

DEPPO: Dual Experience Pool Policy Optimization for Long-Horizon Reinforcement Learning

Haosen Dong*, Jiaxin Guo*, MUYANG Yan, Ye Yuan, Mengxi Jia[†], Hao Sun[†]

China Telecom Artificial Intelligence Technology (Beijing) Co., Ltd.

*Equal contribution; [†]corresponding authors.

mxjia@pku.edu.cn, sunh10@chinatelecom.cn

Abstract

Reinforcement learning for long-horizon agents and multi-step reasoning models typically relies on outcome-level rewards, which provide only trajectory-level supervision and make it difficult to identify the contribution of individual intermediate actions. As a result, redundant actions in successful trajectories may be over-reinforced, while useful actions in failed trajectories may be undesirably penalized. To address this issue, we propose **Dual Experience Pool Policy Optimization (DEPPO)**, a critic-free and experience-driven framework for adaptive step-level credit assignment. DEPPO maintains two structured experience pools for successful and failed trajectories, and estimates the relative quality of each action by comparing its state-conditional likelihood under the two pools. The resulting success–failure preference signal is used to adaptively rescale the advantage, amplifying actions aligned with successful patterns while suppressing those associated with failure modes. Furthermore, DEPPO introduces an asymmetric and dynamically evolving update mechanism: the success pool emphasizes both correctness and novelty through freshness-aware insertion, while the failure pool focuses on consistently risky behaviors, with decay and pruning to keep the memory compact and up-to-date. Without additional value models or process-level supervision, DEPPO provides a lightweight, interpretable, and effective solution for fine-grained credit assignment in long-horizon reinforcement learning.

1 Introduction

Reinforcement learning (RL) for long-horizon agents and multi-step reasoning has become a key paradigm for improving large language models on complex tasks (Wang et al., 2026d; Zhang et al., 2025). However, these settings typically provide supervision only at the trajectory level, while deci-

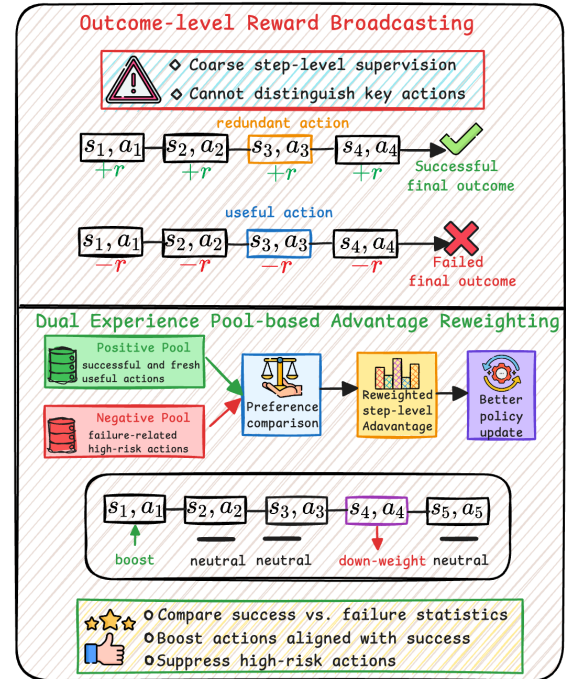


Figure 1: Outcome-level reward broadcasting cannot distinguish key intermediate actions. DEPPO addresses this issue by comparing task-conditioned success and failure statistics from dual experience pools to produce experience-guided step-level credit signals for adaptive policy optimization.

sions are made at the step level, leading to a fundamental credit assignment mismatch (Xi et al., 2025; Chen et al., 2025). As illustrated in Fig. 1, uniformly broadcasting terminal rewards to all steps fails to distinguish causally relevant actions from incidental ones, often reinforcing suboptimal behaviors or penalizing useful decisions (Li et al., 2026; Wang et al., 2026c). This issue is exacerbated in long-horizon and high-uncertainty scenarios, resulting in noisy training signals and unstable policy optimization (Wang et al., 2026b; Lattimer et al., 2025).

Existing approaches attempt to address this challenge from different perspectives. Critic-based

methods estimate intermediate values to reduce variance, but introduce additional complexity and potential bias (Mnih et al., 2016). Critic-free methods improve efficiency via uniform reward propagation or intra-group comparisons, yet still rely on trajectory-level signals for supervision. Other approaches leverage process supervision or external reward models, but typically require additional annotations or task-specific designs (Khalifa et al., 2025). **Despite these efforts, most methods derive learning signals solely from the current trajectory, without exploiting the rich historical experience accumulated during training.** This observation raises a fundamental question:

Can historical trajectories be transformed from passive records into state-conditioned, contrastive signals for reliable step-level credit assignment?

We answer this question by proposing DEPPO, an experience-driven framework that explicitly utilizes historical successful and failed trajectories for adaptive step-level credit assignment. DEPPO organizes past trajectories into separate success and failure pools under a task–state–action hierarchy, capturing reusable patterns of effective and risky decisions. For each action in the current trajectory, it estimates its state-conditional likelihood under both pools and derives a preference signal to reweight the advantage. This enables step-level supervision that reflects cross-trajectory statistical regularities rather than relying solely on trajectory-level returns. To ensure continual adaptation, DEPPO adopts asymmetric updates with freshness-aware insertion and decay-based pruning, forming a closed loop between experience accumulation and policy optimization. Experiments demonstrate that DEPPO improves training stability and final performance across long-horizon tasks.

In summary, our contributions are threefold: (1) we introduce an experience-guided policy optimization paradigm that converts historical trajectories into task-conditioned success–failure credit signals; (2) we propose dual experience pools with asymmetric and dynamic updates to capture effective behaviors and risky patterns; (3) experiments on ALFWorld, WebShop, and Sokoban demonstrate the effectiveness and generality of DEPPO across different long-horizon decision-making settings.

2 Related Work

Outcome-level Reinforcement Learning for LLM Reasoning. Recent RL methods for LLM reasoning mainly rely on outcome-level or verifiable rewards. DeepSeekMath (Shao et al., 2024), DeepSeek-R1 (Guo et al., 2025a), and Kimi k1.5 (Team et al., 2025) demonstrate the effectiveness of large-scale outcome-reward RL for long-chain reasoning and self-correction (Shao et al., 2024; Guo et al., 2025a). Follow-up methods such as DAPO (Yu et al., 2025), GSPO (Zheng et al., 2025), and AReaL (Fu et al., 2025) further improve training stability and efficiency, while related frameworks extend outcome-level RL to retrieval-augmented reasoning and long-horizon agents (Jin et al., 2025). However, these methods still mainly optimize sequence-level objectives, making fine-grained credit assignment difficult.

Process-level Supervision and Credit Assignment. Another line of work introduces process reward models or step-wise verifiers to provide finer-grained supervision. Math-Shepherd (Wang et al., 2024), OmegaPRM (Luo et al., 2024), TreePLV (He et al., 2024), and PQM (Li and Li, 2024) construct process-level signals through automatic annotation, tree search, preference learning, or Q-value ranking. Recent methods such as ThinkPRM (Khalifa et al., 2025), R-PRM (She et al., 2025), and Reward Reasoning Models (Guo et al., 2025b) further improve step-wise verification with generative reasoning or reward reasoning.

3 Method

3.1 Overview

We propose DEPPO, a critic-free and experience-driven policy optimization framework for adaptive step-level credit assignment in long-horizon reinforcement learning.

The key idea is to transform historical trajectories into structured experience, by explicitly modeling state-conditioned success and failure statistics. Instead of treating all actions within a trajectory equally, DEPPO evaluates, under the exact same task-state context, whether an action is more associated with successful or failed outcomes, and uses this contrastive signal to reweight its advantage.

As illustrated in Fig. 2, DEPPO maintains dual experience pools, performs experience-guided advantage reweighting, and updates the pools in a closed loop with policy optimization. This de-

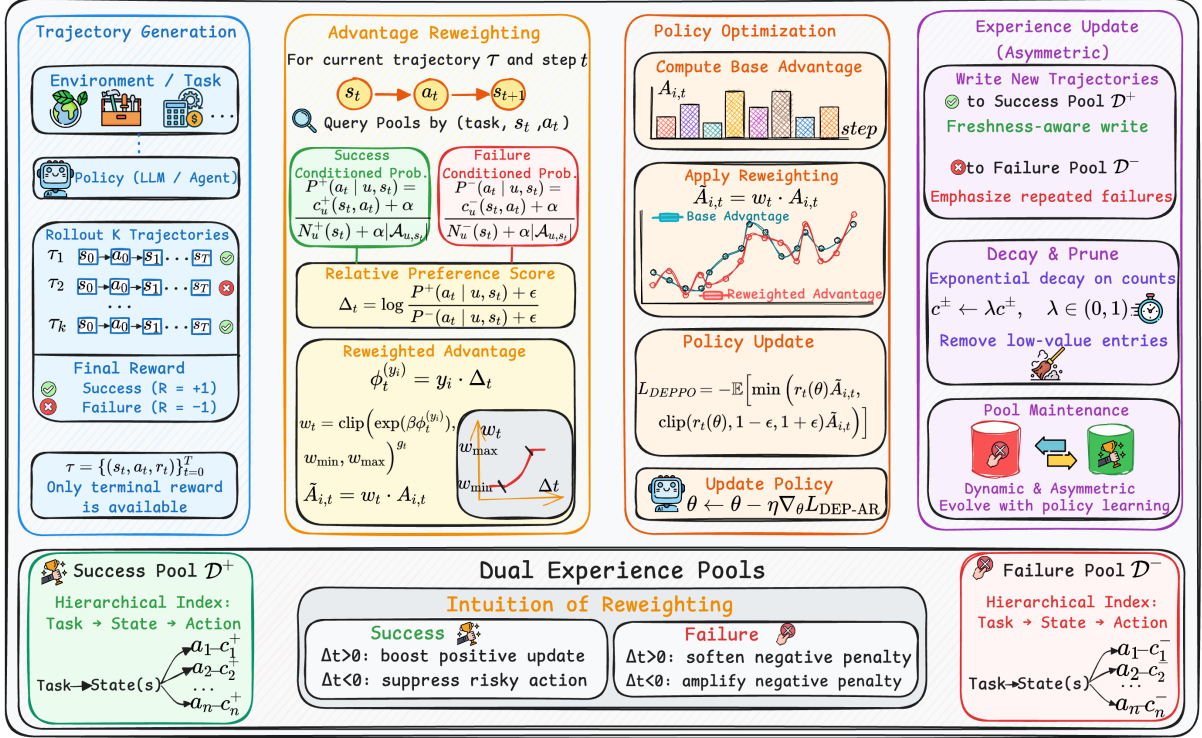


Figure 2: Overview of **DEPPPO**. DEPPPO performs experience-driven policy optimization for adaptive step-level credit assignment. Historical trajectories are organized into dual experience pools that model state-conditional success and failure statistics. For each action, DEPPPO compares its likelihood under the two distributions and adaptively rescales the advantage used for policy optimization. The experience pools are updated asymmetrically with freshness-aware insertion and decay-based pruning, forming a closed-loop learning process.

sign enables more accurate and stable credit assignment without requiring additional value models or process-level supervision.

3.2 Dual Experience Pools

DEPPPO maintains two structured experience pools: $\mathcal{D}^+$ (storing state-action statistics from successful trajectories) and $\mathcal{D}^-$ (storing state-action statistics from failed trajectories).

Each pool is organized as a hierarchical mapping (**task id** $\rightarrow$ **state** $\rightarrow$ **action statistics**). Equivalently, each entry is indexed by a task-state key (u, s) and stores an action histogram. Following the notation in Fig. 2, the count of action a under state s for task u is written as

$$\mathcal{D}^\pm[(u, s)][a] = c_u^\pm(s, a), \quad (1)$$

where the task index u is explicitly included in the statistics. Here, $c_u^+(s, a)$ denotes how often action a has appeared under state s in successful trajectories of task u , while $c_u^-(s, a)$ denotes the corresponding count in failed trajectories.

For each state, the total count in the success and

failure pools is defined as

$$\begin{aligned} N_u^+(s) &= \sum_a c_u^+(s, a), \\ N_u^-(s) &= \sum_a c_u^-(s, a). \end{aligned} \quad (2)$$

The task-level key is important because identical or highly similar states may require different actions under different tasks. In implementation, keys are constructed from normalized environment-level observations and executable actions rather than token IDs or free-form reasoning traces.

For each current step (u, s_t, a_t) , DEPPPO queries both $\mathcal{D}^+$ and $\mathcal{D}^-$ to estimate how likely the action is under historical success and failure patterns. Let $\mathcal{A}_{u, s_t}$ denote the union of actions observed under state s_t for task u in both pools, together with the current action:

$$\begin{aligned} \mathcal{A}_{u, s_t} &= \{a : c_u^+(s_t, a) > 0\} \\ &\cup \{a : c_u^-(s_t, a) > 0\} \cup \{a_t\}. \end{aligned} \quad (3)$$

Since the experience pools may be sparse during early training, directly using raw frequencies can

produce unstable estimates. DEPPO therefore applies additive smoothing to obtain task-conditioned state-action probabilities:

$$P^\pm(a_t | u, s_t) = \frac{c_u^\pm(s_t, a_t) + \alpha}{N_u^\pm(s_t) + \alpha|\mathcal{A}_{u,s_t}|}, \quad (4)$$

where $\alpha > 0$ is the smoothing coefficient, P^+ (c^+ , N^+) corresponds to successful trajectories and P^- (c^- , N^-) to failed ones.

To compare success and failure tendencies, DEPPO uses the relative preference score:

$$\Delta_t = \log \frac{P^+(a_t | u, s_t) + \epsilon}{P^-(a_t | u, s_t) + \epsilon}, \quad (5)$$

where $\epsilon > 0$ is a small constant for numerical stability.

The log-ratio form has two advantages. First, it provides a symmetric comparison between success-conditioned and failure-conditioned statistics. Second, it naturally measures the relative evidence of success over failure: $\Delta_t > 0$ indicates that a_t is more associated with successful trajectories under the same task-state condition, while $\Delta_t < 0$ indicates that it is more associated with failed trajectories.

3.3 Support-Aware Advantage Reweighting

The preference score Δ_t can be unreliable when the corresponding task-state condition has insufficient historical support. Therefore, DEPPO introduces a support-aware gate to activate reweighting only when the dual-pool statistics are sufficiently reliable.

For each task-state pair (u, s_t) , we denote the success and failure supports as $S_{u,s_t}^+ = N_u^+(s_t)$ and $S_{u,s_t}^- = N_u^-(s_t)$, with total support $S_{u,s_t} = S_{u,s_t}^+ + S_{u,s_t}^-$.

The gate is defined as

$$g_t = \mathbb{I}[S_{u,s_t} \geq S_{\min} \wedge S_{u,s_t}^+ \geq S_{\min}^{\text{each}} \wedge S_{u,s_t}^- \geq S_{\min}^{\text{each}} \wedge |\Delta_t| \geq \tau_\Delta], \quad (6)$$

where $S_{\min}$ is the total support threshold, $S_{\min}^{\text{each}}$ is the minimum support threshold for each pool, and τ_Δ is the preference-margin threshold. If $g_t = 0$, DEPPO falls back to the base advantage and does not apply experience-based reweighting.

When the gate is activated ($g_t = 1$), DEPPO constructs a trajectory-outcome-aware reweighting factor. Let

$$\phi_t^{(y_i)} = y_i \cdot \Delta_t, \quad (7)$$

where $y_i = +1$ for successful trajectories and $y_i = -1$ for failed trajectories.

The reweighted advantage is defined as

$$\tilde{A}_{i,t} = \text{clip}\left(\exp(\beta \cdot y_i \Delta_t), w_{\min}, w_{\max}\right)^{g_t} \cdot A_{i,t}, \quad (8)$$

where $A_{i,t}$ denotes the base step-level advantage computed using the state-aware group-based optimization strategy in GiGPO (Feng et al., 2026a), β controls the sharpness of reweighting, $w_{\min}$ and $w_{\max}$ bound the weight range, and $g_t = 0$ disables reweighting (i.e., weight = 1).

This matches the intuition in Fig. 2: successful trajectories amplify actions with $\Delta_t > 0$ and suppress those with $\Delta_t < 0$, while failed trajectories exhibit the opposite behavior.

3.4 Policy Optimization

DEPPO keeps the underlying policy optimization objective unchanged while introducing experience-guided advantage reweighting through the adaptive advantage $\tilde{A}_{i,t}$. With a PPO-style clipped objective, the training loss is

$$\mathcal{L}_{\text{DEPPO}} = -\mathbb{E} \left[\frac{1}{T} \sum_{t=1}^T \min \left(r_t(\theta) \tilde{A}_t, \text{clip} \left(r_t(\theta), 1 - \varepsilon_{\text{clip}}, 1 + \varepsilon_{\text{clip}} \right) \tilde{A}_t \right) \right], \quad (9)$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio, and $\varepsilon_{\text{clip}}$ is the PPO clipping coefficient.

DEPPO is an experience-driven policy optimization framework built upon GiGPO, without requiring an additional critic, process reward model, or auxiliary supervision. By leveraging historical successful and failed trajectories, DEPPO performs experience-guided advantage reweighting for adaptive step-level policy optimization.

3.5 Asymmetric Experience Update

After policy optimization, new trajectories are written back into the dual experience pools. The update is asymmetric: the success pool $\mathcal{D}^+$ emphasizes effective and fresh successful behaviors, while the failure pool $\mathcal{D}^-$ emphasizes repeated risky behaviors, invalid actions, and late-stage failure patterns.

Importantly, DEPPO queries the pools before writing the current batch into them. This prevents direct self-leakage, where a trajectory could influence its own reweighting score.

Freshness-aware success update. For a successful trajectory with $y_i = +1$, each task-conditioned state-action pair (u, s_t, a_t) is written into the success pool. To avoid making the success pool overly conservative, DEPPO assigns a larger insertion weight to actions that are successful but still underrepresented under the same task-state condition:

$$\psi_t^+ = \psi_0^+ (1 + \gamma_f \max(0, \eta - P^+(a_t | u, s_t))) , \quad (10)$$

where ψ_0^+ is the base success insertion weight, η is the freshness threshold, and γ_f controls the strength of freshness-aware insertion.

The success pool is updated as

$$c_u^+(s_t, a_t) \leftarrow c_u^+(s_t, a_t) + \psi_t^+ . \quad (11)$$

This mechanism allows $\mathcal{D}^+$ to preserve not only frequent successful actions, but also rare actions that may represent newly emerging effective strategies under the same task-state condition.

Repeated failure emphasis. For a failed trajectory with $y_i = -1$, DEPPO writes each task-conditioned state-action pair (u, s_t, a_t) into the failure pool with a risk-aware insertion weight:

$$\begin{aligned} \psi_t^- = \psi_0^- & \left(1 + \gamma_{\text{late}} \frac{t-1}{T_i-1} \right) \\ & \cdot (1 + \gamma_{\text{rep}} \log(1 + c_u^-(s_t, a_t))) \\ & \cdot (1 + \gamma_{\text{invalid}} \mathbb{I}[a_t = \text{INVALID}]) , \end{aligned} \quad (12)$$

where ψ_0^- is the base failure insertion weight. The late-stage term gives larger weights to actions closer to the final failure, the repeated-failure term emphasizes actions that have repeatedly appeared in failed trajectories under the same task-state condition, and the invalid-action term explicitly strengthens the memory of invalid decisions.

The failure pool is updated as

$$c_u^-(s_t, a_t) \leftarrow c_u^-(s_t, a_t) + \psi_t^- . \quad (13)$$

This update encourages $\mathcal{D}^-$ to concentrate on high-risk action patterns under the same task-state condition, rather than treating all failed actions equally.

3.6 Decay and Pruning

The experience pools should evolve together with the policy. Early trajectories generated by a weak policy may become outdated after policy improvement, and unbounded accumulation would increase

memory cost. Therefore, DEPPO periodically applies decay and pruning.

All task-conditioned state-action counts are exponentially decayed:

$$c_u^\pm(s, a) \leftarrow \lambda c_u^\pm(s, a), \quad \lambda \in (0, 1), \quad (14)$$

where λ is the decay rate. This gradually weakens stale experiences that are not reinforced by recent trajectories.

DEPPO then removes low-value action entries:

$$c_u^\pm(s, a) < \delta_a \Rightarrow \text{prune}(u, s, a), \quad (15)$$

where δ_a is the action-level pruning threshold. For each task-state pair (u, s) , only the top- K_a actions with the largest counts are retained. If the total task-state count becomes too small, the corresponding state entry under task u is removed:

$$\sum_a c_u^\pm(s, a) < \delta_s \Rightarrow \text{remove}(u, s), \quad (16)$$

where δ_s is the state-level pruning threshold.

To further bound memory size, DEPPO limits the maximum number of states under each task. When the number of states for a task exceeds the budget K_s , states with both low total counts and old update timestamps are removed first. This combines importance and recency, allowing the pools to remain compact, adaptive, and aligned with the evolving policy.

4 Experiment

4.1 Experimental Setup

Benchmarks. We evaluate DEPPO on representative sparse-reward long-horizon tasks: ALF-World (Shridhar et al., 2020), WebShop (Yao et al., 2022a). ALFWorld requires multi-step embodied textual reasoning, while WebShop evaluates web-based decision-making with reward determined by the final selected product; both provide limited intermediate feedback, making them suitable for credit assignment under sparse terminal rewards.

Baselines. We compare DEPPO with strong baselines on ALFWorld and WebShop under Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct backbones. For prompting-based methods, we include the vanilla Qwen2.5 policy, ReAct(Yao et al., 2022b), and Reflexion(Shinn et al., 2023). For trajectory-level reinforcement learning, we compare with PPO(Schulman et al., 2017), RLOO(Ahmadian et al., 2024), GRPO(Shao et al.,

Table 1: Performance comparison on ALFWorld and WebShop across prompting-based and RL-based methods under Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct settings. For ALFWorld, Avg. denotes the unweighted average over the six subtasks. Red numbers indicate absolute improvements relative to the corresponding Qwen2.5-Instruct baseline of the same model scale. Best and second-best results within each model block are highlighted in bold and underline, respectively.

Type	Method	ALFWorld							WebShop	
		Pick	Look	Clean	Heat	Cool	Pick2	Avg.	Score	Succ.
Qwen2.5-1.5B-Instruct										
Prompting	Qwen2.5	5.9	5.5	3.3	9.7	4.2	0.0	4.7	23.1	5.2
Prompting	ReAct	17.4 ^{↑11.5}	20.5 ^{↑15.0}	15.7 ^{↑12.4}	6.2 ^{↓3.5}	7.7 ^{↑3.5}	2.0 ^{↑2.0}	11.6 ^{↑6.9}	40.1 ^{↑17.0}	11.3 ^{↑6.1}
Prompting	Reflexion	35.3 ^{↑29.4}	22.2 ^{↑16.7}	21.7 ^{↑18.4}	13.6 ^{↑3.9}	19.4 ^{↑15.2}	3.7 ^{↑3.7}	19.3 ^{↑14.6}	55.8 ^{↑32.7}	21.9 ^{↑16.7}
RL Training	PPO	64.8 ^{↑58.9}	40.5 ^{↑35.0}	57.1 ^{↑53.8}	60.6 ^{↑50.9}	46.4 ^{↑42.2}	47.4 ^{↑47.4}	52.8 ^{↑48.1}	73.8 ^{↑50.7}	51.5 ^{↑46.3}
RL Training	RLOO	88.3 ^{↑82.4}	52.8 ^{↑47.3}	71.0 ^{↑67.7}	62.8 ^{↑53.1}	66.4 ^{↑62.2}	56.9 ^{↑56.9}	66.4 ^{↑61.7}	73.9 ^{↑50.8}	52.1 ^{↑46.9}
RL Training	GRPO	85.3 ^{↑79.4}	53.7 ^{↑48.2}	84.5 ^{↑81.2}	78.2 ^{↑68.5}	59.7 ^{↑55.5}	53.5 ^{↑53.5}	69.1 ^{↑64.4}	75.8 ^{↑52.7}	56.8 ^{↑51.6}
RL Training	DAPO	87.9 ^{↑82.0}	40.0 ^{↑34.5}	78.1 ^{↑74.8}	35.7 ^{↑26.0}	65.2 ^{↑61.0}	44.4 ^{↑44.4}	58.6 ^{↑53.9}	78.0 ^{↑54.9}	58.2 ^{↑53.0}
RL Training	EMPG	85.5 ^{↑79.6}	33.5 ^{↑28.0}	78.9 ^{↑75.6}	76.2 ^{↑66.5}	74.7 ^{↑70.5}	69.1 ^{↑69.1}	69.7 ^{↑65.0}	80.4 ^{↑57.3}	60.8 ^{↑55.6}
RL Training	RewardFlow	77.4 ^{↑71.5}	64.3 ^{↑58.8}	84.0 ^{↑80.7}	62.5 ^{↑52.8}	86.4 ^{↑82.2}	25.0 ^{↑25.0}	66.6 ^{↑61.9}	78.3 ^{↑55.2}	60.9 ^{↑55.7}
RL Training	GiGPO	96.0 ^{↑90.1}	76.5 ^{↑71.0}	91.8 ^{↑88.5}	91.3 ^{↑81.6}	71.7 ^{↑67.5}	79.5 ^{↑79.5}	84.5 ^{↑79.8}	83.1 ^{↑60.0}	65.0 ^{↑59.8}
RL Training	HCAPO	88.6 ^{↑82.7}	75.0 ^{↑69.5}	97.6 ^{↑94.3}	90.7 ^{↑81.0}	84.2 ^{↑80.0}	74.2 ^{↑74.2}	85.1 ^{↑80.4}	83.8 ^{↑60.7}	68.5 ^{↑63.3}
RL Training	DEPPO	96.8 ^{↑90.9}	82.9 ^{↑77.4}	97.3 ^{↑94.0}	98.7 ^{↑89.0}	94.1 ^{↑89.9}	83.6 ^{↑83.6}	92.2 ^{↑87.5}	86.5 ^{↑63.4}	73.1 ^{↑67.9}
Qwen2.5-7B-Instruct										
Prompting	Qwen2.5	33.4	21.6	19.3	6.9	2.8	3.2	14.5	26.4	7.8
Prompting	ReAct	48.5 ^{↑15.1}	35.4 ^{↑13.8}	34.3 ^{↑15.0}	13.2 ^{↑6.3}	18.2 ^{↑15.4}	17.6 ^{↑14.4}	27.9 ^{↑13.4}	46.2 ^{↑19.8}	19.5 ^{↑11.7}
Prompting	Reflexion	62.0 ^{↑28.6}	41.6 ^{↑20.0}	44.9 ^{↑25.6}	30.9 ^{↑24.0}	36.3 ^{↑33.5}	23.8 ^{↑20.6}	39.9 ^{↑25.4}	58.1 ^{↑31.7}	28.8 ^{↑21.0}
RL Training	PPO	92.3 ^{↑58.9}	64.0 ^{↑42.4}	92.5 ^{↑73.2}	89.5 ^{↑82.6}	80.3 ^{↑77.5}	68.8 ^{↑65.6}	81.2 ^{↑66.7}	81.4 ^{↑55.0}	68.7 ^{↑60.9}
RL Training	RLOO	87.6 ^{↑54.2}	78.2 ^{↑56.6}	87.3 ^{↑68.0}	81.3 ^{↑74.4}	71.9 ^{↑69.1}	48.9 ^{↑45.7}	75.9 ^{↑61.4}	80.3 ^{↑53.9}	65.7 ^{↑57.9}
RL Training	GRPO	90.8 ^{↑57.4}	66.1 ^{↑44.5}	89.3 ^{↑70.0}	74.7 ^{↑67.8}	72.5 ^{↑69.7}	64.7 ^{↑61.5}	76.4 ^{↑61.9}	79.3 ^{↑52.9}	66.1 ^{↑58.3}
RL Training	DAPO	91.4 ^{↑58.0}	72.0 ^{↑50.4}	79.5 ^{↑60.2}	82.7 ^{↑75.8}	75.4 ^{↑72.6}	51.7 ^{↑48.5}	75.5 ^{↑61.0}	80.2 ^{↑53.8}	67.2 ^{↑59.4}
RL Training	EMPG	92.9 ^{↑59.5}	75.2 ^{↑53.6}	74.8 ^{↑55.5}	86.3 ^{↑79.4}	73.7 ^{↑70.9}	65.3 ^{↑62.1}	78.0 ^{↑63.5}	81.0 ^{↑54.6}	69.3 ^{↑61.5}
RL Training	RewardFlow	94.6 ^{↑61.2}	70.0 ^{↑48.4}	90.0 ^{↑70.7}	86.4 ^{↑79.5}	70.0 ^{↑67.2}	70.0 ^{↑66.8}	80.2 ^{↑65.7}	84.4 ^{↑58.0}	73.4 ^{↑65.6}
RL Training	GiGPO	97.7 ^{↑64.3}	82.7 ^{↑61.1}	98.8 ^{↑79.5}	83.7 ^{↑76.8}	89.3 ^{↑86.5}	79.2 ^{↑76.0}	88.6 ^{↑74.1}	86.2 ^{↑59.8}	75.2 ^{↑67.4}
RL Training	HCAPO	99.1 ^{↑65.7}	90.3 ^{↑68.7}	97.3 ^{↑78.0}	81.8 ^{↑74.9}	90.8 ^{↑88.0}	81.9 ^{↑78.7}	90.2 ^{↑75.7}	85.1 ^{↑58.7}	73.8 ^{↑66.0}
RL Training	DEPPO	100.0 ^{↑66.6}	92.2 ^{↑70.6}	97.0 ^{↑77.7}	97.7 ^{↑90.8}	92.3 ^{↑89.5}	95.0 ^{↑91.8}	95.7 ^{↑81.8}	90.1 ^{↑63.7}	84.3 ^{↑76.5}

2024), and DAPO(Yu et al., 2025). To further evaluate the effectiveness of structured credit assignment in long-horizon sparse-reward settings, we include recent RL training methods such as EMPG(Wang et al., 2025) RewardFlow(Feng et al., 2026b), GiGPO(Feng et al., 2026a), and HCAPO(Tan et al., 2026).

Training Details. For fair comparison, we keep the rollout setup, optimization protocol, and evaluation procedure consistent with GiGPO. Detailed hyperparameters and implementation settings are reported in the Appendix F. Unless otherwise specified, all experimental results and analyses are reported on ALFWorld using Qwen2.5-1.5B-Instruct.

4.2 Main Results

Table 1 reports the main results under Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct backbones. DEPPO achieves the best overall performance in

most settings. With the 1.5B backbone, DEPPO improves the ALFWorld average from 84.5/85.1 to 92.2 over GiGPO/HCAPO, and obtains the best WebShop Score/Success of 86.5/73.1. With the 7B backbone, DEPPO further reaches 95.7 on ALFWorld and 90.1/84.3 on WebShop, exceeding GiGPO by 7.1 points on ALFWorld average and 9.1 points on WebShop Success. These results show that task-conditioned success–failure statistics provide effective step-level optimization signals for long-horizon sparse-reward learning.

4.3 Analysis

Computational and Memory Overhead. DEPPO stores only task–state–action statistics in CPU dictionaries and introduces no additional trainable modules. As shown in Table 2, most of the cost comes from pool lookup and update, while advantage reweighting requires only 0.04 ms per step.

Table 2: Runtime overhead of DEPPO’s experience-pool operations.

Operation	Measured runtime
Pool lookup	42.21 ms/step
Advantage reweighting	0.04 ms/step
Pool update	34.91 ms/step
Decay and pruning	79.55 ms/trigger
Amortized total overhead	85.12 ms/step
Total on a decay/pruning step	156.72 ms

Decay and pruning are triggered periodically rather than at every step. The resulting amortized overhead is 85.12 ms per step, corresponding to only 0.04–0.07% additional training time over GiGPO. The serialized pools occupy 1.02 MB, use approximately 3–8 MB of live CPU memory, and require no persistent GPU memory, confirming that DEPPO remains lightweight. Because experience management is decoupled from model computation, DEPPO can be applied across backbone scales without increasing parameter or activation memory.

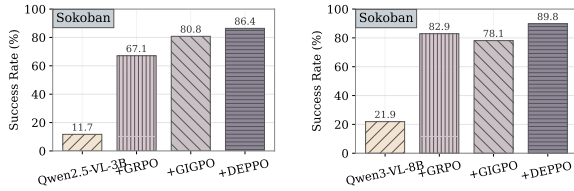


Figure 3: Comparison of the base model and different RL optimization methods on Sokoban.

Evaluation on Vision-Language Benchmarks.

We further evaluate DEPPO on the visual reasoning environment Sokoban(Culberson, 1997) under both Qwen2.5-VL-3B and Qwen3-VL-8B backbones. As shown in Figure 3, DEPPO consistently outperforms GRPO and GiGPO across different model scales, demonstrating that the proposed experience-driven policy optimization framework is also effective in vision-language decision-making tasks. The results suggest that DEPPO can generalize beyond text-only interactive agents and provide stable improvements in visually grounded sequential reasoning environments.

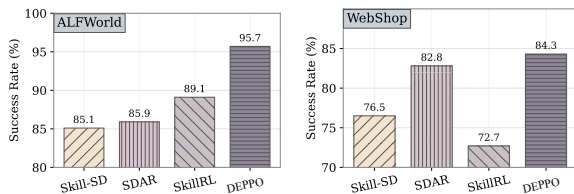


Figure 4: Comparison of skill-based agent mechanisms on ALFWorld and WebShop.

Comparison with Skill-based Mechanisms.

Figure 4 compares DEPPO with representative skill-based agent methods on ALFWorld and WebShop. Skill-SD(Wang et al., 2026a) and SDAR(Lu et al., 2026) leverage externally generated skill guidance during training, while SkillRL(Xia et al., 2026) further builds a dynamically evolving skill library for skill-augmented reinforcement learning. All external skills are generated using OpenAI o3 under the same training configuration.

DEPPO consistently achieves the best performance across both environments, indicating that merely injecting external skills or internalizing skills alone is insufficient for reliable long-horizon decision-making. In contrast, DEPPO explicitly models task-conditioned success–failure preferences over state-action patterns, enabling more effective utilization of historical experience during policy optimization.

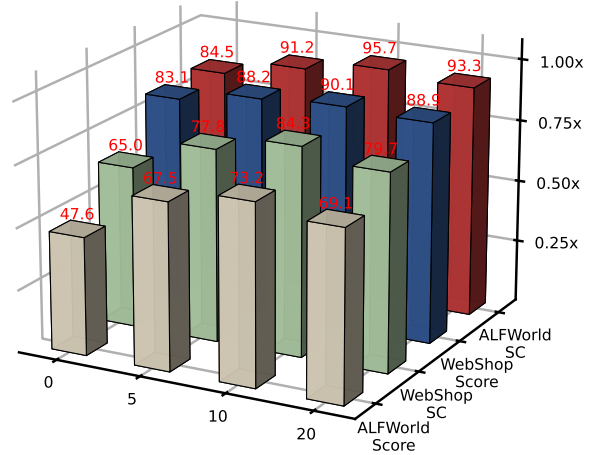


Figure 5: Sensitivity analysis of the reweighting sharpness β in DEPPO. The x-axis denotes scaling ratios relative to the default β setting. A moderate sharpness achieves the best trade-off between filtering weak experience signals and preserving informative success–failure preferences.

Sensitivity Analysis of the Reweighting Sharpness β . Figure 5 presents the sensitivity analysis of the reweighting sharpness parameter β in DEPPO. Excessively small β values lead to weak experience discrimination, while overly large values may over-amplify success–failure preferences and reduce policy flexibility. A moderate sharpness achieves the best overall performance across both ALFWorld and WebShop, highlighting the importance of balanced advantage reweighting.

Experience Pool Dynamics Analysis. Figures 6 and 7 show how the dual experience pools evolve during training. As training progresses, the

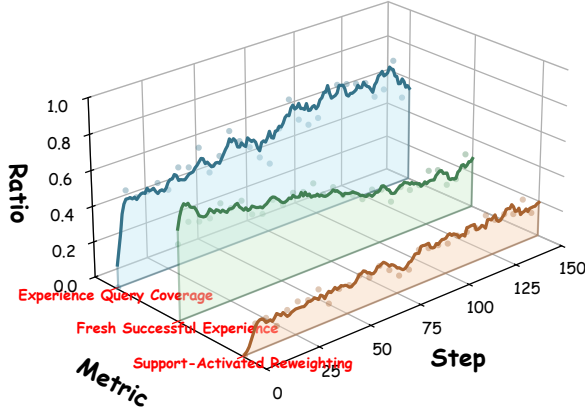


Figure 6: Training dynamics of DEPPO during reinforcement learning. The figure illustrates the evolution of experience query coverage, fresh successful experience ratio, and support-activated reweighting ratio over training steps.

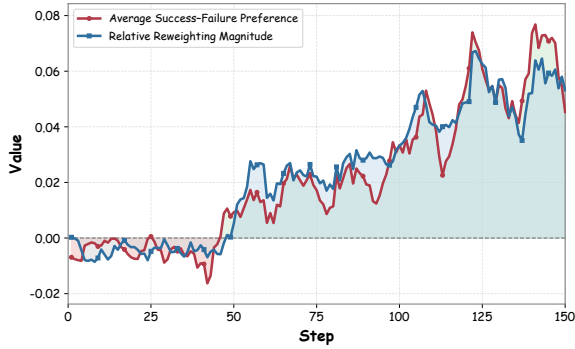


Figure 7: Evolution of the average success–failure preference and the relative advantage reweighting magnitude during DEPPO training. As the experience pools become more informative, the preference signal gradually strengthens and induces larger step-level advantage reweighting.

experience query coverage and support-activated reweighting ratio both steadily increase, indicating that the pools gradually accumulate sufficient historical evidence for reliable advantage reweighting. Meanwhile, the fresh successful experience ratio decreases and then stabilizes, suggesting that the positive pool first absorbs many newly discovered successful behaviors and later converges to a more mature set of reusable success patterns. The dynamics further reveal a stage-wise transition: early training is mainly guided by the negative pool, which suppresses high-risk actions from frequent failures, while later training increasingly relies on the positive pool to reinforce stable successful behaviors. This demonstrates that the dual pools gradually evolve from risk-control memories into positive guidance structures for long-horizon policy optimization.

Table 3: Ablation study of DEPPO components on ALFWorld and WebShop under different model scales.

Variant	ALFWorld		WebShop	
	SC	Score	SC	Score
Qwen2.5-1.5B with GiGPO	84.5	47.6	65.0	83.1
+ Positive experience pool	90.4	58.3	69.7	85.2
+ Negative experience pool	91.3	64.4	71.9	85.8
+ Dynamic pool update	92.2	68.0	73.1	86.5
Qwen2.5-7B with GiGPO	88.6	55.7	75.2	86.2
+ Positive experience pool	93.5	68.3	79.8	88.5
+ Negative experience pool	95.1	71.1	82.7	89.6
+ Dynamic pool update	95.7	73.2	84.3	90.1

5 Ablation Study

Table 3 presents the ablation study of different DEPPO components under both 1.5B and 7B model scales. Adding the positive experience pool consistently improves performance over the vanilla GiGPO baseline, showing that successful historical trajectories provide useful reusable decision patterns. Incorporating the negative experience pool further brings stable gains, indicating that suppressing high-risk or failure-prone actions is equally important for long-horizon optimization. Finally, dynamic pool update achieves the best overall performance across both ALFWorld and WebShop, demonstrating that continuously evolving experience statistics are critical for maintaining effective and adaptive step-level credit assignment during reinforcement learning. These gains remain consistent across both evaluated backbone scales and environments.

6 Conclusion

We propose DEPPO, a critic-free and experience-driven policy optimization framework for long-horizon reinforcement learning. DEPPO transforms historical trajectories into dual experience pools that model task-conditioned success and failure statistics, enabling more reliable step-level credit assignment under sparse terminal rewards. By dynamically emphasizing effective behaviors and repeated risky patterns, DEPPO provides adaptive optimization signals without requiring additional critics, process reward models, or auxiliary supervision. Experiments on ALFWorld, WebShop, and Sokoban show that DEPPO consistently improves policy optimization across different model scales and decision-making environments.

Limitations

DEPPO relies on historical successful and failed trajectories to estimate adaptive advantage reweighting signals. Therefore, its effectiveness depends on the diversity and coverage of the maintained experience pools, especially in highly open-ended environments with sparse successful trajectories.

In addition, our experiments mainly focus on long-horizon agent and reasoning benchmarks. Further evaluation on broader real-world settings and larger-scale foundation models would help better understand the scalability and generalization of DEPPO.

Ethics Statement

DEPPO is designed for research on long-horizon reinforcement learning and step-level credit assignment. Our method does not involve human subjects, private user data, or sensitive personal information. The experiments are conducted on publicly available benchmarks and open-source models. Like other RL methods for large language models, DEPPO may amplify biased or suboptimal behaviors if the reward signals are poorly designed. Therefore, careful reward design and evaluation are important for real-world deployment.

References

- Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. 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)*, pages 12248–12267.
- Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. 2025. Reinforcement learning for long-horizon interactive llm agents. *arXiv preprint arXiv:2502.01600*.
- Joseph Culberson. 1997. Sokoban is pspace-complete.
- Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2026a. Group-in-group policy optimization for llm agent training. *Advances in Neural Information Processing Systems*, 38:46375–46408.
- Xiao Feng, Bo Han, Zhanke Zhou, Jiaqi Fan, Jiangchao Yao, Ka Ho Li, Dahai Yu, and Michael Kwok-Po Ng. 2026b. Rewardflow: Topology-aware reward propagation on state graphs for agentic rl with large language models. *arXiv preprint arXiv:2603.18859*.
- Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, and 1 others. 2025. Areal: A large-scale asynchronous reinforcement learning system for language reasoning. *arXiv preprint arXiv:2505.24298*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025a. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Jiaxin Guo, Zewen Chi, Li Dong, Qingxiu Dong, Xun Wu, Shaohan Huang, and Furu Wei. 2025b. Reward reasoning model. *arXiv preprint arXiv:2505.14674*.
- Mingqian He, Yongliang Shen, Wenqi Zhang, Zeqi Tan, and Weiming Lu. 2024. Advancing process verification for large language models via tree-based preference learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2086–2099.
- Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.
- Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. 2025. Process reward models that think. *arXiv preprint arXiv:2504.16828*.
- Barrett Martin Lattimer, Varun Prashant Gangal, Ryan McDonald, and Yi Yang. 2025. Sparse rewards can self-train dialogue agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 25395–25413.
- Jiazheng Li, Yawei Wang, Qiaojing Yan, Yijun Tian, Zhichao Xu, Huan Song, Panpan Xu, and Lin Lee Cheong. 2026. Salt: Step-level advantage assignment for long-horizon agents via trajectory graph. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 4709–4725.
- Wendi Li and Yixuan Li. 2024. Process reward model with q-value rankings. *arXiv preprint arXiv:2410.11287*.
- Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and 1 others. 2026. Self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2605.15155*.
- Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, and 1 others. 2024. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*.

- Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PmLR.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Shuaijie She, Junxiao Liu, Yifeng Liu, Jiajun Chen, Xin Huang, and Shujian Huang. 2025. R-prm: Reasoning-driven process reward modeling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 13438–13451.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652.
- Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.
- Hui-Ze Tan, Xiao-Wen Yang, Hao Chen, Jie-Jing Shao, Yi Wen, Yuteng Shen, Weihong Luo, Xiku Du, Lan-Zhe Guo, and Yu-Feng Li. 2026. Hindsight credit assignment for long-horizon llm agents. *arXiv preprint arXiv:2603.08754*.
- Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.
- Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, and 1 others. 2026a. Skill-sd: Skill-conditioned self-distillation for multi-turn llm agents. *arXiv preprint arXiv:2604.10674*.
- Jiawei Wang, Jiakai Liu, Yuqian Fu, Yingru Li, Xintao Wang, Yuan Lin, Yu Yue, Lin Zhang, Yang Wang, and Ke Wang. 2025. Harnessing uncertainty: Entropy-modulated policy gradients for long-horizon llm agents. *arXiv preprint arXiv:2509.09265*.
- Jichao Wang, Liuyang Bian, Yufeng Zhou, Han Xiao, Yue Pan, Guozhi Wang, Hao Wang, Zhaoxiong Wang, Yafei Wen, Xiaoxin Chen, and 1 others. 2026b. Solar-rl: Semi-online long-horizon assignment reinforcement learning. *arXiv preprint arXiv:2604.22558*.
- Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.
- Tao Wang, Suhang Zheng, and Xiaoxiao Xu. 2026c. Rtmc: Step-level credit assignment via rollout trees. *arXiv preprint arXiv:2604.11037*.
- Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, and 1 others. 2026d. Reinforcement learning for reasoning in large language models with one training example. *Advances in Neural Information Processing Systems*, 38:122721–122764.
- Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Honglin Guo, Jiaqi Liu, Rui Zheng, Junjie Ye, Jiazhang Zhang, Wenxiang Chen, and 1 others. 2025. Agentgym-rl: Training llm agents for long-horizon decision making through multi-turn reinforcement learning. *arXiv preprint arXiv:2509.08755*.
- Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, and 1 others. 2026. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*.
- Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022a. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022b. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.
- Kaiyan Zhang, Yuxin Zuo, Bingxiang He, Youbang Sun, Runze Liu, Che Jiang, Yuchen Fan, Kai Tian, Guoli Jia, Pengfei Li, and 1 others. 2025. A survey of reinforcement learning for large reasoning models. *arXiv preprint arXiv:2509.08827*.
- Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, and 1 others. 2025. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*.

Appendix

A Experience Pool Dynamics

To further analyze the behavior of DEPPO, we track the evolution of the positive and negative experience pools on ALFWorld under Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct. Figure 8 shows the task-level coverage of the dual pools. Overall, the number of task entries increases as training proceeds. For the 1.5B model, the negative pool grows faster in the early stage, reflecting that the weaker policy produces more failed trajectories at the beginning. As the policy improves, the positive pool gradually increases and becomes comparable to the negative pool. For the 7B model, the positive pool grows more steadily, suggesting that the stronger backbone can accumulate successful task-level experiences more effectively.

Figure 9 presents the state-action entry dynamics. The negative pool, especially under the 1.5B backbone, contains more state-action entries, indicating that failed trajectories cover more diverse and unstable decision states. In contrast, the positive pool grows more smoothly and records relatively stable successful patterns. These results show that DEPPO benefits from both successful and failed experiences: the positive pool provides reusable behavior guidance, while the negative pool helps identify and suppress risky actions during step-level credit assignment.

B Hyperparameter Sensitivity

Sensitivity of Filtering Coverage. We first study how the preference threshold τ_Δ and the clipping range affect the gated ratio, i.e., the fraction of training steps receiving experience-guided reweighting.

As shown in Table 4, increasing the preference threshold reduces the gated ratio, while a wider scaling range yields slightly greater shaping coverage. These results show that the filtering mecha-

Table 4: Effect of filtering thresholds and scaling ranges on the gated ratio.

τ_Δ	Scale Range	Gated Ratio
0.03	[0.5, 1.5]	0.23
0.05	[0.5, 1.5]	0.18
0.10	[0.5, 1.5]	0.09
0.03	[0.2, 1.8]	0.26
0.05	[0.2, 1.8]	0.21
0.10	[0.2, 1.8]	0.11

nism provides direct control over the proportion of samples affected by historical experience.

Table 5: Performance sensitivity of DEPPO on ALFWorld.

Parameter	Setting	Avg.
Smoothing α	0.05	90.7
	0.1	92.2
	0.2	91.3
Support $S_{\min}$	4	89.2
	8	92.2
	16	90.7
Clipping range	[0.5, 1.5]	90.5
	[0.2, 1.8]	92.2
	[0.1, 2.0]	91.4

Performance Sensitivity. Table 5 shows that DEPPO remains stable across different smoothing coefficients, support thresholds, and clipping ranges. Moderate settings perform best: overly small values can introduce unreliable reweighting, whereas overly large values may reduce experience coverage or amplify noisy preferences. The default configuration $\alpha = 0.1$, $S_{\min} = 8$, and $[w_{\min}, w_{\max}] = [0.2, 1.8]$ achieves the best overall trade-off.

C Pruned Action Statistics

To further analyze the memory efficiency and dynamic evolution of DEPPO, we examine the number of pruned action entries during ALFWorld training. Following the decay and pruning strategy described in Section 3.6, low-frequency or outdated state-action entries are periodically removed from the dual experience pools to maintain compact and adaptive memory structures.

As shown in Table 6, the number of pruned actions is relatively high during early training and gradually decreases as training progresses. This trend suggests that the experience pools initially absorb a large amount of unstable exploration behavior, while later training stages become increasingly dominated by reusable and stable action patterns.

We also observe a substantial difference between model scales. The Qwen2.5-1.5B model consistently produces significantly more pruned actions than Qwen2.5-7B, especially during the early training stage. For example, at training step 40, the 1.5B model prunes 9320 action entries, whereas the 7B model prunes only 531. This phenomenon indicates that smaller models generate noisier and less stable trajectories, leading to more transient

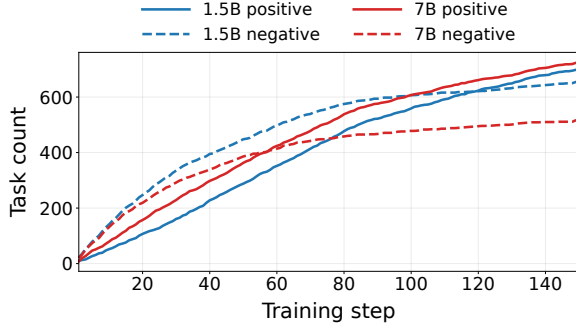


Figure 8: Evolution of task coverage in the dual experience pools during ALFWorld training under different model scales.

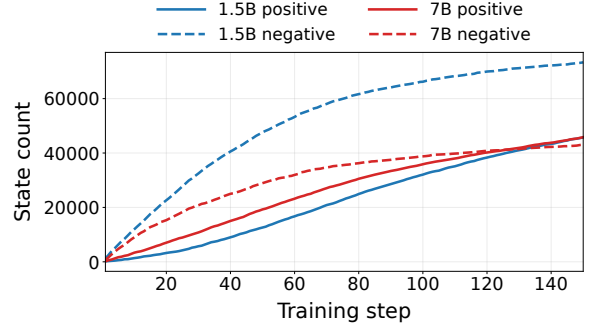


Figure 9: Evolution of state-action statistics in the dual experience pools during ALFWorld training under different model scales.

Table 6: Pruned action count every 10 training steps during the first 150 training steps on ALFWorld.

Step	Qwen2.5-1.5B	Qwen2.5-7B
10	5956	525
20	8402	693
30	7955	706
40	9320	531
50	7826	538
60	5567	421
70	3737	424
80	2887	237
90	2322	234
100	1301	201
110	1164	200
120	893	213
130	612	189
140	634	154
150	514	185

state-action statistics that are eventually removed by the pruning mechanism. In contrast, larger models maintain more stable experience structures and require less aggressive cleanup.

D Additional Training Dynamics

To further analyze the optimization behavior of DEPPO on long-horizon interactive tasks, we report additional training dynamics on ALFWorld, including subtask-level validation success rates, response length evolution, and entropy regularization dynamics. All figures are generated from smoothed validation statistics during reinforcement learning training.

Figure 10 presents the validation success rates for different ALFWorld subtasks under GRPO, GiGPO, and DEPPO. Across nearly all subtasks, DEPPO achieves faster convergence and consistently higher final performance. The improvement is especially significant on long-horizon subtasks such as *Heat*, *Cool*, and *Pick2*, where effective

step-level credit assignment is particularly important. Compared with GRPO and GiGPO, DEPPO exhibits both stronger learning efficiency during early training and more stable convergence behavior in later stages.

Figure 11 further reports the evolution of average response length and entropy loss during training. DEPPO maintains relatively stable response lengths while gradually reducing entropy loss, indicating that the policy becomes increasingly confident and structured during optimization. Compared with GRPO, DEPPO avoids excessively long and unstable trajectories, suggesting that experience-guided advantage reweighting helps suppress inefficient exploration behaviors. Meanwhile, the entropy dynamics remain smoother than GiGPO in later training stages, demonstrating improved optimization stability under sparse terminal rewards.

E Additional Component Analysis

E.1 Effect of Support-Aware Filtering

To evaluate the importance of the support-aware filtering mechanism in DEPPO, we compare training dynamics and final performance with and without threshold-based filtering. The filtering mechanism selectively applies advantage reweighting only when the queried state-action statistics satisfy sufficient support and preference confidence conditions. This design aims to suppress noisy or unreliable reweighting signals during early-stage exploration.

Figure 13 presents several internal shaping statistics during training, including the gated ratio, mean reweighting scale, mean preference delta, and mean absolute delta. Without filtering, the gated ratio rapidly increases to a much higher level, indicating that advantage reweighting is applied to a sub-

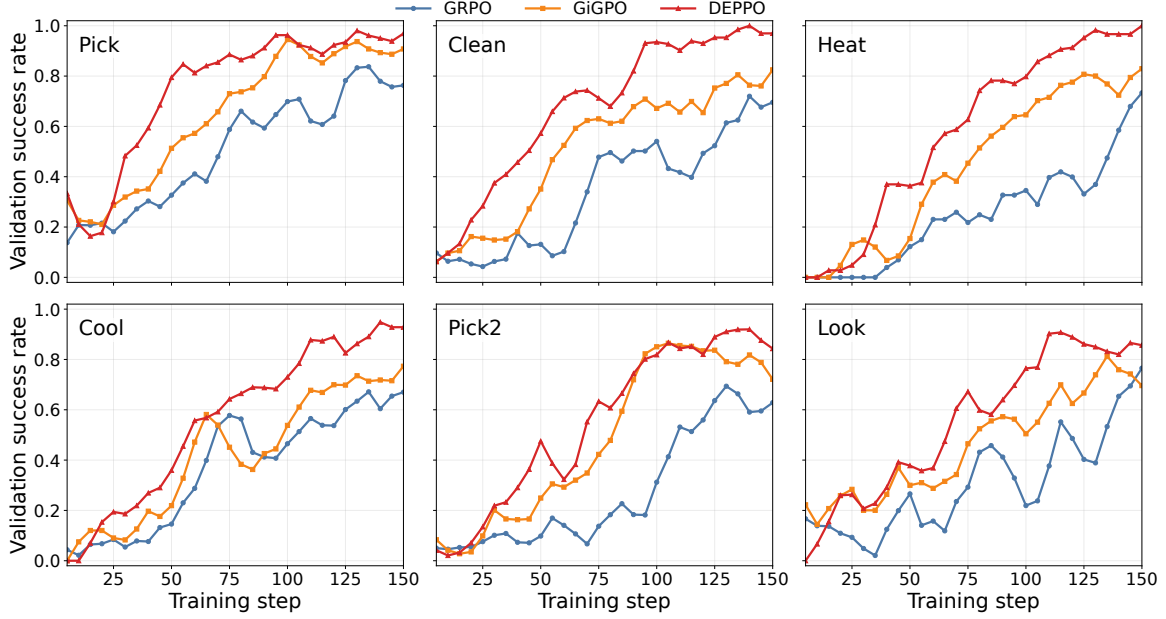


Figure 10: Validation success-rate dynamics on different ALFWorld subtasks during reinforcement learning training. DEPPPO consistently achieves faster convergence and higher final success rates across most subtasks.

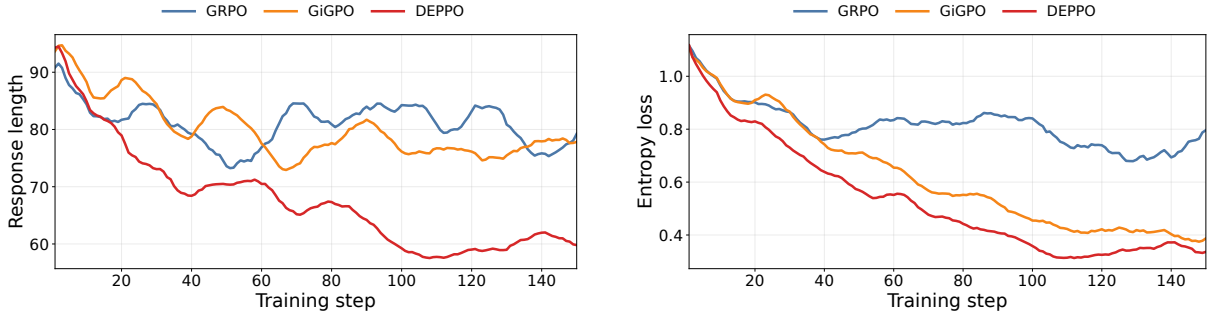


Figure 11: Training dynamics of response length (left) and entropy loss (right) on ALFWorld. DEPPPO exhibits more stable trajectory generation and smoother optimization behavior compared with GRPO and GiGPO.

stantially larger fraction of training samples. This behavior also leads to noticeably larger scaling factors and more aggressive preference updates. In contrast, the filtering mechanism maintains a more conservative and stable shaping process, preventing unreliable low-support experience statistics from dominating policy optimization.

Interestingly, although the unfiltered variant produces larger average scaling values, its preference statistics become substantially noisier throughout training. The filtered variant instead maintains smoother and more consistent delta dynamics, suggesting that support-aware gating effectively stabilizes experience-guided credit assignment under sparse terminal rewards.

Figure 12 further compares the validation success ratio. Despite applying reweighting to fewer training samples, the filtered variant consistently

achieves higher final success rates and more stable convergence behavior. This result demonstrates that selectively applying experience-guided shaping based on sufficient support is critical for preventing noisy reinforcement signals and improving long-horizon reinforcement learning stability.

F Training Hyperparameters

Unless otherwise specified, all experiments use a unified training configuration. We initialize the policy from the corresponding backbone model, including Qwen2.5-1.5B-Instruct, Qwen2.5-7B-Instruct, Qwen2.5-VL-3B, and Qwen3-VL-8B. All baselines and DEPPPO variants share the same roll-out protocol, optimization settings, and evaluation procedures for fair comparison.

For policy optimization, we use a learning rate of 1×10^{-6} , a PPO mini-batch size of 256, and

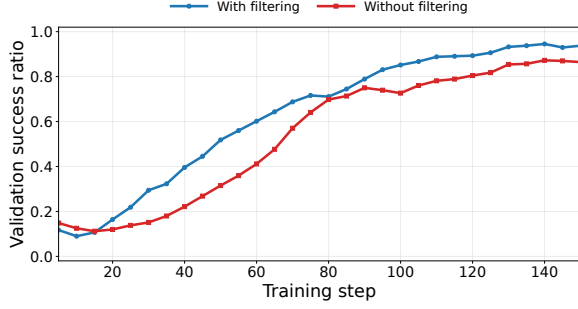


Figure 12: Validation success ratio comparison with and without support-aware filtering. Threshold-based filtering significantly improves optimization stability and final task performance.

Table 7: Key hyperparameters used in DEPPO.

Hyperparameter	Value
Learning rate	1×10^{-6}
Training batch size	16
Rollouts per prompt	8
Maximum response length	512
Preference threshold τ_{Δ}	0.03
Reweighting range	[0.2, 1.8]
Decay rate λ	0.995
Top- k actions	32
Number of GPUs	8
Training epochs	150

a KL coefficient of 0.01. Each prompt generates 8 rollouts with a training batch size of 16. The maximum prompt and response lengths are set to 4096 and 512, respectively. All experiments are trained on 8 GPUs for 150 epochs.

For DEPPO, the smoothing coefficient α is set to 0.1, the preference threshold τ_{Δ} is 0.03, and the minimum support threshold is 8. The advantage reweighting range is clipped to [0.2, 1.8]. We use a decay rate of 0.995 for memory updates and retain at most 32 actions per state. Table 7 summarizes several key hyperparameters.

G Benchmark Suite and Environment Statistics

This section presents the benchmark environments used for empirical evaluation. Our experiments focus exclusively on interactive long-horizon agent tasks with sparse terminal rewards. To comprehensively evaluate sequential decision-making and credit assignment ability, we consider three representative environments: ALFWorld, WebShop, and Sokoban. These benchmarks cover embodied household interaction, web-based decision making, and symbolic planning, respectively, providing

complementary challenges for long-horizon reinforcement learning.

Interactive agent environments. All selected environments require sequential decision making under sparse terminal rewards, making them suitable for evaluating long-horizon credit assignment.

ALFWorld is an embodied household interaction benchmark derived from TextWorld and ALFRED/THOR. The agent completes multi-step tasks such as object manipulation, transportation, heating, and cleaning through textual actions. Rewards are mainly provided upon successful task completion, creating a challenging sparse-reward setting for long-horizon planning and reasoning.

WebShop is a web-based decision-making environment built from a large-scale online product database. Given a shopping instruction, the agent must search, browse, compare products, and select a final item through sequential interactions. Because rewards depend primarily on the final purchase outcome, WebShop serves as a representative sparse-reward benchmark for web navigation and retrieval-based decision making.

Sokoban is a symbolic planning environment where the agent pushes boxes to target locations in a grid world while avoiding irreversible deadlock states. Despite its simple action space, Sokoban requires long-term planning and precise state transitions under highly delayed rewards, making it suitable for evaluating long-horizon reinforcement learning.

H Detailed Algorithms of DEPPO

This appendix presents the detailed training and memory-update procedures of DEPPO. DEPPO performs experience-guided step-level credit assignment by maintaining two task-conditioned experience pools that separately model successful and failed historical behaviors. During training, the current policy first queries the dual pools to estimate the relative success–failure preference of each action under the same task-state condition, and then adaptively reweights the advantage used for policy optimization.

The complete framework consists of two coupled procedures. Algorithm 1 describes the overall training pipeline, including trajectory rollout, dual-pool query, support-aware advantage reweighting, and policy optimization. Algorithm 2 presents the asymmetric experience update mechanism, including freshness-aware successful insertion, repeated-

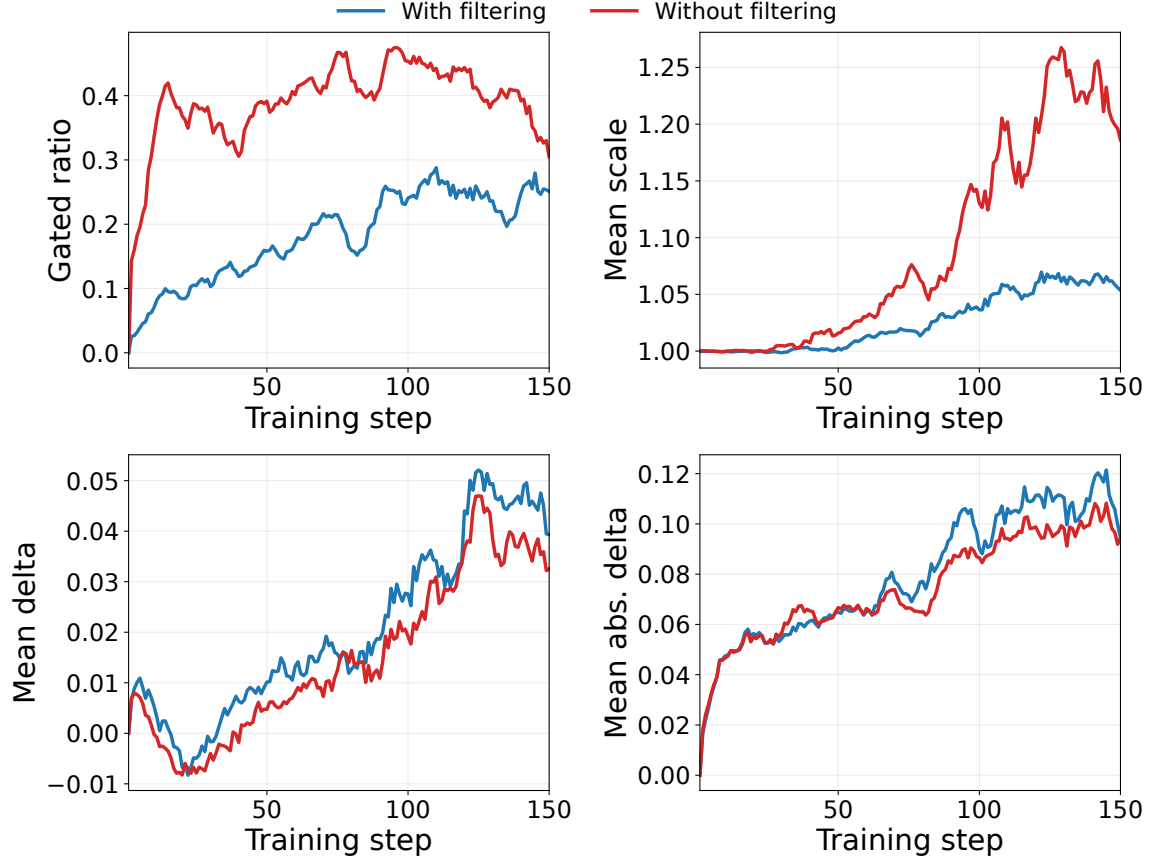


Figure 13: Comparison of internal shaping statistics with and without support-aware filtering. The filtering mechanism produces more stable gating ratios, reweighting scales, and preference dynamics during training.

failure emphasis, exponential decay, and memory pruning.

Importantly, DEPPO always queries the experience pools before inserting the current batch, preventing self-leakage between experience construction and advantage estimation. Together, the two procedures form a closed-loop learning process in which historical trajectories continuously provide structured success–failure supervision for long-horizon reinforcement learning.

Algorithm 1 DEPPO Training with Experience-Guided Advantage reweighting

Input: Policy π_θ , success pool $\mathcal{D}^+$, failure pool $\mathcal{D}^-$, smoothing coefficient α , stability constant ϵ , sharpness β , support thresholds $S_{\min}$, $S_{\min}^{\text{each}}$, preference threshold τ_Δ , weight bounds $w_{\min}$, $w_{\max}$.

Output: Optimized policy π_θ .

- 1: Initialize $\mathcal{D}^+ \leftarrow \emptyset$, $\mathcal{D}^- \leftarrow \emptyset$.
- 2: **for** each training iteration **do**
- 3: Sample tasks $\{u_i\}_{i=1}^K$ and roll out trajectories $\{\tau_i\}_{i=1}^K$ with π_θ .
- 4: Obtain terminal outcome labels $y_i \in \{+1, -1\}$ from trajectory-level rewards.
- 5: Compute base advantages $A_{i,t}$ using the state-aware group-based optimization strategy in GiGPO.
- 6: **for** each trajectory $\tau_i = \{(s_{i,t}, a_{i,t})\}_{t=1}^{T_i}$ **do**
- 7: **for** each step t **do**
- 8: Query $\mathcal{D}^+$ and $\mathcal{D}^-$ using the task-state key $(u_i, s_{i,t})$.
- 9: Construct action support set:

$$\mathcal{A}_{u_i, s_{i,t}} = \{a : c_{u_i}^+(s_{i,t}, a) > 0\} \cup \{a : c_{u_i}^-(s_{i,t}, a) > 0\} \cup \{a_{i,t}\}.$$

- 10: Estimate smoothed success/failure probabilities:

$$P^\pm(a_{i,t} \mid u_i, s_{i,t}) = \frac{c_{u_i}^\pm(s_{i,t}, a_{i,t}) + \alpha}{N_{u_i}^\pm(s_{i,t}) + \alpha |\mathcal{A}_{u_i, s_{i,t}}|}.$$

- 11: Compute success–failure preference:

$$\Delta_{i,t} = \log \frac{P^+(a_{i,t} \mid u_i, s_{i,t}) + \epsilon}{P^-(a_{i,t} \mid u_i, s_{i,t}) + \epsilon}.$$

- 12: Compute support-aware gate:

$$g_{i,t} = \mathbb{I} \left[S_{u_i, s_{i,t}} \geq S_{\min} \wedge S_{u_i, s_{i,t}}^+ \geq S_{\min}^{\text{each}} \wedge S_{u_i, s_{i,t}}^- \geq S_{\min}^{\text{each}} \wedge |\Delta_{i,t}| \geq \tau_\Delta \right].$$

- 13: Compute reweighting coefficient:

$$w_{i,t} = \text{clip}(\exp(\beta y_i \Delta_{i,t}), w_{\min}, w_{\max})^{g_{i,t}}.$$

- 14: Reweight the advantage:

$$\tilde{A}_{i,t} = w_{i,t} A_{i,t}.$$

- 15: **end for**
 - 16: **end for**
 - 17: Update π_θ with PPO-style optimization using $\tilde{A}_{i,t}$.
 - 18: Update $\mathcal{D}^+$ and $\mathcal{D}^-$ using Algorithm 2.
 - 19: **end for**
 - 20: **return** π_θ .
-

Algorithm 2 Asymmetric Experience Pool Update in DEPPO

Input: Trajectories $\{\tau_i\}_{i=1}^K$, labels $\{y_i\}_{i=1}^K$, pools $\mathcal{D}^+, \mathcal{D}^-$, decay rate λ , pruning thresholds δ_a, δ_s , action budget K_a , state budget K_s .

Output: Updated pools $\mathcal{D}^+$ and $\mathcal{D}^-$.

1: **for** each trajectory $\tau_i = \{(s_{i,t}, a_{i,t})\}_{t=1}^{T_i}$ with task u_i **do**

2: **for** each step t **do**

3: **if** $y_i = +1$ **then**

4: Compute freshness-aware insertion weight:

$$\psi_{i,t}^+ = \psi_0^+ (1 + \gamma_f \max(0, \eta - P^+(a_{i,t} \mid u_i, s_{i,t}))) .$$

5: Update success pool:

$$c_{u_i}^+(s_{i,t}, a_{i,t}) \leftarrow c_{u_i}^+(s_{i,t}, a_{i,t}) + \psi_{i,t}^+ .$$

6: **else**

7: Compute risk-aware insertion weight:

$$\psi_{i,t}^- = \psi_0^- \left(1 + \gamma_{\text{late}} \frac{t-1}{T_i-1} \right) (1 + \gamma_{\text{rep}} \log(1 + c_{u_i}^-(s_{i,t}, a_{i,t}))) (1 + \gamma_{\text{invalid}} \mathbb{I}[a_{i,t} = \text{INVALID}]) .$$

8: Update failure pool:

$$c_{u_i}^-(s_{i,t}, a_{i,t}) \leftarrow c_{u_i}^-(s_{i,t}, a_{i,t}) + \psi_{i,t}^- .$$

9: **end if**

10: **end for**

11: **end for**

12: **for** each pool $\mathcal{D} \in \{\mathcal{D}^+, \mathcal{D}^-\}$ **do**

13: **for** each task-state-action entry (u, s, a) in $\mathcal{D}$ **do**

14: Apply exponential decay:

$$c_u(s, a) \leftarrow \lambda c_u(s, a) .$$

15: **if** $c_u(s, a) < \delta_a$ **then**

16: Remove action entry (u, s, a) .

17: **end if**

18: **end for**

19: **for** each task-state pair (u, s) **do**

20: Retain only the top- K_a actions with the largest counts.

21: **if** $\sum_a c_u(s, a) < \delta_s$ **then**

22: Remove state entry (u, s) .

23: **end if**

24: **end for**

25: **for** each task u **do**

26: **if** the number of states under u exceeds K_s **then**

27: Remove states with low total counts and old update timestamps first.

28: **end if**

29: **end for**

30: **end for**

31: **return** $\mathcal{D}^+, \mathcal{D}^-$.
