DEPTH BEFORE BREADTH: REVERSE-TO-FORWARD CURRICULUM WITH ADVANTAGE DECOMPOSITION FOR LLM AGENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) has become a common approach for training large language model (LLM) agents on long-horizon tasks. However, learning from sparse terminal rewards makes credit assignment and exploration difficult. We approach these challenges through a *policy-induced capability region*, the set of states from which the current policy can reliably complete a task. Near its boundary, an agent may need only a few actions to reach a more solvable state, even when the goal remains distant. This reduces long-horizon learning to two coupled subproblems of reaching that state and completing the task from there. We address them with paired advantages and a progressive reverse-to-forward curriculum. The advantages provide separate signals for reaching more solvable states and completing the task from them, while the curriculum expands the capability region backward along successful reference trajectories and then into off-reference states through on-policy exploration. On ALFWorld and WebShop, our method outperforms competitive agentic RL methods at both model scales, achieving success rates of $90.1\%/96.1\%$ and $80.2\%/82.6\%$ with Qwen2.5-1.5B/7B-Instruct, respectively.

1 INTRODUCTION

Large language models (LLMs) (Achiam et al., 2023; Grattafiori et al., 2024; Liu et al., 2024) are increasingly deployed as agents that interact with external environments on behalf of users (Yao et al., 2023; Wang et al., 2023). Moving beyond traditional single-turn generation, these agents operate through a stateful, sequential decision process where each action continuously adapts to evolving observations. Reinforcement learning (RL) has emerged as the dominant paradigm for training such interactive agents(Zhou et al., 2024; Qi et al., 2025; Wang et al., 2025; Xi et al., 2025), with group-based policy gradient methods, particularly Group Relative Policy Optimization (GRPO)(Shao et al., 2024), fast becoming the prevailing standard (Feng et al., 2025; Zheng et al., 2025). However, scaling these algorithms to long-horizon agent tasks remains bottlenecked by reward sparsity (Andrychowicz et al., 2017; Jin et al., 2025; Yu et al., 2026). All failed rollouts produce no learning signal at any step, and even successful ones deliver only a single terminal reward that has to be attributed across the entire trajectory. This sparse feedback makes credit assignment particularly challenging as LLM agents navigate vast action spaces and diverse tasks through partially observable environments with textual observations.

To address reward sparsity, a standard strategy is to reshape the training distribution through curriculum learning, which exposes the policy to simpler tasks before introducing progressively harder ones (Bengio et al., 2009; Narvekar et al., 2020). Reverse curriculum strategies operationalize this principle by initializing the agent near the goal and incrementally pushing the starting state farther back (Florensa et al., 2017). Recent LLM agent work adopts a reference-anchored variant (Zhang & Yang, 2026). Instead of sampling start states by random walks, the start regresses backward along a successful reference trajectory, so the policy first masters local execution near the goal and is only later asked to complete the task from the original start (Resnick et al., 2018; Tao et al., 2024).

However, reverse curriculum leaves a structural credit assignment problem unresolved. Although initializing closer to the goal allows terminal feedback to be attributed more precisely to individual

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

actions, moving the starting point farther back stretches the trajectory, forcing the same coarse feedback to be distributed over a longer sequence. Consequently, isolating early actions that genuinely drove the outcome remains difficult (Arjona-Medina et al., 2019; Pignatelli et al., 2023). A separate limitation is that reverse curriculum starts are anchored to reference trajectories, limiting exposure to the off-reference states encountered during the policy's own exploration (Ross et al., 2011).

Our key insight is that moving the training start backward also expands the policy's *capability region* toward the original start. As Figure 1 shows, later checkpoints in reverse curriculum achieve high success from states progressively farther from the goal along the reference trajectory. A newly introduced starting state near this boundary can still be far from the goal, yet only a few actions away from a state within the policy's existing capability region. We exploit this structure by splitting credit assignment into two subproblems: **(i)** *how to enter* the capability region from a state just outside it, a short-horizon problem; and **(ii)** *how to complete* the task from the state reached inside it, a longer-horizon problem that earlier training has already partially addressed. The same decomposition can then be applied to states near the capability boundary encountered in the policy's own rollouts, extending training beyond the reference trajectory.

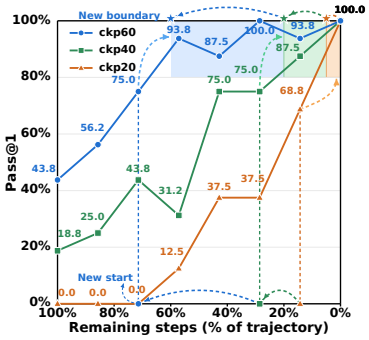


Figure 1: The three curves show Pass@1 at checkpoints 20, 40, 60 in reverse curriculum training , as the training start moves backward along a successful reference trajectory from near the goal toward the original task start. Defining the capability region on this trajectory as starting positions with Pass@1 above 80%, we observe that this region also expands toward the original start. By ckp60, its new boundary reaches approximately 60% remaining steps.

We operationalize this decomposition through a Paired Advantage (PA) update. For each training group, we sample prefixes from a starting state near the capability boundary, group them by their endpoints, and sample multiple suffixes from each distinct endpoint. The left advantage compares estimated completion rates across prefix endpoints, crediting prefixes that reach states the policy can solve more reliably and thereby guiding it into the capability region. The right advantage compares suffix outcomes from the same endpoint to identify which continuations lead to task completion. The suffix outcomes thus provide both the endpoint estimates used for prefix credit and the comparisons used for suffix credit. We apply this update in two stages. The reverse stage establishes the capability region along successful reference trajectories, moving training starts backward from near the goal until the original starts become reliably solvable for most tasks. The forward stage then broadens the region beyond those trajectories by training on states reached through prefixes sampled by the policy from the original starts. Our contributions are threefold:

- We introduce a policy-induced capability region view of reverse curriculum training. It reveals a short horizon problem of entering the region within a long horizon task.
- We propose a paired advantage method with a reverse-to-forward curriculum. Using terminal rewards alone, it compares completion rates across prefix endpoints to guide entry into the capability region and compares suffix outcomes from the same endpoint to improve task completion.
- Comprehensive experiments on ALFWorld(Shridhar et al., 2021) and WebShop (Yao et al., 2022) show that our method outperforms strong baselines and achieves a larger capability region under matched environment interaction budgets.

2 Preliminaries

Problem setup. We consider a distribution of tasks $x \sim \mathcal{D}$. For each task x, an LLM agent interacts with an environment over multiple steps. At step t, the agent receives observation o_t and emits a textual action a_t; the next observation o_{t+1} then arrives in response. A trajectory $\tau = \{(o_t, a_t)\}_{t=0}^{T-1}$ together with its terminal reward $R(\tau) \in \{0, 1\}$ captures one full interaction, where $R(\tau) = 1$ if the task is completed successfully and 0 otherwise. The agent's behavior is governed by an LLM policy π_θ that conditions the interaction. We define a *decision context* $c_t = (o_{\leq t}, a_{<t}, x)$

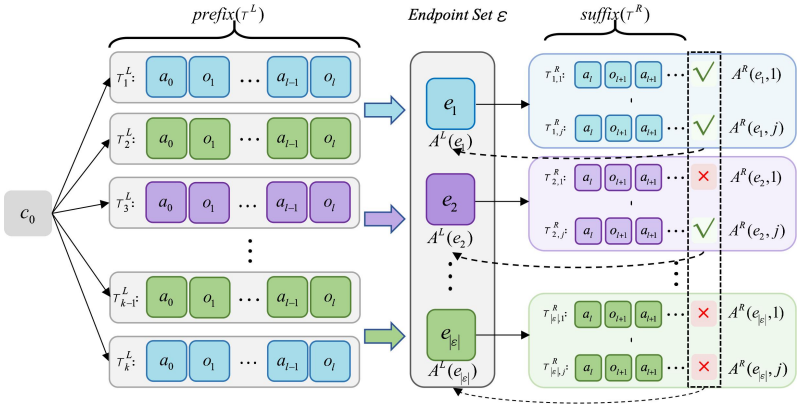


Figure 2: **Overview.** Starting from a context c_0, the policy samples n prefixes τ_k^{L} and groups those that reach the same endpoint context e, yielding the set of distinct endpoints $\mathcal{E}$. From each $e \in \mathcal{E}$, it samples N_{R} suffixes $\tau_{e,j}^{\mathrm{R}}$ and observes their terminal rewards. The left advantage compares estimated completion rates across endpoints to credit prefixes for where they lead, while the right advantage compares suffix outcomes from the same endpoint to credit the actions taken after reaching it.

at step t, yielding an action according to $\pi_\theta(a_t \mid c_t)$. The learning objective is:

$$J(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, \tau\sim\pi_\theta(\cdot|x)}\left[R(\tau)\right]. \tag{1}$$

Group-based policy optimization. Group-based methods estimate relative advantages by comparing several rollouts of the same task. Given a task x, a policy $\pi_{\theta_{\text{old}}}$ samples a group of n trajectories $\{\tau^{(k)}\}_{k=1}^n \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$, with terminal rewards $\{R^{(k)}\}_{k=1}^n$. A GRPO-style update assigns each trajectory an advantage based on its reward relative to the group:

$$A^{(k)} = \frac{R^{(k)} - \text{mean}(R)}{\text{std}(R) + \epsilon}, \tag{2}$$

the policy update weights each trajectory's log-probabilities by $A^{(k)}$. The same trajectory-level signal is broadcast to every action inside the rollout.

Policy-induced capability region. Let $p_\pi(c)$ denote the probability that policy π completes the task when acting from decision context c. Given a reliability threshold $\eta \in (0, 1)$, we define the policy-induced capability region by:

$$\mathbb{C}_\eta(\pi) = \{c : p_\pi(c) \geq \eta\}, \tag{3}$$

this region contains the decision contexts from which the current policy completes the task with probability at least η. The threshold layer around $p_\pi(c) = \eta$ is the *capability boundary*. For a tolerance $\delta > 0$, the corresponding near-boundary set is:

$$\mathbb{N}_{\eta,\delta}(\pi) = \{c : |p_\pi(c) - \eta| \leq \delta\}, \tag{4}$$

where $0 < \delta < \min\{\eta, 1 - \eta\}$. Contexts in $\mathbb{N}_{\eta,\delta}(\pi)$ sit close to the current reliability threshold and are informative for expanding the capability region. The region and its boundary are induced by the current policy and need no geometric structure on the context space.

3 Methodology

This section develops a method that expands the capability region $\mathbb{C}_\eta(\pi)$ using terminal feedback alone. The method draws training contexts from near the capability boundary $\mathbb{N}_{\eta,\delta}(\pi)$ and separates the learning signal into two parts. One part pushes contexts from outside the region into it, the other sharpens completion from inside the region, as illustrated in Figure 2. We implement this expansion with a two-stage curriculum: the reverse stage uses successful reference trajectories to initialize the solvable region and expand it backward along trajectory depth, after which the forward stage broadens the capability region through natural rollouts.

3.1 PAIRED ADVANTAGE DECOMPOSITION

For a single task, we sample a rollout group of n trajectories from starting contexts near the capability boundary $\mathbb{N}_{\eta,\delta}(\pi)$. Let $\pi_{\theta_{\text{old}}}$ denote the behavior policy used to collect the rollouts. Each rollout is divided into a prefix and a suffix. The prefix determines which decision context is reached, while the suffix determines whether the task can be completed from that context.

For the k-th rollout, the prefix is $\tau_k^{\text{L}} = (c_{0,k},\, a_{0,k},\, o_{1,k},\, a_{1,k},\, \ldots,\, a_{\ell-1,k},\, o_{\ell,k})$, for $k = 1, \ldots, n$, where $c_{0,k} \in \mathbb{N}_{\eta,\delta}(\pi)$ is the starting decision context and $o_{\ell,k}$ is the prefix endpoint observation. The prefix is used to determine whether the sampled action sequence gets a context c_ℓ with a high probability of task completion.

The sampled prefixes are deduplicated by their endpoint context c_ℓ. Two prefixes land in the same class if their endpoint contexts coincide, and $\mathcal{E}$ denotes the resulting set of unique endpoint contexts. Let $\mathcal{I}(e) = \{\, k : c_{\ell,k} = e \,\}$ be the indices of prefixes with endpoint $e \in \mathcal{E}$, and let $m(e) = |\mathcal{I}(e)|$ be the number of such prefixes. Although each LLM turn contains both a `<think>` segment and an executable `<action>` segment, we identify duplicate endpoints using only the sequence of `<action>` outputs.

For each endpoint context $e \in \mathcal{E}$, the policy samples $N_{\text{R}} \geq 2$ suffixes from e, where N_{R} is the same for all endpoint contexts. The j-th suffix is denoted by $\tau_j^{\text{R}} = \big(a_{j,0}, o_{j,1}, \ldots, a_{j,L_j-1}, o_{j,L_j}\big)$, and let $R_{e,j} \in \{0, 1\}$ denote its outcome. The empirical task completion value at e is estimated by:

$$\widehat{V}(e) = \frac{1}{N_{\text{R}}} \sum_{j=1}^{N_{\text{R}}} R_{e,j}. \tag{5}$$

We compare endpoint contexts using a prefix-weighted baseline and assign the left advantage:

$$A^{\text{L}}(e) = \widehat{V}(e) - \frac{1}{n} \sum_{e' \in \mathcal{E}} m(e')\, \widehat{V}(e'). \tag{6}$$

Thus, $A^{\text{L}}(e)$ measures whether prefixes reaching endpoint e have a higher empirical task-completion value than the prefix-weighted average over all endpoints. The suffixes associated with the same endpoint context are compared separately. For suffix j sampled from e, define the right advantage by:

$$A^{\text{R}}(e, j) = R_{e,j} - \frac{1}{N_{\text{R}} - 1} \sum_{j' \neq j} R_{e,j'}. \tag{7}$$

This signal evaluates a suffix relative to the other suffixes sampled from the same endpoint context. When all suffixes associated with e have the same outcome, the right signal vanishes, while e can still contribute through the left signal in Eq. 6.

The paired decomposition therefore separates two aspects of capability expansion. The left signal evaluates which prefix executions reach contexts with higher completion probabilities, whereas the right evaluates how the task should be completed after a particular context has been reached.

To realize this decomposition in the policy update, each prefix is completed jointly with a suffix sampled from its endpoint. For prefix τ_k^{L}, let $e_k = c_{\ell,k}$ denote its endpoint context and independently sample j_k uniformly from $\{1, \ldots, N_R\}$. The selected suffix $\tau_k^{\text{R}} = \tau_{e_k,j_k}^{\text{R}}$ forms a completed trajectory $\tau_k = \tau_k^{\text{L}} \oplus \tau_k^{\text{R}}$, where $\oplus$ denotes concatenation. With the estimated advantages held fixed, the policy optimization objective is:

$$\begin{aligned} \mathcal{J}(\theta) = \mathbb{E}\Bigg[\frac{1}{n} \sum_{k=1}^{n} \Bigg(& \frac{1}{\ell} \sum_{t=0}^{\ell-1} \min\Big[\rho_{t,k}(\theta) A^{\text{L}}(e_k), \text{clip}\big(\rho_{t,k}(\theta), 1-\epsilon, 1+\epsilon\big) A^{\text{L}}(e_k) \Big] \\ & + \frac{1}{T_k - \ell} \sum_{t=\ell}^{T_k-1} \min\Big[\rho_{t,k}(\theta) A^{\text{R}}(e_k, j_k), \text{clip}\big(\rho_{t,k}(\theta), 1-\epsilon, 1+\epsilon\big) A^{\text{R}}(e_k, j_k) \Big] \Bigg) \Bigg]. \end{aligned} \tag{8}$$

where $\rho_{t,k}(\theta) = \dfrac{\pi_\theta(a_{t,k} \mid c_{t,k})}{\pi_{\theta_{\text{old}}}(a_{t,k} \mid c_{t,k})}$ is the importance sampling ratio, ϵ is the clipping parameter, and T_k is the number of actions in the completed trajectory τ_k. The expectation is over task sampling,

rollout collection under $\pi_{\theta_{\text{old}}}$, and suffix selection. The two terms apply clipped relative advantages to the prefix and suffix actions, respectively. The two-stage curriculum uses the same objective and differs in their starting contexts and prefix lengths used to construct these trajectories.

3.2 TWO-STAGE CURRICULUM: REVERSE THEN FORWARD

The paired advantage update samples starting contexts from the near-boundary set $\mathbb{N}_{\eta,\delta}(\pi)$. The curriculum constructs these contexts in two stages. The reverse stage expands $\mathbb{C}_\eta(\pi)$ backward along successful reference trajectories, while the forward stage extends it to contexts encountered in the policy's own rollouts.

Reverse stage: expanding backward along references. The reverse stage operates on training tasks drawn from $\mathcal{D}$. For each task, we assume a successful reference trajectory $\tau^* = \{(o_t^*, a_t^*)\}_{t=0}^{T-1}$. The reverse stage starts from a context $c_{T-1}^* = (\, o_{\le T-1}^*,\, a_{<T-1}^*, x)$ near the end of τ^*, where the remaining effective horizon is short, and progressively moves toward the original task start.

Let $r_0 = T-1$ denote the initial position of the reverse curriculum. At stage i, we select the current near-boundary context $c_i = c_{r_i}^*$, where $r_i = r_0 - id$ and d is the fixed backward stride along the reference trajectory. We sample prefixes of length $\ell_i = d+2$ from c_i, group them by their endpoint contexts, and apply the paired advantage decomposition to the resulting groups.

The update continues until the estimated success probability at c_i reaches the reliability threshold $p_\pi(c_i) \ge \eta$, after which c_i is treated as belonging to $\mathbb{C}_\eta(\pi)$. The curriculum then proceeds to the earlier reference context indexed by r_{i+1}. The reverse stage terminates when the original task start $c_0^* = (o_0^*, x)$ is included in $\mathbb{C}_\eta(\pi)$ for most training tasks. This produces a policy that can reliably solve the training tasks from their original starts, while its capability region may still be concentrated around the reference trajectories.

Forward stage: extending into the policy's own distribution. The forward stage uses the now-solvable original context $c_0 = (o_0, x)$ as a rollout anchor and extends $\mathbb{C}_\eta(\pi)$ to contexts generated by the policy itself. At forward stage i, prefixes are sampled from c_0 with length ℓ_i, initialized at $\ell_0 = d+2$ to match the final reverse-stage horizon and then increased according to a monotone schedule, for example $\ell_i = d + 2 + i\Delta\ell$ with $\Delta\ell > 0$. The resulting prefixes may reach contexts outside the reference trajectories, and these endpoint contexts are then trained with the paired advantage update and incorporated into the expanding capability region.

The two stages therefore use the same update but different prefix distributions. The reverse stage samples prefixes from successive contexts along successful references and moves the capability boundary toward earlier task contexts. The forward stage samples prefixes from policy-generated rollouts starting at c_0 and expands the boundary to contexts outside the support of the references.

Curriculum progression. In the reverse stage, successful reference trajectories provide an ordered sequence of candidate starting contexts. Once the current start reaches the reliability threshold η, moving backward along the reference trajectory tends to place the next start near the expanding capability boundary, as illustrated in Figure 1. We therefore use these starts as practical near-boundary contexts without explicitly computing $\mathbb{N}_{\eta,\delta}(\pi)$ or specifying a numerical δ. In the forward stage, we monitor the mean number of distinct prefix endpoints $|\mathcal{E}|$ per rollout group. When this mean stabilizes over successive updates, indicating that endpoint diversity at the current prefix length has plateaued, we advance the curriculum and increase the prefix length by $\Delta\ell$.Increasing the prefix length lets the policy search for endpoint contexts e that may lie across the capability boundary.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate on ALFWorld (Shridhar et al., 2021) and WebShop (Yao et al., 2022). **ALFWorld** comprises 3,827 embodied household tasks across six categories (PICK, LOOK, CLEAN, HEAT, COOL, and PICK2). Given a textual goal, the agent completes each task through multi-turn interaction with the environment. **WebShop** simulates online shopping through an HTML interface

Table 1: Performance on ALFWorld and WebShop. We report success rates (%) for ALFWorld and task scores and success rates (%) for WebShop. Subscripts denote standard deviations where available. For our method, they are computed over three evaluation seeds using a fixed checkpoint. Baseline results are taken from GiGPO (Feng et al., 2025) and CRPS (Zhang & Yang, 2026). Bold indicates the best mean in each column within a model scale.

	Type	Method	ALFWorld							WebShop	
			Pick	Look	Clean	Heat	Cool	Pick2	All	Score	Succ.
Closed-Source Models											
	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
1.5B	Prompting	Qwen2.5	5.9	5.5	3.3	9.7	4.2	0.0	4.1	23.1	5.2
1.5B	Prompting	ReAct	17.4	20.5	15.7	6.2	7.7	2.0	12.8	40.1	11.3
1.5B	Prompting	Reflexion	35.3	22.2	21.7	13.6	19.4	3.7	21.8	55.8	21.9
1.5B	RL Training	PPO	$64.8_{\pm 3.5}$	$40.5_{\pm 6.9}$	$57.1_{\pm 4.9}$	$60.6_{\pm 6.6}$	$46.4_{\pm 4.0}$	$47.4_{\pm 1.9}$	$54.4_{\pm 3.1}$	$73.8_{\pm 3.0}$	$51.5_{\pm 2.9}$
1.5B	RL Training	RLOO	$88.3_{\pm 3.0}$	$52.8_{\pm 8.6}$	$71.0_{\pm 5.9}$	$62.8_{\pm 8.7}$	$66.4_{\pm 5.5}$	$56.9_{\pm 4.7}$	$69.7_{\pm 2.5}$	$73.9_{\pm 5.6}$	$52.1_{\pm 6.7}$
1.5B	RL Training	GRPO	$85.3_{\pm 1.5}$	$53.7_{\pm 8.0}$	$84.5_{\pm 6.8}$	$78.2_{\pm 7.9}$	$59.7_{\pm 5.0}$	$53.5_{\pm 5.6}$	$72.8_{\pm 3.6}$	$75.8_{\pm 3.5}$	$56.8_{\pm 3.8}$
1.5B	RL Training	CRPS	91.0	72.0	88.0	90.0	73.0	76.0	83.0	73.0	61.0
1.5B	RL Training	GiGPO	$94.4_{\pm 5.9}$	$67.5_{\pm 4.6}$	$\mathbf{94.8}_{\pm 3.8}$	$94.4_{\pm 7.8}$	$79.8_{\pm 4.7}$	$76.4_{\pm 5.4}$	$86.7_{\pm 1.7}$	$83.1_{\pm 1.6}$	$65.0_{\pm 3.2}$
1.5B	RL Training	**Ours**	$\mathbf{97.9}_{\pm 3.7}$	$\mathbf{97.2}_{\pm 4.8}$	$84.2_{\pm 4.23}$	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{80.3}_{\pm 4.0}$	$\mathbf{86.4}_{\pm 4.6}$	$\mathbf{90.1}_{\pm 2.0}$	$\mathbf{88.6}_{\pm 1.5}$	$\mathbf{80.2}_{\pm 3.9}$
7B	Prompting	Qwen2.5	33.4	21.6	19.3	6.9	2.8	3.2	14.8	26.4	7.8
7B	Prompting	ReAct	48.5	35.4	34.3	13.2	18.2	17.6	31.2	46.2	19.5
7B	Prompting	Reflexion	62.0	41.6	44.9	30.9	36.3	23.8	42.7	58.1	28.8
7B	RL Training	PPO	$92.3_{\pm 4.0}$	$64.0_{\pm 8.4}$	$92.5_{\pm 2.4}$	$89.5_{\pm 7.0}$	$80.3_{\pm 2.0}$	$68.8_{\pm 8.3}$	$80.4_{\pm 2.7}$	$81.4_{\pm 3.1}$	$68.7_{\pm 5.1}$
7B	RL Training	RLOO	$87.6_{\pm 4.3}$	$78.2_{\pm 8.3}$	$87.3_{\pm 5.8}$	$81.3_{\pm 7.6}$	$71.9_{\pm 5.2}$	$48.9_{\pm 8.4}$	$75.5_{\pm 4.6}$	$80.3_{\pm 3.2}$	$65.7_{\pm 4.0}$
7B	RL Training	GRPO	$90.8_{\pm 5.1}$	$66.1_{\pm 6.7}$	$89.3_{\pm 5.4}$	$74.7_{\pm 6.9}$	$72.5_{\pm 5.4}$	$64.7_{\pm 7.3}$	$77.6_{\pm 5.2}$	$79.3_{\pm 2.8}$	$66.1_{\pm 3.7}$
7B	RL Training	CRPS	91.0	80.0	86.0	83.0	80.0	71.0	86.0	82.0	74.0
7B	RL Training	GiGPO	$97.7_{\pm 1.6}$	$82.7_{\pm 7.9}$	$98.8_{\pm 1.6}$	$83.7_{\pm 7.2}$	$\mathbf{89.3}_{\pm 8.2}$	$79.2_{\pm 6.6}$	$90.8_{\pm 1.3}$	$84.4_{\pm 2.9}$	$72.8_{\pm 3.2}$
7B	RL Training	**Ours**	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{100.0}_{\pm 0.0}$	$85.9_{\pm 6.0}$	$\mathbf{92.4}_{\pm 2.6}$	$\mathbf{96.1}_{\pm 1.4}$	$\mathbf{92.5}_{\pm 1.1}$	$\mathbf{82.6}_{\pm 1.2}$

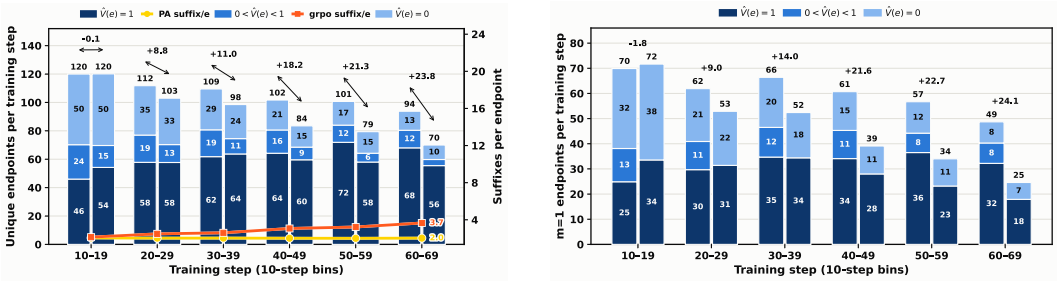
in which the agent searches for products, navigates pages, and makes a purchase based on a user instruction. It contains over 1.1 million products and 12,000 user instructions.

Baselines. We compare against closed-source LLMs, prompting agents, and RL-trained agents as three families of baselines. **Closed-source models** are GPT-4o (Achiam et al., 2023) and Gemini-2.5-Pro (Team et al., 2023), representing strong general-purpose reasoning. **Prompting agents** include Qwen2.5-Instruct, ReAct (Yao et al., 2023) and Reflexion (Shinn et al., 2023), relying on prompts without parameter updates. **RL-based training methods** cover PPO (Schulman et al., 2017), RLOO (Kool et al., 2019; Ahmadian et al., 2024), and GRPO (Shao et al., 2024) under trajectory-level supervision, alongside methods that address long-horizon credit assignment through step-level grouping GiGPO(Feng et al., 2025) and reverse replay CRPS(Zhang & Yang, 2026).

Implementation Details. We adopt Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct (Qwen et al., 2025) as base models. All RL methods use the same backbone and training settings, with 16 tasks per update, 8 rollouts per task. Agents generate reasoning within `<think>...</think>` tags before emitting an action within `<action>...</action>` tags. Episodes are capped at 50 turns in ALFWorld and 15 in WebShop. Our method samples $n = 8$ prefixes per task and 2 suffixes per unqiue endpoint. All training runs on 4 NVIDIA H100 GPUs. More training setups and hyperparameter details are provided in Appendix B.

4.2 Main Results

Table 1 reports the result of our method at both model scales, compared with prompting and representative reinforcement learning methods under the matched evaluation protocol. RL training improves substantially over prompting alone on both benchmarks, confirming the difficulty of learning long-horizon interaction from prompts alone. On ALFWorld, our method reaches 90.1% overall success with Qwen2.5-1.5B-Instruct and 96.1% with Qwen2.5-7B-Instruct. These results exceed GRPO baseline by 17.3 and 18.5 percentage points, respectively, and the competitive GiGPO by 3.4 and 5.3 points. It leads in five of the six task categories at each scale. The exceptions are Clean at 1.5B (84.2% versus GiGPO's 94.8%) and Cool at 7B (85.9% versus 89.3%). The overall gains over GRPO and GiGPO indicate that the method remains effective across tasks with different interaction requirements, rather than relying on improvements in a single category.



(a) (b)

Figure 3: **Prefix endpoint coverage during ALFWorld training with Qwen2.5-7B-Instruct.** In two panels, the left bar of each pair represents PA and the right bar represents GRPO. Both methods are initialized from the same model checkpoint and trained on the same dataset. PA samples exactly two suffixes from each unique endpoint, whereas GRPO observes one continuation for each prefix reaching that endpoint, so its suffix count is determined by $m(e)$. (a) Bars show the mean number of distinct prefix endpoints per training step, grouped by value $\widehat{V}(e)$.Curves show the mean number of suffixes per endpoint. (b) Endpoint counts restricted to those reached by exactly one prefix, $m(e) = 1$. PA's two suffixes can reveal mixed outcomes $\widehat{V}(e) = 0.5$ at these singleton endpoints.

On WebShop, our method achieves a score of 88.6 and a success rate of 80.2% at 1.5B, compared with 83.1 and 65.0% for GiGPO. At 7B, it reaches 92.5 and 82.6%. The score exceeds GiGPO by 8.1 points, while the success rate exceeds CRPS by 8.6 percentage points. Our method leads on both WebShop metrics at both scales. The larger gains in success rate than in task score are consistent with more agents completing the full sequence of search, navigation, and product-selection decisions. The ablation results below examine the contribution of paired credit assignment to these gains.

4.3 Analysis

PA decomposition expands capability region. Figure 3a analyzes the prefix endpoints reached during ALFWorld training with Qwen2.5-7B-Instruct. The bars show the mean number of distinct endpoint contexts e per training step, grouped by value $\widehat{V}(e)$, and the curves show the mean number of observed suffixes per endpoint. PA has 0.1 fewer endpoints per step than GRPO in the first bin but 23.8 more in the final bin, reaching 93.8 versus 70.0 endpoints per step. including 68.0 versus 55.5 with $\widehat{V}(e) = 1$. Meanwhile, PA maintains a fixed budget of $N_{\mathrm{R}} = 2$ suffixes per endpoint by design (Eq. 5), in contrast to GRPO's observed increase from 2.1 to 3.7. This pattern is consistent with a winner-takes-all effect under GRPO: trajectory-level advantages reinforce prefixes associated with successful rollouts, so sampled prefixes may converge on fewer endpoint contexts. PA instead evaluates each distinct endpoint with two suffixes and separates prefix selection from continuation quality, keeping more potentially solvable endpoints available for learning.

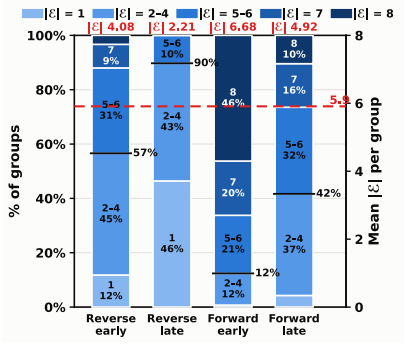


Figure 4: Distribution of the number of unique prefix endpoint contexts $|\mathcal{E}|$ reached per training on Webshop.

Figure 3b offers a clearer view of this divergence by isolating endpoints reached by exactly one prefix, $m(e) = 1$. For these singleton endpoints, GRPO observes one binary outcome and cannot assign an intermediate empirical value, PA's two suffixes produce $\widehat{V}(e) = 0.5$ when one succeeds and the other fails, enabling the within-endpoint comparison in its right advantage (Eq. 7). From the first to the final bin, PA's singleton coverage falls from about 70 to 48.7 endpoints per step, yet its value-1 count rises from about 25 to 32.2, while its value-0.5 count falls from about 13 to 8.1. GRPO loses both singleton coverage (about 72 to 24.6) and value-1 endpoints (about 34 to 17.9). These contrasting trends are consistent with PA's within-endpoint comparisons helping partially solvable

Table 2: Ablation study on WebShop using Qwen2.5-1.5B-Instruct.

Curriculum	Rev. PA	Fwd. PA	Succ. (%)
Reverse only	–	–	68.8
Reverse + Forward	–	–	74.7
Reverse + Forward	✓	–	76.6
Reverse + Forward	✓	✓	**80.2**

contexts become more reliably solvable, while GRPO's observed endpoint distribution contracts in both breadth and high-value count.

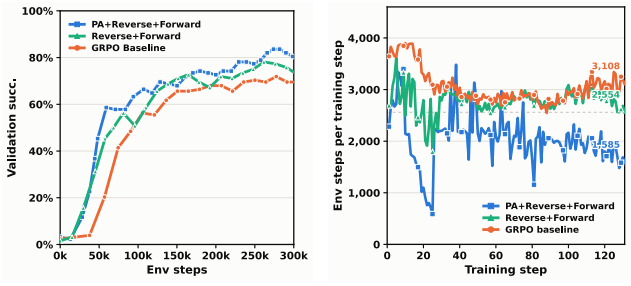
Why reverse, then forward? Figure 4 reports the distribution of $|\mathcal{E}|$, the number of unique prefix endpoint contexts reached per training group, across four consecutive training stages on WebShop. The reverse stage drives the mean $|\mathcal{E}|$ from 4.08 down to 2.21 with the $|\mathcal{E}|{=}1$ fraction reaching 46% by late reverse, evidence that the policy has collapsed most rollouts onto a single reachable endpoint per group. Through this collapse, the reverse stage establishes the initial capability region, with tight concentration on the reference trajectory yielding a small but reliable solvable region. The forward stage takes this region as its anchor and pushes the mean $|\mathcal{E}|$ back to 6.68 in early forward, with 66% of groups at $|\mathcal{E}| \geq 7$, before settling at 4.92 in late forward. Read across the four stages, the capability region is first anchored then expanded. The reverse stage establishes it along the reference trajectory, and the forward stage continues expansion into the off-reference states the policy now visits.

4.4 Ablation Study

We conduct an ablation study to evaluate the effectiveness of each component in our method. As illustrated in Table 2, the two curriculum stages and paired advantages both contribute to the performance on WebShop using Qwen2.5-1.5B-Instruct. The reverse-only configuration baseline without paired advantages achieves a success rate of 68.8%. Adding the forward stage increases performance to 74.7%. Incorporating paired advantages into the reverse stage further improves the success rate to 76.6%, and in both curriculum stages achieves the best performance of 80.20%, which is 11.4 points higher than the reverse-only baseline. The gains from applying PA in both stages support the benefit of separately crediting prefix actions for the endpoints they reach and suffix actions for their outcomes from those endpoints.

4.5 Training Efficiency and Cost Analysis

Figure 5a shows that PA achieves higher validation success at matched environment-interaction budgets, despite reaching about 5.9 distinct prefix endpoints per group in Figure 4 and sampling two suffixes from each non-terminal endpoint. Figure 5b helps explain why the additional branching does not necessarily increase interaction cost. Reverse training starts near the ends of successful references, keeping the remaining rollouts short. By late reverse, eight sampled prefixes reach only 2.21 distinct endpoints per group on average, limiting the number of endpoints that require suffix sampling. In forward training, endpoint diversity increases as PA explores contexts beyond the references. The reverse-trained policy helps avoid long, unproductive continuations from these endpoints, allowing PA to explore more contexts without a proportional increase in environment steps. The occasional cost spikes are consistent with batches reaching more endpoints, but PA generally uses fewer environment steps per training step than the baselines.



(a) (b)

Figure 5: (a) Validation success against cumulative model-generated environment steps. (b) Environment steps consumed per training step for each arm.

5 Related Work

Reinforcement Learning for Long-Horizon LLM Agents. LLM agents extend language models to long-horizon sequential decisions through repeated interaction with environments. The paradigm has evolved from static text generation, through single-turn tool use (Schick et al., 2023), to multi-turn sequential decision making, exemplified by ReAct and Reflexion (Yao et al., 2023; Shinn et al., 2023). In LLM training, the RLHF pipeline paired PPO with a learned value function (Ouyang et al., 2022; Schulman et al., 2017), while RLOO and GRPO removed the need for a separate critic (Ahmadian et al., 2024; Shao et al., 2024), and AGILE and WebRL subsequently applied reinforcement learning to agent settings (Feng et al., 2024; Qi et al., 2025). Recent work has brought online RL to long-horizon agents through dedicated multi-turn environments and scalable training systems such as RAGEN, AgentGym-RL, and SkyRL-Agent (Wang et al., 2025; Xi et al., 2025; Cao et al., 2025). These efforts extend RL training to complete agent episodes across search, software engineering, and mobile GUI tasks (Kalyan & Andrews, 2025; Cao et al., 2025; Xu et al., 2026).

Credit Assignment. In sparse reward settings, long-horizon agents typically receive supervision only from a terminal outcome, which makes credit assignment over intermediate decisions difficult. Process reward models and search-based approaches introduce intermediate signals, but require additional manual annotations and expensive search computation, respectively (Lightman et al., 2024; Hao et al., 2023). Later agentic RL methods such as GiGPO and HGPO derive finer grained comparisons from repeated states across rollouts without externally labeled process rewards (Feng et al., 2025; He et al., 2026), whereas STAPO derives a selective step-level signal via normalized entropy (Qi et al., 2026). More recent approaches modify the rollout structure itself by aggregating trajectories into transition graphs or introducing structured branching and information guided exploration, as in GraphGPO, SPARK, Tree GRPO, and AEPO (Cheng et al., 2026; Wu et al., 2026; Ji et al., 2026; Dong et al., 2025). E3-TIR takes a complementary approach by reusing a common continuation to compare prefixes, allowing downstream outcomes to provide evidence about prefix quality (Guo et al., 2026). Taken together, these studies suggest that long-horizon credit assignment depends on how rollout structure determines which decisions share, inherit, or receive separately estimated learning signals.

Curriculum Learning. Curriculum learning structures the training distribution so that competence acquired on simpler subproblems supports learning on more difficult ones (Bengio et al., 2009). In reinforcement learning and robot control, this principle has been implemented by adapting restart states or task sequences around near-goal and demonstrated states (Tavakoli et al., 2019; Florensa et al., 2017; Resnick et al., 2018; Ivanovic et al., 2019; Wöhlke et al., 2020; Sharma et al., 2021). In agentic RL, curricula modify task distribution, the available supervision, or the interaction environment to regulate the difficulty of multi-turn interaction (Tan et al., 2025; Tzannetos et al., 2026; Xia et al., 2025). Recent work derives curricula directly from interaction outcomes, using failed attempts to generate new tasks and successful trajectories to identify informative restart states (Qi et al., 2025; Zhang & Yang, 2026). Our method builds on curriculum learning by first expanding the training start backward along successful reference trajectories and then broadening training to states generated by the policy itself.

6 Conclusion

We train long-horizon agents from sparse terminal rewards by progressively expanding the policy's capability region. Paired Advantage separately credits prefixes for reaching more solvable contexts and suffixes for task completion with a reverse-to-forward curriculum moves from successful references to policy-generated contexts. The method outperforms strong baselines on ALFWorld and WebShop at both model scales.

Limitations. Reverse training requires successful references, and suffix sampling requires restarting from intermediate contexts. These requirements restrict use when references or reliable resets are unavailable. If all sampled suffixes fail, both advantages vanish. Starting near known solutions reduces this risk, but expansion may stall when solvable contexts remain beyond the reach of sampled prefixes.

AI USE STATEMENT

In this work, we used generative AI tools to assist with code implementation and to improve the grammar, clarity, and readability of the manuscript. We have not used generative AI tools for generating research proposals, methodological designs, experimental plans, or scientific conclusions. We have reviewed and tested all AI-assisted code, and all reported results come from running the verified implementation by 2 authors. We also reviewed all AI-assisted writing to ensure it accurately reflects the methods, results, and claims of this work. We take full responsibility for the final content of this paper, including all text, claims, code, and other artifacts produced with the assistance of generative AI.

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 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024.

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

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 41–48, 2009.

Shiyi Cao, Dacheng Li, Fangzhou Zhao, Shuo Yuan, Sumanth R Hegde, Connor Chen, Charlie Ruan, Tyler Griggs, Shu Liu, Eric Tang, et al. Skyrl-agent: Efficient rl training for multi-turn llm agent. *arXiv preprint arXiv:2511.16108*, 2025.

Xin Cheng, Shuo He, Lang Feng, HaiYang Xu, Ming Yan, Lei Feng, and Bo An. Beyond trajectory-level attribution: Graph-based credit assignment for agentic reinforcement learning. *arXiv preprint arXiv:2605.26684*, 2026.

Guanting Dong, Licheng Bao, Zhongyuan Wang, Kangzhi Zhao, Xiaoxi Li, Jiajie Jin, Jinghan Yang, Hangyu Mao, Fuzheng Zhang, Kun Gai, et al. Agentic entropy-balanced policy optimization. *arXiv preprint arXiv:2510.14545*, 2025.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Peiyuan Feng, Yichen He, Guanhua Huang, Yuan Lin, Hanchong Zhang, Yuchen Zhang, and Hang Li. Agile: A novel reinforcement learning framework of llm agents. volume 37, pp. 5244–5284, 2024.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78, pp. 482–495. PMLR, 2017.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Weiyang Guo, Zesheng Shi, Liye Zhao, Jiayuan Ma, Zeen Zhu, Junxian He, Min Zhang, and Jing Li. E3-TIR: Enhanced experience exploitation for tool-integrated reasoning. 2026.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 8154–8173, 2023.

Shuo He, Lang Feng, Xin Cheng, Lei Feng, Bo An, et al. Hierarchy-of-groups policy optimization for long-horizon agentic tasks. In *International Conference on Learning Representations*, volume 2026, pp. 27572–27593, 2026.

Boris Ivanovic, James Harrison, Apoorva Sharma, Mo Chen, and Marco Pavone. Barc: Backward reachability curriculum for robotic reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 15–21. IEEE, 2019.

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for llm agent reinforcement learning. 2026:87362–87388, 2026.

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.

Vivek Kalyan and Martin Andrews. Reinforcement learning for long-horizon multi-turn search agents. *arXiv preprint arXiv:2510.24126*, 2025.

Wouter Kool, Herke Van Hoof, and Max Welling. Buy 4 reinforce samples, get a baseline for free! 2019.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. 2024:39578–39601, 2024.

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.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

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. volume 35, pp. 27730–27744, 2022.

Eduardo Pignatelli, Johan Ferret, Matthieu Geist, Thomas Mesnard, Hado van Hasselt, Olivier Pietquin, and Laura Toni. A survey of temporal credit assignment in deep reinforcement learning. *arXiv preprint arXiv:2312.01072*, 2023.

Qiuyi Qi, Tian Liang, Mutian Bao, Jinjian Zhang, Dongnan Liu, Wei Zhou, Linjian Mo, Ming Kong, Jie Liu, Feng Zhang, et al. Stapo: Selective trajectory-aware policy optimization for llm agent training. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28371–28392, 2026.

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*, volume 2025, pp. 79791–79821, 2025.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. 2025.

Cinjon Resnick, Roberta Raileanu, Sanyam Kapoor, Alexander Peysakhovich, Kyunghyun Cho, and Joan Bruna. Backplay: "man muss immer umkehren". *arXiv preprint arXiv:1807.06919*, 2018.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

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. volume 36, pp. 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, Yang Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Archit Sharma, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Autonomous reinforcement learning via subgoal curricula. In *Advances in Neural Information Processing Systems*, volume 34, pp. 18474–18486, 2021.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. volume 36, pp. 8634–8652, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, 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 (ICLR)*, 2021.

Weiting Tan, Xinghua Qu, Ming Tu, Meng Ge, Andy T. Liu, Philipp Koehn, and Lu Lu. Process-supervised reinforcement learning for interactive multimodal tool-use agents. *arXiv preprint arXiv:2509.14480*, 2025.

Stone Tao, Arth Shukla, Tse-kai Chan, and Hao Su. Reverse forward curriculum learning for extreme sample and demo efficiency. In *International Conference on Learning Representations*, volume 2024, pp. 46955–46976, 2024.

Arash Tavakoli, Vitaly Levdik, Riashat Islam, Christopher M. Smith, and Petar Kormushev. Exploring restart distributions. In *Proceedings of the 4th Multidisciplinary Conference on Reinforcement Learning and Decision Making*, 2019.

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.

Georgios Tzannetos, Parameswaran Kamalaruban, and Adish Singla. Curriculum design for trajectory-constrained agent: Compressing chain-of-thought tokens in llms. volume 38, pp. 19881–19907, 2026.

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.

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.

Jan Wöhlke, Felix Schmitt, and Herke van Hoof. A performance-based start state curriculum framework for reinforcement learning. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1503–1511, 2020.

Jinyang Wu, Shuo Yang, Yuhao Shen, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. Spark: Strategic policy-aware exploration via dynamic branching for long-horizon agentic learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23981–24004, 2026.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Honglin Guo, Jiaqi Liu, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, et al. Agentgym-rl: Training llm agents for long-horizon decision making through multi-turn reinforcement learning. *arXiv preprint arXiv:2509.08755*, 2025.

Peng Xia, Kaide Zeng, Jiaqi Liu, Can Qin, Fang Wu, Yiyang Zhou, Caiming Xiong, and Huaxiu Yao. Agent0: Unleashing self-evolving agents from zero data via tool-integrated reasoning. *arXiv preprint arXiv:2511.16043*, 2025.

Yifan Xu, Xiao Liu, Xinghan Liu, Jiaqi Fu, Jiayu Huang, Hanchen Zhang, Bohao Jing, Shudan Zhang, Yuting Wang, Yuxiao Dong, et al. Mobilerl: Online agentic reinforcement learning for mobile gui agents. In *International Conference on Learning Representations*, volume 2026, pp. 35282–35315, 2026.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. volume 35, pp. 20744–20757, 2022.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Zijing Zhang and Xiajie Yang. CRPS: Curriculum replay via progressive suffixes from successful trajectories for long-horizon LLM agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 13891–13904, 2026.

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

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: Training language model agents via hierarchical multi-turn rl. *arXiv preprint arXiv:2402.19446*, 2024.

A ALGORITHM

Algorithm 1 summarizes the training procedure. Endpoint grouping depends on the full decision context, which includes both the task and the interaction history. As a result, the same observation does not by itself guarantee the same endpoint. For each distinct endpoint, we sample N_{R} suffixes once and allow every prefix reaching that endpoint to reuse them. Each iteration thus samples $|\mathcal{E}|N_{\mathrm{R}}$ suffixes instead of nN_{R}. In the left advantage, the factor $m(e)$ preserves the number of prefixes reaching each endpoint in the group baseline. Deduplication then reduces suffix sampling without changing how prefixes are weighted. The reverse and forward stages share the same paired advantage update. The transition between the two stages is determined by the estimated completion reliability at the original task starts.

Algorithm 1 Training LLM agents with Depth Before Breadth

Require: Policy π_θ, tasks $\mathcal{D}$, successful references $\{\tau_x^*\}$, group size n, suffix count N_{R}, reverse stride d, threshold η, forward horizons $\{\ell_i\}$
Ensure: Updated policy π_θ

Initialize the curriculum: $q \leftarrow \mathrm{reverse}$; $i \leftarrow 0$
for each training iteration **do**
  Update the old policy model $\theta_{\mathrm{old}} \leftarrow \theta$ and sample task $x \sim \mathcal{D}$
  // Select the curriculum anchor
  if $q = \mathrm{reverse}$ **then**
    Move along the reference: $r_i \leftarrow \max\{T_x - 1 - id, 0\}$; $(c, \ell) \leftarrow (c^*_{r_i}, d + 2)$
  else
    Start from the original context: $(c, \ell) \leftarrow (c_0, \ell_i)$
  end if
  // Sample prefixes and group shared endpoints
  for $k = 1, \ldots, n$ **do**
    Sample a prefix: $\tau_k^{\mathrm{L}} \sim \pi_{\theta_{\mathrm{old}}}(\cdot \mid c)$; $e_k \leftarrow c_{\ell,k}$
  end for
  Group endpoints: $\mathcal{E} \leftarrow \{e_k\}_{k=1}^n$; $\mathcal{I}(e) \leftarrow \{k : e_k = e\}$; $m(e) \leftarrow |\mathcal{I}(e)|$
  // Sample suffixes and compute paired advantages
  for each $e \in \mathcal{E}$ **do**
    Sample continuations: $\{(\tau^{\mathrm{R}}_{e,j}, R_{e,j})\}_{j=1}^{N_{\mathrm{R}}}$ from e under $\pi_{\theta_{\mathrm{old}}}$
    Estimate endpoint value $\widehat{V}(e)$ by Eq. 5
    Compute suffix advantages $A^{\mathrm{R}}(e, j)$ by Eq. 7
  end for
  Compute prefix advantages $A^{\mathrm{L}}(e)$ by Eq. 6
  for $k = 1, \ldots, n$ **do**
    Select a suffix: $j_k \sim \mathrm{Unif}\{1, \ldots, N_{\mathrm{R}}\}$
    Stitch the trajectory: $\tau_k \leftarrow \tau_k^{\mathrm{L}} \oplus \tau^{\mathrm{R}}_{e_k, j_k}$
    Assign $A^{\mathrm{L}}(e_k)$ to τ_k^{L} and $A^{\mathrm{R}}(e_k, j_k)$ to $\tau^{\mathrm{R}}_{e_k, j_k}$
  end for
  // Update the policy and curriculum
  Update θ by maximizing $\mathcal{J}(\theta)$ in Eq. 8
  if $q = \mathrm{reverse}$ and $\widehat{p}_{\pi_\theta}(c^*_{r_i}) \geq \eta$ **then**
    if the original starts pass the threshold for most training tasks **then**
      Switch to forward: $q \leftarrow \mathrm{forward}$; $i \leftarrow 0$
    else
      Move the start backward: $i \leftarrow i + 1$
    end if
  else if $q = \mathrm{forward}$ and the schedule advances **then**
    Increase the prefix horizon: $i \leftarrow i + 1$
  end if
end for

B Implementation Details

B.1 Details of Training

Hyperparameters for ALFWorld. The actor is optimized for one epoch per update using Adam with a constant learning rate of 10^{-6} and gradient clipping at 1.0. Training rollouts use temperature 1.0 and top-p 0.95. The terminal reward is binary and is scaled by 10 for training. The reverse curriculum stride d and forward curriculum stride $\Delta\ell$ are both 3, and the reliability threshold η is 0.6.

Table 3: Hyperparameters of ALFWorld and WebShop.

Config	ALFWorld	WebShop
`train_batch_size`	32	32
`n_samples_per_prompt`	8	8
`ppo_mini_batch_size`	16	16
`ppo_micro_batch_size_per_gpu`	2	1
`ppo_max_token_len_per_gpu`	12288	15360
`log_prob_max_token_len_per_gpu`	12288	15360
`log_prob_micro_batch_size_per_gpu`	2	1
`use_dynamic_bsz`	False	False
`ulysses_sequence_parallel_size`	1	1
`tensor_model_parallel_size`	1	1
`temperature` (rollout)	1.0	1.0
`top_p` (rollout)	0.95	0.95
`temperature` (val)	0	0
`top_p` (val)	1.0	1.0
`actor_learning_rate`	10^{-6}	10^{-6}
β_1, β_2 / weight_decay	0.9, 0.999 / 0.01	0.9, 0.999 / 0.01
`grad_clip`	1.0	1.0
`max_epochs`	1	1
`lr_scheduler`	constant	constant
`max_prompt_length`	2048	2048
`max_response_length`	10240	13312
`max_model_len`	12288	15360
`multi_turn_max_turns`	50	15
`action_budget`	120	80
`clip_ratio`	0.2	0.2
`clip_ratio_c` (dual-clip)	3.0	3.0
`use_kl_loss` / `kl_coef`	False / 0	False / 0
`use_kl_in_reward`	False	False
`entropy_coeff`	0	0
`loss_agg_mode`	seq-mean-token-mean	
`norm_adv_by_std_in_grpo`	True	True
Reverse curriculum stride (d)	3	3
Forward curriculum stride ($\Delta\ell$)	3	3
Reliability threshold (η)	0.6	0.78
Reward type	Binary	Binary
Reward scale	10	10

Hyperparameters for WebShop. WebShop shares ALFWorld's training batch size of 32, eight samples per prompt, learning rate, rollout sampling settings, and terminal reward scale. Its maximum prompt and response lengths are 2,048 and 13,312 tokens, respectively. Both curriculum strides are 3, and the reliability threshold η is 0.78. Table 3 lists the full configuration.

Computing Details. For ALFWorld and WebShop, both Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct models are trained on 4×NVIDIA H100 GPUs. ALFWorld is trained for 100 iterations and is evaluated on the seen task categories and instance variants, containing 140 tasks. WebShop is

trained for 150 iterations and is evaluated on the test split of 500 tasks. For both environments, we sample 128 tasks at random per run and average results over 3 matched random seeds.

B.2 PROMPTS

Figures 6 and 7 show the prompt templates for WebShop and ALFWorld. At the start of an episode, the system message establishes the agent's task and output format. The {task_description} and {current_observation} denote information supplied by the environment. WebShop provides the shopping instruction and current page, while ALFWorld also provides the canonical TextWorld goal. Each environment message supplies the actions currently available and the remaining action count.

Prompt Template for WebShop

 # System message
You are an autonomous agent operating the WebShop text environment.
Your goal is to find, configure, and buy the product that best satisfies the shopping instruction.
Every assistant turn is already prefixed with <think>. Continue with the reasoning, close it with </think>, then output exactly one listed action inside <action> ... </action>.
Never invent a click target that is not listed for the current page.

 # User message at each turn
Shopping instruction: {task_description}
Current page: {current_observation}
Available actions:
- {action_1}
- {action_2}
- ...

Actions remaining: {max_steps - step}

Figure 6: Prompt template used for the WebShop agent.

The agent then interacts within a continuing conversation. After each action, the resulting observation is added to the interaction history and the agent generates its next response from the updated context. The prompt therefore supports a structured trajectory of observations and actions rather than a separate, complete prompt for each decision. In both environments, the <think>...</think> block contains the agent's reasoning, followed by exactly one listed action in the <action>...</action> block. WebShop supplies the opening <think> tag as an assistant prefix, whereas ALFWorld requires the model to generate both complete blocks.

Prompt Template for ALFWorld

 # System message

You are an autonomous agent operating the ALFWorld text environment.
Follow the task and choose only from the admissible actions.
Every reply must contain exactly one <think>... </think> block followed by exactly one <action> ... </action> block.
Do not omit either block or output more than one action.
Never invent an action that is not listed under "Admissible actions" for the current step.

 # User message at each turn

Task instruction: {task_description}
Canonical TextWorld goal: {goal_description}
Current observation: {current_observation}
Admissible actions:
- {action_1}
- {action_2}
- ...
- {action_N}

Actions remaining: {max_steps - step}

Figure 7: Prompt template used for the ALFWorld agent.

C Case Study

C.1 WebShop

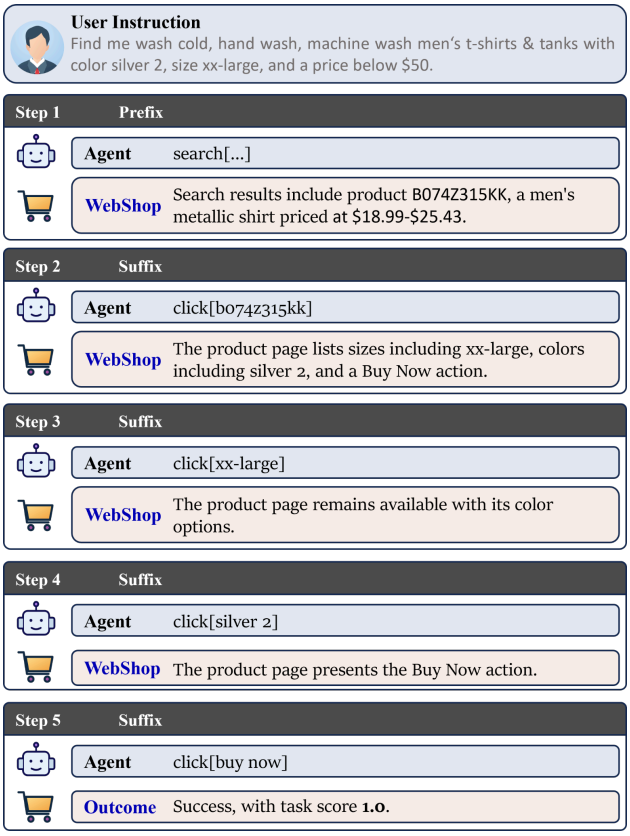


Figure 8: Illustration of a WebShop shopping task through multi-turn interaction.

Figure 8 traces a WebShop shopping task from search to purchase. The prefix issues a search based on the user's constraints and reaches a results page with a candidate product. The suffix opens its product page, selects the requested size and color, and completes the purchase. The example shows how a search action determines the context from which the remaining decisions are made.

C.2 ALFWORLD

Figure 9 follows an agent tasked with placing a chilled apple on a counter. In the prefix, it locates and picks up the apple before reaching the fridge. The suffix cools the apple and returns it to the countertop. The recorded suffix also includes one turn in which the agent produces reasoning but takes no environment action.

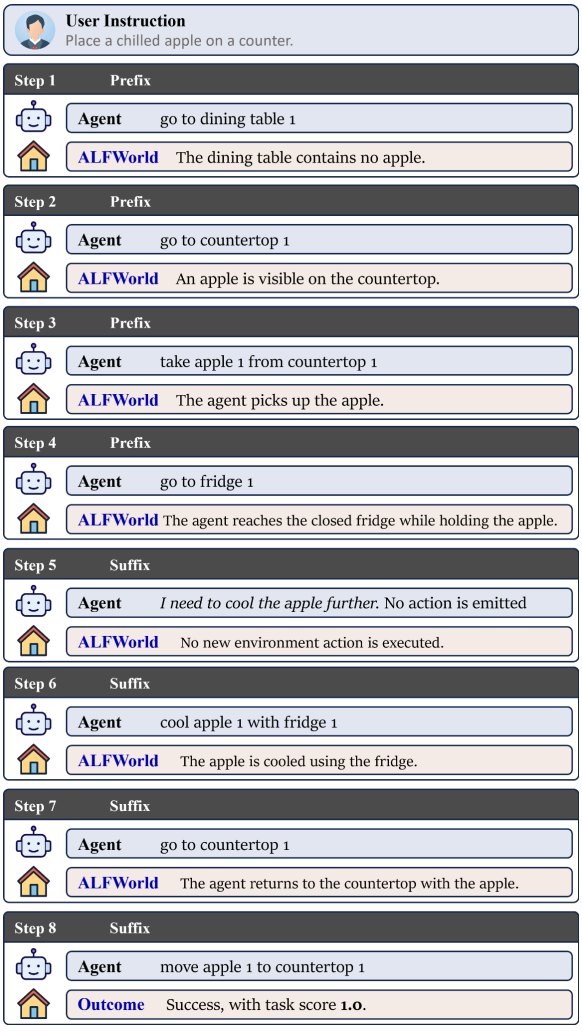


Figure 9: Illustration of an ALFWorld household task through multi-turn interaction.