiv:2502.01600*, 2025a.

Wentse Chen, Jiayu Chen, Hao Zhu, and Jeff Schneider. Context-lite multi-turn reinforcement learning for LLM agents. In *Proceedings of the International Conference on Machine Learning Workshop*, 2025b. URL https://openreview.net/forum?id=6CE5PLsZdW.

Yongchao Chen, Yueying Liu, Junwei Zhou, Yilun Hao, Jingquan Wang, Yang Zhang, and Chuchu Fan. R1-code-interpreter: Training llms to reason with code via supervised and reinforcement learning. *ArXiv preprint arXiv:2505.21668*, 2025c.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *ArXiv preprint arXiv:2504.19413*, 2025.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *ArXiv preprint arXiv:2501.17161*, 2025.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *ArXiv preprint arXiv:2505.22617*, 2025.

Lang Feng, Weihao Tan, Zhiyi Lyu, Longtao Zheng, Haiyang Xu, Ming Yan, Fei Huang, and Bo An. Towards efficient online tuning of VLM agents via counterfactual soft reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2025a.

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

Hiroki Furuta, Kuang-Huei Lee, Ofir Nachum, Yutaka Matsuo, Aleksandra Faust, Shixiang Shane Gu, and Izzeddin Gur. Multimodal web navigation with instruction-finetuned foundation models. In *Proceedings of the Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=efFmBWioSc.

Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard Mella, Quentin Carbonneaux, Taco Cohen, and Gabriel Synnaeve. Rlef: Grounding code llms in execution feedback with reinforcement learning. *ArXiv preprint arXiv:2410.02089*, 2025.

Boyu Gou, 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 *Proceedings of the International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=kxnoqaisCT.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *ArXiv preprint arXiv:2501.12948*, 2025.

Izzeddin Gur, Hiroki Furuta, Austin V. Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. In *Proceedings of the International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=9JQtrumvg8.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. *ArXiv preprint arXiv:2305.14992*, 2023.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. CogAgent: A visual language model for GUI agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14281–14290, 2024.

Siyuan Hu, Mingyu Ouyang, Difei Gao, and Mike Zheng Shou. The dawn of GUI agent: A preliminary case study with claude 3.5 computer use. *ArXiv preprint arXiv:2411.10323*, 2024.

Arnav Kumar Jain, Gonzalo Gonzalez-Pumariega, Wayne Chen, Alexander M Rush, Wenting Zhao, and Sanjiban Choudhury. Multi-turn code generation through single-step rewards. *ArXiv preprint arXiv:2502.20380*, 2025.

Nan Jiang, Xiaopeng Li, Shiqi Wang, Qiang Zhou, Soneya B Hossain, Baishakhi Ray, Varun Kumar, Xiaofei Ma, and Anoop Deoras. Ledex: Training llms to better self-debug and explain code. *Proceedings of the Advances in Neural Information Processing Systems*, 37:35517–35543, 2024.

Bowen Jin, Hansi Zeng, Zhenrui Yue, 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*, 2025a.

Mingyu Jin, Weidi Luo, Sitao Cheng, Xinyi Wang, Wenyue Hua, Ruixiang Tang, William Yang Wang, and Yongfeng Zhang. Disentangling memory and reasoning ability in large language models. *ArXiv preprint arXiv:2411.13504*, 2024.

Yiyang Jin, Kunzhao Xu, Hang Li, Xueting Han, Yanmin Zhou, Cheng Li, and Jing Bai. Reveal: Self-evolving code agents via iterative generation-verification. *ArXiv preprint arXiv:2506.11442*, 2025b.

Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 reinforce samples, get a baseline for free! In *ICLR 2019 Workshop*, 2019.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. Llms get lost in multi-turn conversation. *ArXiv preprint arXiv:2505.06120*, 2025.

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Erran Li Li, Ruohan Zhang, et al. Embodied agent interface: Benchmarking LLMs for embodied decision making. *Proceedings of the Advances in Neural Information Processing Systems*, 37:100428–100534, 2024.

Hongyu Lin, Yuchen Li, Haoran Luo, Kaichun Yao, Libo Zhang, Mingjie Xing, and Yanjun Wu. Os-r1: Agentic operating system kernel tuning with reinforcement learning, 2025. URL https://arxiv.org/abs/2508.12551.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. DeepSeek-V3 technical report. *ArXiv preprint arXiv:2412.19437*, 2024.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *ArXiv preprint arXiv:2503.20783*, 2025.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Guanjing Xiong, and Hongsheng Li. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. *ArXiv preprint arXiv:2503.21620*, 2025.

Michael Luo, Naman Jain, Jaskirat Singh, Sijun Tan, Ameen Patel, Qingyang Wu, Alpay Ariyak, Colin Cai, Tarun Venkat, Shang Zhu, Ben Athiwaratkun, Manan Roongta, Ce Zhang, Li Erran Li, Raluca Ada Popa, Koushik Sen, and Ion Stoica. Deepswe: Training a state-of-the-art coding agent from scratch by scaling rl, 2025a. Notion Blog.

Run Luo, Lu Wang, Wanwei He, and Xiaobo Xia. Gui-r1: A generalist r1-style vision-language action model for gui agents. *ArXiv preprint arXiv:2504.10458*, 2025b.

Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K. Qiu, and Yuqing Yang. Agent lightning: Train any ai agents with reinforcement learning, 2025c. URL https://arxiv.org/abs/2508.03680.

Ziyang Luo, Zhiqi Shen, Wenzhuo Yang, Zirui Zhao, Prathyusha Jwalapuram, Amrita Saha, Doyen Sahoo, Silvio Savarese, Caiming Xiong, and Junnan Li. Mcp-universe: Benchmarking large language models with real-world model context protocol servers. *ArXiv preprint arXiv:2508.14704*, 2025d.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 1–11, 2015.

OpenAI. Introducing OpenAI o1, 2024. URL https://openai.com/o1.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *ArXiv preprint arXiv:1910.00177*, 2019.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent Q: Advanced reasoning and learning for autonomous ai agents. *ArXiv preprint arXiv:2408.07199*, 2024.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. ToolRL: Reward is all tool learning needs. *ArXiv preprint arXiv:2504.13958*, 2025.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. *ArXiv preprint arXiv:2501.12326*, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Proceedings of the Advances in Neural Information Processing Systems*, 36, 2024.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Android in the wild: A large-scale dataset for android device control. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 36, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Proceedings of the Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

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

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

Maohao Shen, Guangtao Zeng, Zhenting Qi, Zhang-Wei Hong, Zhenfang Chen, Wei Lu, Gregory Wornell, Subhro Das, David Cox, and Chuang Gan. Satori: Reinforcement learning with chain-of-action-thought enhances llm reasoning via autoregressive search, 2025. URL https://arxiv.org/abs/2502.02508.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Proceedings of the Advances in Neural Information Processing Systems*, 36, 2024.

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

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization for llm agents. *ArXiv preprint arXiv:2403.02502*, 2024.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Proceedings of the Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Fei Huang, and Yan Zhang. ZeroSearch: Incentivize the search capability of llms without searching. *ArXiv preprint arXiv:2505.04588*, 2025.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

Weihao Tan, Wentao Zhang, Xinrun Xu, Haochong Xia, Gang Ding, Boyu Li, Bohan Zhou, Junpeng Yue, Jiechuan Jiang, Yewen Li, et al. Cradle: Empowering foundation agents towards general computer control. In *Proceedings of the Advances in Neural Information Processing Systems Workshop*, 2024.

Weihao Tan, Wentao Zhang, Xinrun Xu, Haochong Xia, Ziluo Ding, Boyu Li, Bohan Zhou, Junpeng Yue, Jiechuan Jiang, Yewen Li, et al. Cradle: Empowering foundation agents towards general computer control. In *Proceedings of the International Conference on Machine Learning*, 2025.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: A family of highly capable multimodal models. *ArXiv preprint arXiv:2312.11805*, 2023.

OpenManus-RL Team. Openmanus-rl: Open platform for generalist llm reasoning agents with rl optimization, 2025. URL https://github.com/OpenManus/OpenManus-RL.

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. *Transactions on Machine Learning Research*, 2024a. ISSN 2835-8856.

Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. OTC: Optimal tool calls via reinforcement learning. *ArXiv preprint arXiv:2504.14870*, 2025a.

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-Agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. *Proceedings of the Advances in Neural Information Processing Systems*, 37:2686–2710, 2024b.

Renxi Wang, Rifo Ahmad Genadi, Bilal El Bouardi, Yongxin Wang, Fajri Koto, Zhengzhong Liu, Timothy Baldwin, and Haonan Li. Agentfly: Extensible and scalable reinforcement learning for lm agents. *ArXiv preprint arXiv:2507.14897*, 2025b.

Weixun Wang, Shaopan Xiong, Gengru Chen, Wei Gao, Sheng Guo, Yancheng He, Ju Huang, Jiaheng Liu, Zhendong Li, Xiaoyang Li, et al. Reinforcement learning optimization for large-scale learning: An efficient and user-friendly scaling library. *arXiv preprint arXiv:2506.06122*, 2025c.

Yanlin Wang, Yanli Wang, Daya Guo, Jiachi Chen, Ruikai Zhang, Yuchi Ma, and Zibin Zheng. Rlcoder: Reinforcement learning for repository-level code completion, 2024c. URL https://arxiv.org/abs/2407.19487.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, et al. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *ArXiv preprint arXiv:2504.20073*, 2025d.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I Wang. SWE-RL: Advancing llm reasoning via reinforcement learning on open software evolution. *ArXiv preprint arXiv:2502.18449*, 2025a.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I. Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *ArXiv preprint arXiv:2502.18449*, 2025b.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, et al. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. *ArXiv preprint arXiv:2505.16421*, 2025c.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Proceedings of the Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

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.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. *Proceedings of the Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Chengyue Yu, Siyuan Lu, Chenyi Zhuang, Dong Wang, Qintong Wu, Zongyue Li, Runsheng Gan, Chunfeng Wang, Siqi Hou, Gaochi Huang, et al. Aworld: Orchestrating the training recipe for agentic ai. *ArXiv preprint arXiv:2508.20404*, 2025a.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, et al. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *ArXiv preprint arXiv:2507.02259*, 2025b.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *ArXiv preprint arXiv:2503.14476*, 2025c.

Chaoyun Zhang, Liqun Li, Shilin He, Xu Zhang, Bo Qiao, Si Qin, Minghua Ma, Yu Kang, Qingwei Lin, Saravan Rajmohan, et al. UFO: A UI-focused agent for windows OS interaction. *ArXiv preprint arXiv:2402.07939*, 2024a.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, Yifan Zhou, Yang Chen, Chen Zhang, Yutao Fan, Zihu Wang, Songtao Huang, Yue Liao, Hongru Wang, Mengyue Yang, Heng Ji, Michael Littman, Jun Wang, Shuicheng Yan, Philip Torr, and Lei Bai. The landscape of agentic reinforcement learning for llms: A survey, 2025. URL https://arxiv.org/abs/2509.02547.

Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. CodeAgent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 13643–13658, 2024b.

Zhuosheng Zhang and Aston Zhang. You only look at screens: Multimodal chain-of-action agents. In *Proceedings of the Findings of the Association for Computational Linguistics*, pp. 3132–3149, 2024.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. GPT-4V (ision) is a generalist web agent, if grounded. *ArXiv preprint arXiv:2401.01614*, 2024.

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

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. In *Proceedings of the International Conference on Machine Learning*, pp. 62138–62160, 2024a.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn rl. In *Proceedings of the International Conference on Machine Learning*, pp. 62178–62209. PMLR, 2024b.

Yifei Zhou, Song Jiang, Yuandong Tian, Jason Weston, Sergey Levine, Sainbayar Sukhbaatar, and Xian Li. Sweet-rl: Training multi-turn llm agents on collaborative reasoning tasks, 2025a. URL https://arxiv.org/abs/2503.15478.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. *ArXiv preprint arXiv:2506.15841*, 2025b.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *ArXiv preprint arXiv:1909.08593*, 2019.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of the Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

A ALGORITHM

Algorithm 1 The pseudo-code of HGPO

Require: Initial policy $\pi_{\theta_{\text{old}}}$, task distribution $p(X)$, discount factor γ, weighting ω, clipping parameter ϵ, KL penalty β, group size N, the length of historical context K, parameter α
for each training iteration **do**
  Update the old policy model: $\theta_{\text{old}} \leftarrow \theta$
  // Multi-step rollout phase
  Sample task $x \sim p(X)$ and initialize N identical environments
  for $t = 1$ to T **do**
    Sample actions $\left\{\boldsymbol{a}_t^{(i)} \sim \pi_{\theta_{\text{old}}}(\cdot \mid \boldsymbol{s}_t^{(i)}, x)\right\}_{i=1}^{N}$
    Execute actions, observe rewards $\{r_t^{(i)}\}_{i=1}^{N}$ and next state $\{\boldsymbol{s}_{t+1}^{(i)}\}_{i=1}^{N}$
  end for
  // Grouping phase
  Context-aware hierarchical grouping by Eq. (5)
  // Advantage computation phase
  Compute multiple advantages within each group by Eq. (7)
  // Policy update phase
  Update policy θ by maximizing objective $\mathcal{J}_{\text{HGPO}}(\theta)$
end for

B MORE DETAILS AND PROOF FOR THEOREM

Table 3: Overall comparison of three different advantage estimators.

Type	Advantage estimation	Granularity	Bias	Variance
Trajectory-level	$A^T(\tau_i) = \left(R(\tau_i) - 1/\lvert G_\tau \rvert \sum_{j \in G_\tau} R(\tau_i)\right) / \sigma_{G_\tau}$	Coarse-grained	B_T	V_T
Step-level	$A^S(\tilde{\boldsymbol{s}}_i) = \left(R(\tilde{\boldsymbol{s}}_i) - 1/\lvert G_{\tilde{\boldsymbol{s}}_i} \rvert \sum_{j \in G_{\tilde{\boldsymbol{s}}_i}} R(\tilde{\boldsymbol{s}}_j)\right) / \sigma_{G_{\tilde{\boldsymbol{s}}_i}}$	Fine-grained	b_0	v_0
Hierarchy-of-Groups	$A^H(\boldsymbol{s}_t^{(i)}) = \sum_{k=0}^{K} \boldsymbol{w}_k A_k^H(\boldsymbol{s}_t^{(i)})$	*Fine-grained*	$\sum_{k=0}^{K} w_k b_k \downarrow$	$\sum_{k=0}^{K} w_k^2 v_k \downarrow$

Here, we provide more details of Proposition 4.1. Let A_k^H denote the advantage estimator for the k-th hierarchical group. Let b_k and v_k denote the bias and variance of the estimated advantage A_k^H within the k-th group G_k^H. The definition of bias is $b_k = \text{Bias}[A] = A - A^*$ where A^* is the unknown true advantage. We make the following conditions:

(1) *Bias decreases monotonically* with k:

$$B_T \geq b_0 \geq b_1 \geq b_2 \geq \cdots > b_K \geq 0, \quad b_k = \text{Bias}[A_k^H],$$

(2) *Variance increases monotonically and independently* with k:

$$v_0 \leq v_1 \leq v_2 \leq \cdots \leq v_K \leq V_T, \quad v_k = \text{Var}[A_k^H],$$

where B_T and V_T denote the bias and variance of trajectory-level advantage estimation, and b_0 and v_0 represent those of step-level estimation. We now justify the assumptions. First, the number of trajectories in a group is generally smaller (set to 8 in our experiments) than the step-level group size, which leads to higher bias and variance in trajectory-level estimation. Second, as K increases, the group size of G_k^H decreases, which can result in higher variance.

Bias and variance of HGPO. Recall the advantage estimation in Eq. (5) and Eq. (7), we first analyze the bias.

$$\text{Bias}[A^H] = \text{Bias}\left[\sum\nolimits_{k=0}^{K} w_k A_k^H\right] = \sum\nolimits_{k=0}^{K} w_k b_k.$$

Since $b_0 \geq b_1 \geq \cdots \geq b_K$ and $\sum_k w_k = 1$, it follows that

$$b_K = \sum\nolimits_{k=0}^{K} w_k b_K \leq \text{Bias}[A^H] \leq \sum\nolimits_{k=0}^{K} w_k b_0 = b_0 \leq B_T$$

Hence, HGPO trades off the bias between the bias of the step-level advantage estimation and the bias of the advantage estimation in Oracle (K) groups. Correspondingly, consider that $\text{Cov}(A_k^H, A_{k'}^H) = 0 \quad (k \neq k')$ due to the independent condition, the variance is

$$\text{Var}[A^H] = \text{Var}\left[\sum_{k=0}^{K} w_k A_k^H\right] = \sum_{k=0}^{K} w_k^2 \text{Var}[A_k^H] = \sum_{k=0}^{K} w_k^2 v_k.$$

Since $v_0 < v_1 < \cdots < v_K$ and $\sum_k w_k^2 < 1$, it follows that

$$\begin{aligned}
\text{Var}[A^H] &= \sum_{k=0}^{K} w_k^2 v_k = \sum_{k=0}^{K} \left(\frac{(k+1)^\alpha}{\sum_k (k+1)^\alpha}\right)^2 v_k \\
&= \frac{\sum_{k=0}^{K} (k+1)^{2\alpha}}{\left(\sum_k (k+1)^\alpha\right)^2} v_k \\
&\leq \frac{K(K+1)^{2\alpha}}{K^2} v_K = \frac{(K+1)^{2\alpha}}{K} v_K
\end{aligned}$$

$$\begin{aligned}
\text{Var}[A^H] &= \sum_{k=0}^{K} w_k^2 v_k = \sum_{k=0}^{K} \left(\frac{(k+1)^\alpha}{\sum_k (k+1)^\alpha}\right)^2 v_k \\
&= \frac{\sum_{k=0}^{K} (k+1)^{2\alpha}}{\left(\sum_k (k+1)^\alpha\right)^2} v_k \\
&\geq \frac{K^2}{K(K+1)^{2\alpha}} v_0 = \frac{1}{K(K+1)^{2\alpha}} v_0
\end{aligned}$$

HGPO achieves a trade-off of bias and variance in advantage estimation. First, the bias of the HGPO advantage estimator is lower than that of both trajectory- and step-level estimators, and trades off against the Oracle estimator. Second, its variance trades off against the step-level and Oracle estimators, depending on the number of hierarchical groups K and the weight parameter α. In summary, HGPO provides a principled framework for advantage estimation that systematically leverages historical context while maintaining statistical efficiency through weighted aggregation.

C Experiment Details

C.1 Comparing methods

- *GPT-4o:* A closed-source, large-scale LLM used as a baseline for multi-turn agentic tasks (Achiam et al., 2023).
- *Gemini-2.5-Pro:* Another closed-source LLM, comparable in scale and capability to GPT-4o (Team et al., 2023).
- *ReAct:* A prompting-based agent that integrates reasoning and acting in an interleaved chain-of-thought framework (Yao et al., 2023).
- *Reflexion:* A prompting agent that incorporates self-reflection and iterative improvement over generated outputs (Shinn et al., 2024).
- *PPO:* Proximal Policy Optimization, a classic RL algorithm for policy learning (Schulman et al., 2017).
- *RLOO:* Reinforcement Learning with Offline Observations, a group-based RL approach that estimates advantages without value networks (Kool et al., 2019; Ahmadian et al., 2024).
- *GRPO:* Group-based RL with trajectory-level advantage estimation, designed to scale RL to multi-step tasks (Shao et al., 2024).
- *GiGPO:* Grouped Incremental GPO, a prior hierarchical RL method that performs group-wise advantage estimation for LLM-based agents (Feng et al., 2025b).

C.2 Environment details

In each episode, the agent receives a text goal and must accomplish it through multi-turn interaction with the environment. It includes 4,639 task instances across six categories of common household activities: Pick & Place (Pick), Examine in Light (Look), Clean & Place (Clean), Heat & Place (Heat), Cool & Place (Cool), and Pick Two & Place (Pick2). *WebShop* is a complex, web-based interactive environment designed to test the LLM agents in realistic online shopping scenarios. To complete the task, the agent must interact with a simulated HTML-based shopping website to search for, navigate to, and ultimately purchase a suitable item. It contains over 1.1 million products and 12k user instructions, providing a rich and diverse action space.

C.3 Details of Training

Generally, we use the same training settings in (Feng et al., 2025b) for fair comparison.

Hyperparameters for ALFWorld. All methods are configured with identical hyperparameters: the maximum prompt length is 2048 tokens, and the maximum response length is 512 tokens. Each episode allows up to 50 environment steps. The learning rate is set to 1e-6 for the actor and 1e-5 for the critic (used only in PPO). We adopt a rule-based reward, assigning a reward of 10 for success and 0 for failure. To handle invalid actions generated by the agent, we apply a reward penalty of -0.1. For all group-based RL methods, we use a group size of 8 and sample 16 different groups per rollout, resulting in a total of $16 \times 8 = 128$ environments. In contrast, PPO uses 128 separate environments for rollouts. The rollout temperature is set to 1.0, while the validation temperature is set to 0.4. The mini-batch size is 256, and the KL-divergence loss coefficient is set to 0.01. The discount factor γ is set to 0.95.

Prompt Template for ALFWorld

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 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.

Figure 6: The prompt template of ALFWorld agents.

Prompt Template for WebShop

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 for 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.

Figure 7: The prompt template used for WebShop agents.

Hyperparameters for WebShop. All methods are configured with identical hyperparameters: the maximum prompt length is 4096 tokens, and the maximum response length is 512 tokens. Each episode is limited to 15 environment steps. The learning rate is 1e-6 for the actor and 1e-5 for the critic (used only in PPO). We adopt a rule-based reward, assigning a reward of 10 for success and

0 for failure. Invalid actions are penalized with a reward of -0.1. As with ALFWorld, all group-based RL methods use a group size of 8 and sample 16 groups per rollout, totaling $16 \times 8 = 128$ environments. PPO, on the other hand, uses 128 distinct environments for rollouts. The rollout temperature is set to 1.0, while the validation temperature is set to 0.4. The mini-batch size is 64, and the KL-divergence loss coefficient is set to 0.01. The discount factor γ is set to 0.95.

Computing Details. Experiments using Qwen2.5-1.5B-Instruct are conducted on two NVIDIA H100 GPUs, while those using Qwen2.5-7B-Instruct are trained on four NVIDIA H100 GPUs. Each experiment is trained for a total of 160 training iterations. In particular, when computing the weights in Eq. 7, we omit groups with zero advantage to avoid relying on unavailable estimates. The validation data sizes are 128 and 256 for ALFWorld and WebShop, respectively.

C.4 Training metrics

- *Mean Advantages:* This metric shows how much better the chosen actions are compared to the average action. A positive and stable value means the agent usually selects better actions, while large fluctuations suggest unstable training.
- *Policy Gradient Loss:* This loss is the main signal for updating the policy. A smooth and gradually decreasing value indicates stable learning. If the loss becomes too large or changes sharply, it means the updates are too aggressive and may harm training stability.
- *KL Divergence:* KL loss measures how different the new policy is from the old one. It acts as a constraint to prevent the policy from changing too quickly. A moderate KL value means the agent is learning steadily, while a very high value can cause divergence and a very low value may slow down learning.
- *Policy Gradient Clip Fraction:* This metric shows the proportion of gradients that are clipped during optimization. Gradient clipping prevents extreme updates. A moderate fraction suggests stable training, but if the fraction is too high, it means many updates are unstable and are being restricted.
- *Mean Reward:* The mean reward reflects the average return the agent receives per episode. It is a direct measure of progress: higher rewards indicate better performance. If the mean reward increases smoothly, it shows effective learning, while sudden drops suggest instability.
- *Episode Success Rate:* This metric measures the percentage of episodes in which the agent completes the task. It is an intuitive indicator of how well the agent achieves its goal. A rising success rate shows that the agent is improving and that training is effective.

C.5 Prompts

The prompts we use for LLM agents are presented in Figure 6 and Figure 7. These prompt templates are constructed using Python-style string formatting, where placeholders enclosed in curly braces ({}) represent semantic slots. These placeholders, such as {task_description}, {step_count}, and {current_observation}, are dynamically populated at runtime via Python's `.format()` function. To enrich the agent's context, we use historical information and set the history length to 2.

The <think>...</think> block instructs the agent to perform step-by-step reasoning, thereby promoting chain-of-thought style deliberation explicitly. The <action>...</action> block is used to indicate the final action decision clearly.

D More experimental results

D.1 Parameter analysis

We report the experimental results of parameter analysis as shown in Table 4.

D.2 Step utilization ratio

We show the step utilization ratio on ALFWorld and WebShop (using Qwen2.5-1.5B-Instruct at epoch 160) as shown in Table 5.

Table 4: Parameter analysis of α on ALFWorld and WebShop ($K = 2$) using Qwen2.5-1.5B-Instruct.

Parameter	$\alpha = 0$	$\alpha = 1$	$\alpha = 2$
ALFWolrd	$87.23_{\pm 1.80}$	$89.58_{\pm 0.45}$	$84.76_{\pm 1.17}$
WebShop	$68.48_{\pm 0.45}$	$72.66_{\pm 1.78}$	$72.65_{\pm 1.77}$

Table 5: Step utilization ratio on ALFWorld and WebShop (using Qwen2.5-1.5B-Instruct at epoch 160).

Dataset	0-Context	1-Context	2-Context	3-Context	4-Context
ALFWolrd ($K = 2$)	0.97	0.75	0.52	-	-
ALFWolrd ($K = 4$)	0.98	0.77	0.54	0.34	0.19
WebShop ($K = 2$)	0.92	0.64	0.44	-	-
WebShop ($K = 4$)	0.90	0.59	0.4	0.21	0.09

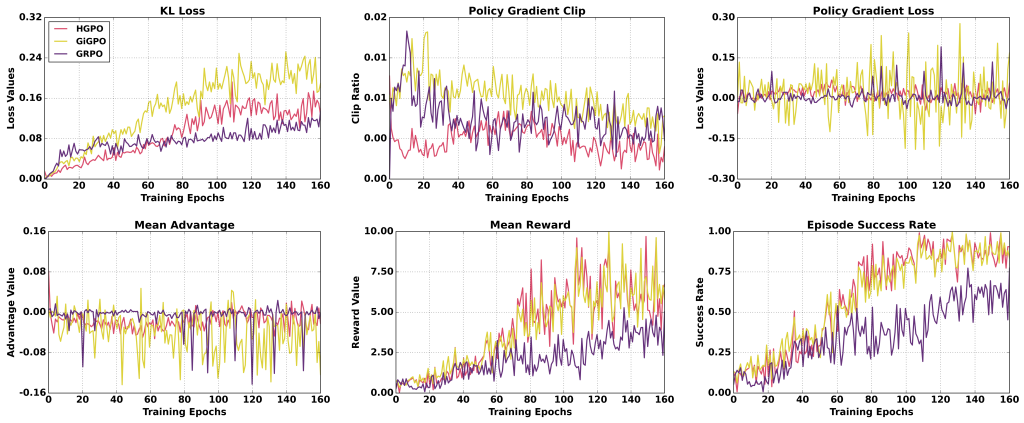

Figure 8: Training dynamics of HGPO (Red), GiGPO (Yellow), and GRPO (Blue) on WebShop using Qwen2.5-1.5B-Instruct. Best viewed in color.

D.3 Training dynamics

We show training dynamics of HGPO (Red), GiGPO (Yellow), and GRPO (Blue) on WebShop using Qwen2.5-1.5B-Instruct as shown in Figure 8.

D.4 The distribution

We report the distributions of hierarchical group sizes (K = 4) on ALFWorld and WebShop using Qwen2.5-1.5B-Instruct as shown in Table 9.

E Use of LLMs

We used LLMs exclusively as writing assistants to refine language. In particular, their use was restricted to grammar correction, style improvement, and phrasing adjustments for clarity and conciseness.

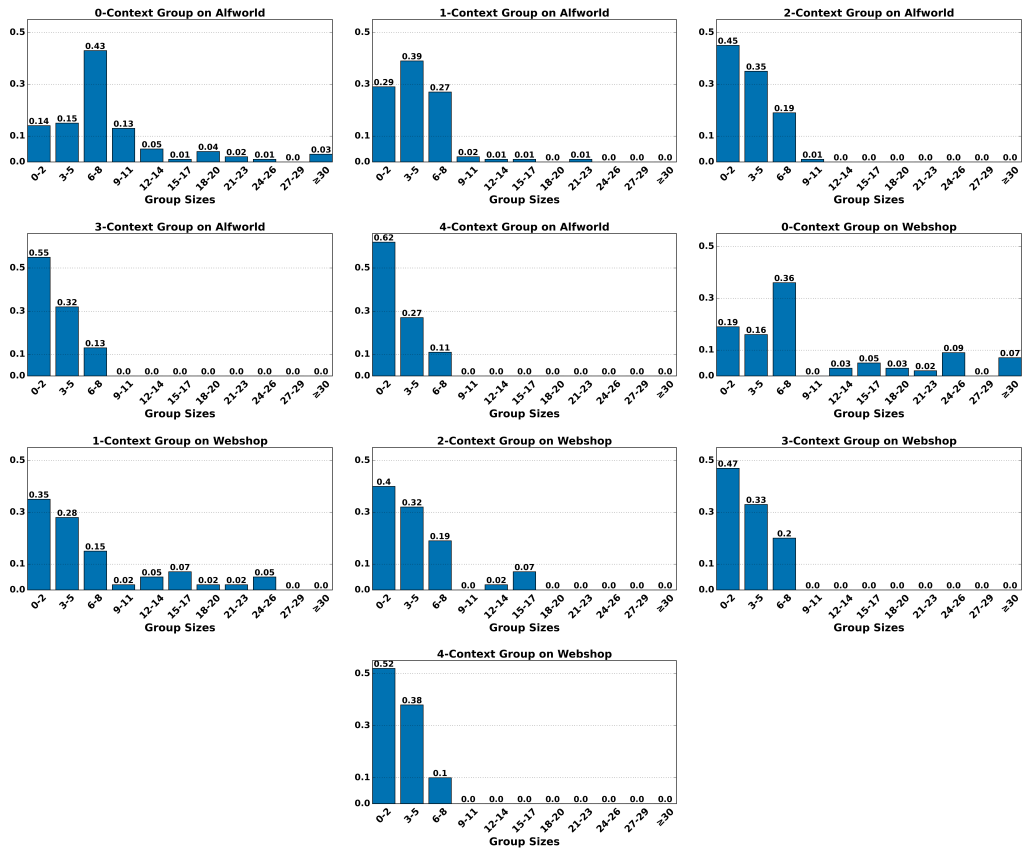

Figure 9: The distributions of hierarchical group sizes ($K = 4$) on ALFWorld and WebShop using Qwen2.5-1.5B-Instruct.