SERO: LEARNING FROM EXECUTABLE ORACLE CONTINUATIONS FOR LONG-HORIZON AGENTIC REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon agentic RL is constrained by the futures visited by the current policy. When rollouts fail to reach useful behaviors, improved credit assignment cannot provide supervision for states and actions that were never encountered. On-policy distillation (OPD) introduces teacher guidance on policy-induced states, and recent multi-turn variants improve how this guidance is scheduled, replayed, refined, or validated. However, they do not directly use the oracle-controlled future itself as the training trajectory. We introduce **State-Rooted Executable Rollout Optimization (SERO)**, which augments on-policy Agentic RL with executable oracle continuations. SERO restores a state actually reached by the current policy and lets a training-only oracle continue interacting with the real environment, making the downstream observations and oracle actions themselves available for learning. It then applies **policy-reachability weighting**, which weights each oracle step by the frozen behavior policy's compatibility with the preceding oracle actions. This allows deeper supervision to become effective as the policy becomes more compatible with the oracle branch. The oracle loss is optimized separately from the base RL estimator and no oracle is used at inference time. We evaluate SERO on two challenging interactive-agent benchmarks, ALFWorld and WebShop, using Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct. On ALFWorld, learning the full executable continuation is substantially more effective than supervising only the root oracle action, while policy-reachability weighting further improves out-of-distribution performance.

1 INTRODUCTION

Large language models and vision-language models are increasingly used as the decision-making backbone of interactive agents for embodied control, web navigation, tool use, and software tasks (Yao et al., 2022b; Shinn et al., 2023; Schick et al., 2023; Brohan et al., 2023; Wang et al., 2023; Xie et al., 2024; Hong et al., 2024; Gur et al., 2024; Gou et al., 2025; Wei et al., 2026). Unlike single-turn reasoning, an agent must repeatedly observe the environment, choose an action, receive a new observation, and continue until the task is finished. This sequential decision-making setting provides a natural opportunity for reinforcement learning (RL), where policies can be improved directly from environment feedback (Sutton & Barto, 2018; Schulman et al., 2017; Ouyang et al., 2022; Guo et al., 2025; Tan et al., 2024; Zhai et al., 2024; Bai et al., 2024). In particular, critic-free group-based methods such as RLOO (Ahmadian et al., 2024), GRPO (Shao et al., 2024) , and their recent variants are attractive for large-scale agent training because they avoid training a separate value model and reduce the overhead associated with critic-based optimization (Yu et al., 2026; Liu et al., 2025; Lin et al., 2026; Zheng et al., 2025).

As agents operate over increasingly long interaction horizons, a central difficulty in long-horizon agent training is credit assignment: an episode may contain many decisions, while reward is often sparse or delayed (Peng et al., 2019; Wen et al., 2024; Chen et al., 2025; Song et al., 2024; Wang et al., 2025; Jin et al., 2025). Recent work therefore seeks more precise feedback for the trajectories collected by the current policy (Feng et al., 2026; He et al., 2026; Cheng et al., 2026). These methods substantially improve how sampled experience is credited. However, the learning signal is still bounded by what appears in the rollout group. If every rollout misses a useful action sequence,

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.

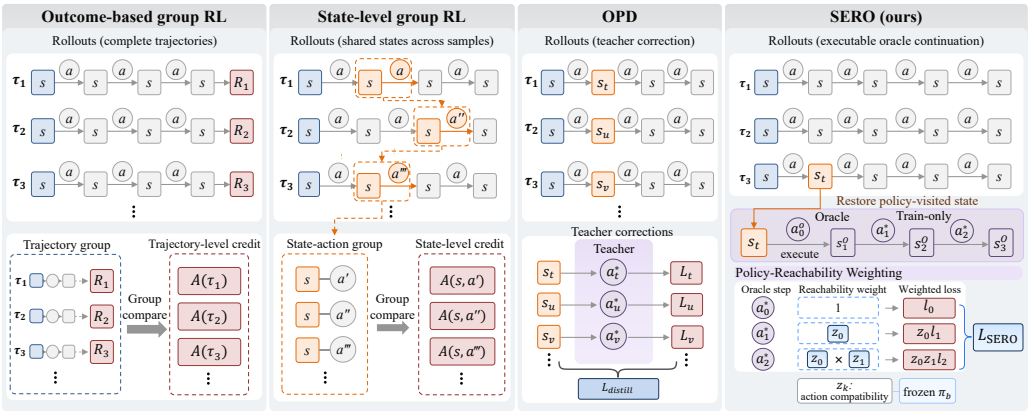


Figure 1: Comparison of training paradigms and learning signals. Outcome-based group RL assigns trajectory-level credit, state-level group RL compares sampled actions at shared states, and OPD distills a single corrective action. In contrast, SERO restores a state reached by the current policy, executes a multi-step oracle continuation, and applies policy-reachability weighting to downstream action losses.

finer attribution can rank the sampled behaviors more accurately, but it cannot provide the missing behavior as a positive target. This limitation is particularly important early in training and on difficult tasks, where successful branches may be absent from an entire group.

Teacher guidance provides a complementary way to recover such missing targets. Interactive imitation and off-policy guidance learn from expert behavior on learner-induced inputs (Ross et al., 2011; Sun et al., 2017; Yan et al., 2026). OraRL shows that privileged supervision can be optimized directly, while on-policy distillation (OPD) queries a teacher on states reached by the current policy (Li et al., 2026b; Agarwal et al., 2024; Sun et al., 2026). Recent multi-turn OPD methods extend teacher guidance over multiple steps or use subsequent policy rollouts to validate an intervention (Wang et al., 2026c; Liao et al., 2026; Chen et al., 2026). However, their supervision is still centered on a corrective action, a teacher prefix, or the outcome of the intervention. In interactive environments, a correction changes the next observation and therefore the decisions that follow. When these downstream transitions are not directly learned, the training signal captures only part of the benefit of the correction and leaves its long-horizon consequences under-supervised.

We propose **State-Rooted Executable Rollout Optimization (SERO)** to learn from executable multi-step oracle continuations rooted at policy-visited states. As is shown in Figure 1, SERO first collects the ordinary on-policy trajectories required by the base RL algorithm. It then selects a state actually reached by the current policy, restores the environment at that state, and lets a training-only oracle continue interacting for multiple steps. Every oracle action is executed in the environment, so each later target is conditioned on the real observation produced by the preceding oracle actions. The resulting continuation therefore shows not only what action should be taken at the root state, but also the sequence of decisions that follows once that correction is actually carried out.

Deeper states in an oracle continuation are increasingly off-policy, since reaching them requires the current policy to follow the preceding oracle actions. We therefore introduce **policy-reachability weighting**. A frozen behavior policy scores each oracle action, and the supervision weight at step k is determined by the compatibility of the preceding oracle prefix under the behavior policy. Within a branch admitted by the improvement gate, the root target receives unit prefix weight, while deeper targets receive larger weights as the policy becomes more compatible with the oracle prefix. This down-weights supervision on poorly supported oracle prefixes and allows the effective supervision horizon to extend as compatibility improves. The oracle objective is added as an auxiliary loss, leaving the rewards, advantages, and policy ratios of the base RL algorithm unchanged, and the oracle is used only during training.

We evaluate SERO on ALFWorld (Shridhar et al., 2020) and WebShop (Yao et al., 2022a) with Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct. Adding SERO to GraphGPO or HGPO improves ALFWorld success and WebShop task success at both scales. We also compare against prompting

and strong RL baselines, and ablate root-only oracle targets, uniformly weighted full oracle continuations, and policy-reachability weighting. On ALFWorld, full executable continuations outperform root corrections, with reachability weighting adding further gains, especially on unseen tasks.

Our contributions are summarized as follows:

- We propose **State-Rooted Executable Rollout Optimization (SERO)**, which turns an oracle correction into a multi-step source of executable training experience. Starting from a state reached by the current policy, SERO executes the oracle in the environment and learns the downstream oracle actions under the observations created by that intervention.
- We introduce **policy-reachability weighting** as a reachability-aware way to learn from oracle-induced states. By tying each downstream target to the behavior-policy compatibility of the preceding oracle prefix, SERO suppresses supervision on poorly supported oracle-induced states and lets deeper guidance become effective as the policy improves.
- We design SERO as an auxiliary objective that complements existing on-policy credit assignment while leaving the base RL credit estimator unchanged. Experiments at both model scales show improvements in ALFWorld success and WebShop strict success for both GraphGPO and HGPO, while controlled ablations isolate the effects of executable continuation learning and reachability weighting.

2 Related Work

Group-based RL for long-horizon agents. Group-based RL has become a practical alternative to actor–critic training for large language models by replacing a learned value function with relative comparisons among sampled rollouts (Ahmadian et al., 2024; Shao et al., 2024; Yu et al., 2026; Liu et al., 2025; Lin et al., 2026; Zheng et al., 2025). In interactive tasks, however, sparse and delayed rewards make it difficult to determine which decisions actually helped. Recent work has also extended online RL to self-evolving web curricula and end-to-end multi-turn interaction, while newer methods study hindsight credit assignment and uncertainty-guided exploration in long-horizon agents (Qi et al., 2025; Wei et al., 2025; Tan et al., 2026; Wang et al., 2026b). Recent agentic RL therefore moves from one advantage per trajectory toward finer signals recovered from the collected experience, using repeated states, interaction history, or cross-trajectory transition structure to refine credit assignment (Zhou et al., 2024; Luo et al., 2025; Feng et al., 2026; He et al., 2026; Cheng et al., 2026; Wang et al., 2026a). These methods improve how the policy learns from its own rollouts, but they share a structural limit: supervision is still extracted from behavior already present in the sampled group. If every rollout misses a useful branch, a better estimator can re-rank the observed decisions but cannot provide that missing branch as a positive target.

Teacher guidance on learner-induced states. Imitation and distillation address the same difficulty from another direction by querying an expert on states induced by the learner rather than only replaying fixed demonstrations (Ross et al., 2011; Sun et al., 2017; Agarwal et al., 2024; Gu et al., 2024; Yan et al., 2026). Related approaches also construct corrective supervision from failed trajectories or expert interaction experience (Yuan et al., 2025; Lu et al., 2026). OPD and recent multi-turn extensions bring this principle to language-model post-training, aligning teacher supervision with the current policy and extending it through prefixes, refined trajectories, or short teacher interventions (Li et al., 2026a; Sun et al., 2026; Wang et al., 2026c; Liao et al., 2026; Jiang et al., 2026; Chen et al., 2026). Their common goal is to align teacher guidance with the learner's current behavior. SERO instead learns the executable future induced by the correction: after restoring a policy-visited state, the oracle continues the branch, and the resulting observations and downstream actions are used directly for training.

Oracle-guided optimization. Privileged information can also enter RL directly as an optimization signal. OraRL shows that an annotation can serve as a positive target rather than only a reward reference, while CAST uses solver information to refine turn-level credit (Li et al., 2026b; Wang et al., 2026d). SERO shares the motivation that finite on-policy samples can benefit from reliable privileged supervision, but interactive dynamics make the target fundamentally sequential: executing an oracle action changes the environment and creates new states. SERO therefore keeps the original on-policy estimator unchanged and learns a separate executable oracle continuation rooted

at on-policy experience. Policy-reachability weighting then controls the influence of deeper oracle-induced states according to the behavior-policy compatibility of the preceding oracle prefix.

3 PRELIMINARIES

3.1 LONG-HORIZON AGENTIC REINFORCEMENT LEARNING

We consider the standard sequential decision-making setting (Sutton & Barto, 2018) used in long-horizon agentic RL (Feng et al., 2026; He et al., 2026; Cheng et al., 2026). At step t, the policy observes an agent state $s_t \in \mathcal{S}$, which contains the current environment observation and the interaction context exposed by the agent interface, and samples an action $a_t \sim \pi_\theta(\cdot \mid s_t)$. The environment executes a_t and returns reward r_t and the next state s_{t+1}. A trajectory is $\tau = \{(s_t, a_t, r_t)\}_{t=0}^{T-1}$ with return $R(\tau)$. In long-horizon tasks such as ALFWorld (Shridhar et al., 2020) and WebShop (Yao et al., 2022a), rewards are sparse or delayed, so the contribution of an individual action is difficult to infer from the final outcome alone. During training, we additionally save the simulator snapshot corresponding to each visited state s_t so that the same decision point can be restored for oracle continuation. This snapshot is used only for training and is never included in the policy input.

3.2 GROUP-BASED POLICY OPTIMIZATION AND ORACLE SUPERVISION

Group-based RL replaces an explicit learned critic with statistics computed from sampled groups (Ahmadian et al., 2024; Shao et al., 2024). Following clipped policy optimization (Schulman et al., 2015; 2017), it samples N trajectories for the same task using a frozen behavior policy $\pi_b = \pi_{\theta_{\text{old}}}$. For example, GRPO assigns

$$\widehat{A}_i^{\text{GRPO}} = \frac{R(\tau_i) - \mu_R}{\sigma_R + \epsilon_{\text{num}}}, \qquad \mu_R = \frac{1}{N}\sum_{j=1}^{N} R(\tau_j), \tag{1}$$

where σ_R is the group reward standard deviation and $\epsilon_{\text{num}} > 0$ is a small constant for numerical stability. Recent agentic variants provide finer credit by comparing repeated states, conditioning on history, or using the transition structure of sampled trajectories (Feng et al., 2026; He et al., 2026; Cheng et al., 2026). We denote the advantage supplied by the chosen base method as $\widehat{A}_{i,t}^{\text{base}}$ and schematically write its clipped objective as

$$\mathcal{L}_{\text{base}} = -\frac{1}{M}\sum_{i,t} \min\Big(\rho_{i,t}\widehat{A}_{i,t}^{\text{base}}, \bar{\rho}_{i,t}\widehat{A}_{i,t}^{\text{base}}\Big) + \beta\mathcal{L}_{\text{reg}}, \tag{2}$$

where T_i is the number of valid decision steps in trajectory i, $M = \sum_i T_i$ is the number of valid decision steps in the batch, $\rho_{i,t} = \pi_\theta(a_{i,t} \mid s_{i,t})/\pi_b(a_{i,t} \mid s_{i,t})$, and $\bar{\rho}_{i,t} = \text{clip}(\rho_{i,t}, 1-\epsilon_c, 1+\epsilon_c)$ for clipping threshold $\epsilon_c > 0$. The optional term $\beta\mathcal{L}_{\text{reg}}$, with $\beta \geq 0$, denotes any regularization used by the base learner (e.g., a KL penalty).

These methods assign credit using trajectories generated by the policy itself. A different source of supervision is a training-time oracle. At a policy-visited state s_t, the oracle can provide a corrective action $a_t^\star = \mathcal{O}(s_t)$. In an interactive environment, this correction changes the next observation and the rest of the trajectory. Training only on the first oracle action therefore ignores the useful decisions that follow. Our method uses the resulting oracle continuation as multi-step supervision.

4 PROPOSED METHOD

4.1 OVERVIEW

SERO is a modular, oracle-augmentation framework that can be integrated with different on-policy RL objectives without modifying their underlying credit-assignment mechanisms. As is shown in Figure 2, each training iteration has three stages. First, the base RL method collects ordinary on-policy rollouts and computes its original RL objectives. Second, SERO selects a state from an on-policy trajectory, restores the environment to that decision point, and lets an oracle execute a fresh multi-step continuation. Third, the oracle actions are optimized with weights that reflect policy

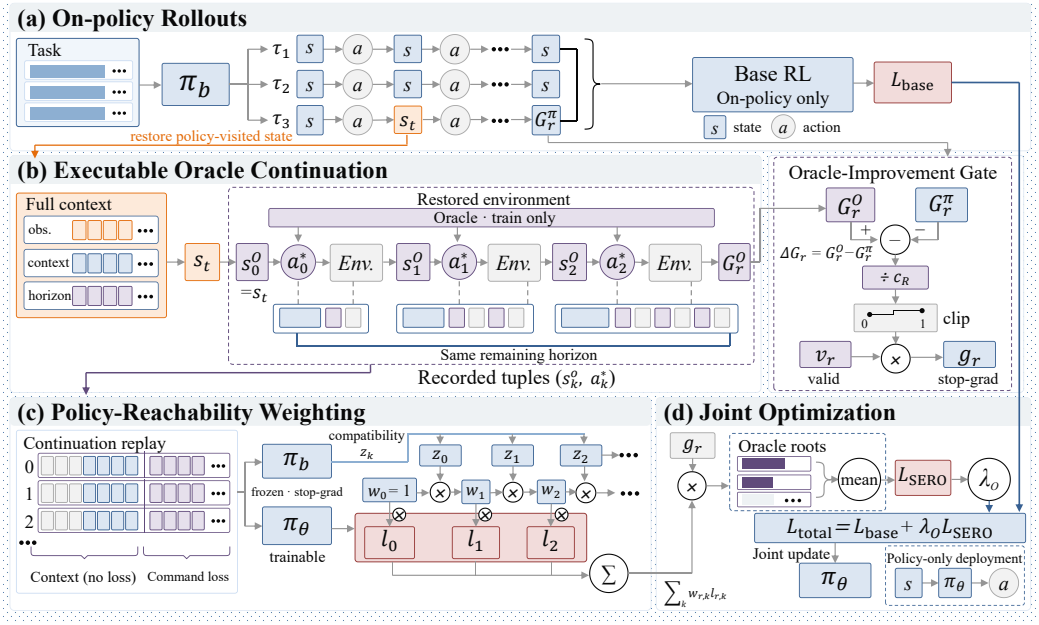


Figure 2: Overview of SERO. (a) The behavior policy collects on-policy rollouts for the base RL objective. (b) From a selected policy-visited state, SERO restores the environment, executes an oracle continuation under the same remaining horizon, and computes a detached improvement gate. (c) The frozen behavior policy provides compatibility scores used to form policy-reachability weights for downstream oracle-action losses. (d) The root-averaged auxiliary loss is combined with the base RL loss in one policy update. The oracle and environment restoration are used only during training, and only the learned policy is used at inference.

support for the oracle prefix leading to each downstream state. Oracle data are used only by the auxiliary branch and never enter the reward statistics, advantages, or importance ratios of the base method.

4.2 Executable Oracle Continuation

To obtain downstream oracle targets that remain valid after an intervention, SERO constructs a fresh oracle continuation from a state reached by the current policy. Let s_t be a selected pre-action state from an on-policy rollout. We restore the corresponding simulator snapshot in a separate environment copy, preserving the same decision point and remaining horizon. The restored state initializes the oracle branch as $s_0^O = s_t$, where s_k^O denotes the state at oracle step k.

$$a_k^\star = \mathcal{O}(s_k^O), \qquad (s_{k+1}^O, r_k^O) = \mathcal{E}(s_k^O, a_k^\star), \tag{3}$$

where $\mathcal{E}$ denotes one environment transition. Repeating this process until task completion or exhaustion of the remaining interaction budget gives

$$\tau_r^O = \left\{(s_k^O, a_k^\star, r_k^O)\right\}_{k=0}^{L_r-1}, \tag{4}$$

where L_r is the realized continuation length. The oracle continuation is generated step by step from the states actually reached in the restored environment. The oracle may use privileged training information for action selection, while the trainable policy is optimized only from the policy-visible state s_k^O. The on-policy suffix and oracle continuation share the same restored root and remaining horizon, allowing their returns to be compared from the same decision point.

4.3 Policy-Reachability Weighting

Executing the oracle makes each downstream target valid in the environment, but does not make it equally relevant to the current policy. The restored root s_0^O comes directly from an on-policy

trajectory, whereas reaching a deeper state s_k^O requires the current policy to follow the preceding oracle actions. SERO captures this with a prefix-reachability weight.

Let the oracle command at step k be $a_k^\star = (y_{k,1}^\star, \ldots, y_{k,m_k}^\star)$. Given the frozen behavior policy π_b, we measure the local compatibility of this command by its length-normalized likelihood:

$$z_k = \exp\left(\frac{1}{m_k}\sum_{j=1}^{m_k}\log \pi_b\left(y_{k,j}^\star \mid s_k^O, c_k, y_{k,<j}^\star\right)\right), \tag{5}$$

where c_k is the reasoning prefix produced by π_b at s_k^O and used as replay context; z_k is therefore a length-normalized compatibility score rather than a marginal action probability.

The weight for the k-th oracle target is then

$$w_0 = 1, \qquad w_k = \prod_{j=0}^{k-1} z_j, \qquad w_{k+1} = w_k z_k. \tag{6}$$

Thus, w_k measures how well the current policy supports the oracle prefix leading to the k-th target. The root action always receives unit prefix weight, while deeper supervision is reduced when earlier oracle actions have low compatibility under the behavior policy. As the policy becomes more compatible with the oracle prefix, the weights decay more slowly and supervision can extend further along the continuation. The corresponding discrete-action formulation is given in Appendix B.2.

4.4 Oracle Continuation Objective

Reachability determines how strongly each oracle target is supervised, but not whether the oracle continuation improves upon the policy's own future from the same root. SERO therefore uses a branch-level improvement gate to control its contribution to the auxiliary objective.

Index attempted oracle roots by $r = 1, \ldots, N_{\text{root}}$. Let G_r^π and G_r^O denote the realized returns of the on-policy suffix and oracle continuation, respectively, over the same remaining horizon. We define

$$\Delta G_r = G_r^O - G_r^\pi, \qquad g_r = v_r \ \text{clip}\left(\frac{\Delta G_r}{c_R}, 0, 1\right), \tag{7}$$

where $v_r \in \{0, 1\}$ indicates a valid restored branch and $c_R > 0$ is a reward-normalization scale, with $c_R = 1$ for binary success returns.

For a contributing oracle branch, SERO trains the policy to reproduce each oracle action using a supervised likelihood loss. For a language command,

$$\ell_k(\theta) = -\frac{1}{m_k}\sum_{j=1}^{m_k}\log \pi_\theta(y_{k,j}^\star \mid s_k^O, c_k, y_{k,<j}^\star), \tag{8}$$

which reduces to $-\log \pi_\theta(a_k^\star \mid s_k^O)$ for a categorical action.

Let L_r be the continuation length of root r, $w_{r,k}$ and $\ell_{r,k}$ its step-k reachability weight and oracle-action loss, and $\text{sg}[\cdot]$ the stop-gradient operator. Combining branch-level improvement, prefix reachability, and oracle supervision gives

$$\mathcal{L}_{\text{SERO}}(\theta) = \frac{1}{\max(N_{\text{root}}, 1)}\sum_{r=1}^{N_{\text{root}}} \underbrace{\text{sg}[g_r]}_{\text{branch improvement}} \sum_{k=0}^{L_r-1} \underbrace{\text{sg}[w_{r,k}]}_{\text{prefix reachability}} \cdot \underbrace{\ell_{r,k}(\theta)}_{\text{oracle supervision}}. \tag{9}$$

The local compatibility score z_k enters through the construction of $w_{r,k}$. Both branch and reachability weights are computed from pre-update quantities and detached from the gradient.

SERO thus adds a decoupled oracle-supervision objective alongside the base RL loss, without altering the on-policy advantages or requiring importance-ratio correction.

$$\mathcal{L}_{\text{total}}(\theta) = \mathcal{L}_{\text{base}}(\theta) + \lambda_O \mathcal{L}_{\text{SERO}}(\theta), \tag{10}$$

where $\lambda_O \geq 0$ controls the strength of oracle-continuation supervision. On-policy trajectories alone determine the base learner's rewards, group statistics, advantages, and policy ratios. This makes SERO compatible with different credit estimators, including GRPO (Shao et al., 2024), GiGPO (Feng et al., 2026), HGPO (He et al., 2026), and GraphGPO (Cheng et al., 2026). The oracle and environment-restoration machinery are used only during training and are removed at inference time.

5 EXPERIMENTS

We evaluate SERO on long-horizon interactive decision making with two complementary goals. First, we ask whether learning an executable oracle continuation improves the policy beyond outcome-based RL and single-step oracle supervision. Second, we isolate whether the gain comes from the executable oracle continuation or from policy-reachability weighting.

5.1 EXPERIMENTAL SETUP

Benchmarks and baselines. ALFWorld evaluates multi-step household instruction following in an interactive text environment (Shridhar et al., 2020). We report success on the in-distribution and out-of-distribution splits (*In-Success* and *Out-Success*). WebShop requires an agent to search, navigate product pages, select attributes, and purchase an item (Yao et al., 2022a); following prior work, we report both task score and strict task success rate. We compare against prompting baselines (Qwen2.5, ReAct (Yao et al., 2022b), and Reflexion (Shinn et al., 2023)), critic-based PPO (Schulman et al., 2017), group-based RLOO (Ahmadian et al., 2024) and GRPO (Shao et al., 2024), and recent long-horizon agentic RL methods GiGPO (Feng et al., 2026), GraphGPO (Cheng et al., 2026) and HGPO (He et al., 2026).

Models and training. Our main experiments use Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct. We use the ALFWorld expert planner and a goal-metadata oracle for WebShop as training-time oracles. An eligible group contains at least one failed rollout, from which one pre-action state is sampled as the oracle root. Unless otherwise noted, SERO uses one root per eligible group and $\lambda_O = 0.1$. Detailed settings are given in Appendix E.

5.2 MAIN RESULTS

Table 1 summarizes results on ALFWorld and WebShop. Across both model scales, SERO improves ALFWorld and WebShop success on both RL backbones, with the largest gains on out-of-distribution metrics. At 1.5B, GraphGPO+SERO improves ALFWorld seen/unseen success from $90.63/88.02$ to $95.83/91.67$ and WebShop score/success from $87.27/77.93$ to $88.90/79.27$; HGPO+SERO improves ALFWorld from $92.19/87.24$ to $94.53/90.10$ and WebShop success from 73.70 to 77.47. At 7B, GraphGPO+SERO improves ALFWorld from $91.67/76.30$ to $95.57/87.50$ and WebShop from $89.77/80.47$ to $91.93/83.67$, while HGPO+SERO improves ALFWorld from $94.01/90.89$ to $94.53/91.41$ and WebShop success from 74.47 to 79.63. The largest gain is 11.20 points on ALFWorld unseen with 7B GraphGPO, showing a particularly large benefit on out-of-distribution evaluation for this backbone.

5.3 ABLATION OF ORACLE SUPERVISION

We ablate two components of SERO: whether supervision extends beyond the root oracle action, and whether downstream oracle actions are weighted by prefix reachability. All variants use Qwen2.5-1.5B-Instruct with GraphGPO.

Root-only uses only the first oracle action and discards the remaining continuation. *Full-Flat* instead trains the complete oracle continuation, but assigns every step in a branch the same average weight,

$$\bar{w}_r = \frac{1}{L_r} \sum_{k=0}^{L_r-1} w_{r,k}, \qquad \mathcal{L}_{\mathrm{flat},r}^{\mathrm{branch}} = g_r \bar{w}_r \sum_{k=0}^{L_r-1} \ell_{r,k}. \tag{11}$$

This construction preserves the total branch-level weight, $L_r \bar{w}_r = \sum_{k=0}^{L_r-1} w_{r,k}$, while removing its step-wise allocation across oracle depth. Full-Flat retains a policy-dependent branch scale through $\bar{w}_r$; this control tests how that weight is distributed across oracle steps. The full SERO trains the same multi-step oracle continuation while retaining the exclusive-prefix reachability weight $w_{r,k}$ for each downstream action.

Effect of the oracle continuation. Root-only does not improve over GraphGPO and instead decreases ALFWorld seen/unseen success by 2.87/4.17 points. It also yields little change on WebShop.

Table 1: Main results on ALFWorld and WebShop. ALFWorld reports in-distribution and out-of-distribution success rates (%); WebShop reports task score and task success rate (%). Results are averaged over 3 random seeds during testing. The best and second-best performances are highlighted in bold and underlined, respectively.

Model	Type	Method	ALFWorld		WebShop	
			In-Success	Out-Success	Task Score	Task Success
Closed-source	Prompting	GPT-4o	48.0	46.0	31.8	23.7
	Prompting	Gemini-2.5-Pro	60.3	50.5	42.5	35.9
Qwen2.5-1.5B-Instruct	Prompting	Qwen2.5	4.1	–	23.1	5.2
	Prompting	ReAct	12.8	–	40.1	11.3
	Prompting	Reflexion	21.8	–	55.8	21.9
	RL Training	PPO (with critic)	54.4 ±3.1	–	73.8 ±3.0	51.5 ±2.9
	RL Training	RLOO	69.7 ±2.5	68.7 ±10.7	73.9 ±5.6	52.1 ±6.7
	RL Training	GRPO	72.8 ±3.6	70.1 ±2.5	75.8 ±3.5	56.8 ±3.8
	RL Training	GiGPO	88.27 ±1.08	86.47 ±2.25	83.10 ±1.6	65.00 ±3.21
	RL Training	GraphGPO	90.63 ±3.40	88.02 ±3.23	87.27 ±2.83	77.93 ±2.90
	RL Training	GraphGPO + SERO	**95.83 ±1.22**	**91.67 ±0.92**	**88.90 ±1.59**	**79.27 ±1.72**
	RL Training	HGPO	92.19 ±0.80	87.24 ±4.01	87.80 ±1.67	73.70 ±2.95
	RL Training	HGPO + SERO	94.53 ±1.33	90.10 ±1.61	88.43 ±0.93	77.47 ±1.88
Qwen2.5-7B-Instruct	Prompting	Qwen2.5	14.8	–	26.4	7.8
	Prompting	ReAct	31.2	–	46.2	19.5
	Prompting	Reflexion	42.7	–	58.1	28.8
	RL Training	PPO (with critic)	77.08 ±1.12	76.23 ±1.46	81.4 ±3.1	68.7 ±5.1
	RL Training	RLOO	77.86 ±0.03	73.95 ±0.05	80.3 ±3.2	65.7 ±4.0
	RL Training	GRPO	78.64 ±0.73	76.82 ±1.47	79.3 ±2.8	66.1 ±3.7
	RL Training	GiGPO	92.29 ±1.56	86.94 ±2.08	79.31 ±2.80	66.10 ±3.71
	RL Training	GraphGPO	91.67 ±0.92	76.30 ±2.76	89.77 ±0.81	80.47 ±1.53
	RL Training	GraphGPO + SERO	**95.57 ±0.46**	87.50 ±2.06	**91.93 ±0.42**	**83.67 ±0.95**
	RL Training	HGPO	94.01 ±2.05	90.89 ±1.97	87.83 ±2.00	74.47 ±3.10
	RL Training	HGPO + SERO	94.53 ±3.15	**91.41 ±2.12**	87.63 ±1.16	79.63 ±0.50

Table 2: Ablation of oracle supervision design on ALFWorld and WebShop. Results use Qwen2.5-1.5B-Instruct with GraphGPO. Root-only supervises only the first oracle action, Full-Flat uses the full oracle continuation with a shared branch-level weight, and SERO applies exclusive-prefix reachability weighting along the continuation.

Variant	Oracle target	Weighting	ALFWorld		WebShop	
			Seen SR	Unseen SR	Task Score	Task Success
Base†	None	—	90.63 ±3.40	88.02 ±3.23	87.27 ±2.83	77.93 ±2.90
+ Root-only	Root action	—	87.76 ±1.67	83.85 ±2.76	87.30 ±1.54	77.73 ±2.38
+ Full-Flat	Full continuation	Branch mean	94.79 ±0.46	88.28 ±2.35	87.63 ±1.91	78.33 ±2.75
+ SERO	Full continuation	Exclusive prefix	95.83 ±1.22	91.67 ±0.92	88.90 ±1.59	79.27 ±1.72

In contrast, Full-Flat substantially improves the results, reaching 94.79 seen success on ALFWorld and 78.33 task success on WebShop. Relative to Root-only, using the full executable continuation improves ALFWorld seen/unseen success by $7.03/4.43$ points and WebShop task score/success by $0.33/0.60$ points. These results indicate that useful oracle supervision extends beyond the first corrective action.

Effect of prefix reachability weighting. Replacing the shared branch weight with the exclusive-prefix weights further improves all four metrics. Compared with Full-Flat, SERO gains 1.04 points on ALFWorld seen success and 3.39 points on unseen success, while WebShop task score and task success improve by 1.27 and 0.94 points. The larger gain on ALFWorld unseen suggests that selectively down-weighting deep oracle actions whose preceding prefix is poorly supported by the current policy may be particularly beneficial for out-of-distribution generalization. To isolate the weighting rule from differences in collected trajectories, we further replay all weighting variants on the same frozen oracle bank (Appendix A.1). Under identical roots and oracle continuations, SERO achieves $88.02/85.94$ seen/unseen success, compared with $85.42/84.64$ for flat within-branch weighting and

$87.24/82.55$ for a depth-decay baseline, supporting the benefit of policy-dependent prefix reachability beyond depth decay alone.

Sensitivity to λ_O. Figure 3 shows that SERO is relatively stable over a moderate range of oracle-loss weights. On ALFWorld, both seen and unseen performance remain strong for $\lambda_O \in [0.1, 0.5]$, while WebShop shows a similar pattern for task score and success rate. Performance drops when the oracle loss is too strong at $\lambda_O = 1.0$, whereas $\lambda_O = 0.05$ yields weaker results, particularly on unseen ALFWorld tasks. We therefore use $\lambda_O = 0.1$ as the default setting in the main experiments.

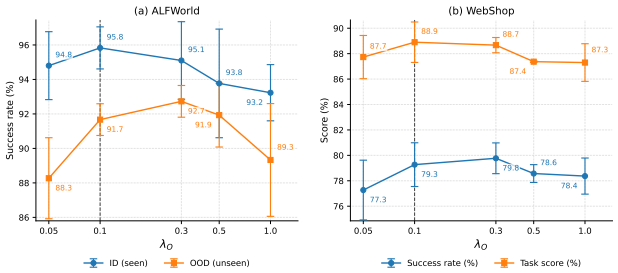


Figure 3: Sensitivity to λ_O on ALFWorld and WebShop. Performance is stable over moderate oracle-loss weights, with $\lambda_O = 0.1$ used as the default.

5.4 Policy Reachability Across Oracle Depth

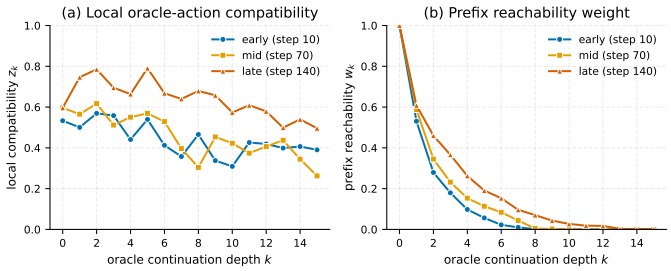


Figure 4: Policy compatibility along oracle continuations. Local compatibility z_k and prefix reachability weight w_k for the Qwen2.5-1.5B GraphGPO+$SERO$ model on ALFWorld at training steps 10, 70, and 140.

Figure 4 shows why local compatibility alone is insufficient for weighting a long oracle continuation. In Figure 4(a), z_k fluctuates with depth and often remains relatively high even for later oracle actions. This is because z_k only measures whether the action at the current oracle state is compatible with the behavior policy; it does not account for whether the policy could follow the preceding oracle actions to reach that state. Figure 4(b) captures this missing information through the prefix weight $w_k = \prod_{j<k} z_j$. Once an earlier oracle action has low compatibility, the weights of all subsequent actions are reduced accordingly. As a result, w_k drops quickly at the early checkpoint, while the later checkpoint maintains larger weights over a longer portion of the continuation. As training progresses, SERO therefore makes use of increasingly deeper oracle supervision, while avoiding assigning the same weight to every downstream oracle action.

6 Conclusion

We presented SERO, a framework that incorporates executable oracle continuations into multi-turn agentic RL. Starting from states visited by the current policy, SERO restores the environment, executes oracle continuations, and uses the resulting action sequences as auxiliary supervision. To better utilize long-horizon guidance, we introduce policy-reachability weighting, which adjusts oracle supervision based on the compatibility between oracle prefixes and the current policy. With Qwen2.5-1.5B and 7B models, adding SERO improves ALFWorld success and WebShop strict success for both GraphGPO and HGPO. Ablations show that learning from full executable continuations provides substantially stronger gains than isolated corrective actions, while reachability weighting further improves generalization. These results highlight the value of executable multi-step oracle experience and the importance of accounting for policy reachability when extending supervision beyond immediate actions.

AI USE STATEMENT

In this work, we used generative AI tools to assist with writing and language polishing, including improving grammar, clarity, and phrasing. We did not use generative AI tools to generate experimental results or make scientific conclusions, and other disclosure categories are not applicable to this work. All AI-assisted text was reviewed and revised by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of AI.

REPRODUCIBILITY STATEMENT

We provide detailed information to support reproducibility throughout the appendix. Appendix C specifies the full training procedure, environment restoration, data separation, and gradient flow of SERO. Appendix D describes the task-specific protocols and oracle backends for ALFWorld and WebShop, while Appendix E reports the training and evaluation configurations used in our experiments. The definitions of all ablation variants are provided in Appendix F, and additional controlled experiments are reported in Appendix A. Together, these sections document the implementation, training, evaluation, and controlled-comparison procedures used to obtain the reported results.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024.

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.

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

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Chishui Chen, Yaoyou Fan, Te Sun, Yi Yang, Chenghao Sun, Delin Mao, Hongbo Qiao, Zuowei Zhang, Junxi Wang, Chenxing Sun, et al. Look ahead before you distill: Future trajectory validation of teacher guidance for agentic on-policy distillation. *arXiv preprint arXiv:2608.01953*, 2026.

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

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

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

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *International Conference on Learning Representations*, volume 2024, pp. 32694–32717, 2024.

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

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. In *International Conference on Learning Representations*, volume 2024, pp. 52690–52717, 2024.

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.

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

Li Jiang, Haoran Xu, Yichuan Ding, and Amy Zhang. Trajectory-refined distillation. *arXiv preprint arXiv:2606.08432*, 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.

Jiaze Li, Hao Yin, Haoran Xu, Boshen Xu, Wenhui Tan, Zewen He, Jianzhong Ju, Zhenbo Luo, and Jian Luan. Video-opd: Efficient post-training of multimodal large language models for temporal video grounding via on-policy distillation. *arXiv preprint arXiv:2602.02994*, 2026a.

Yunheng Li, Guohong Mu, Hao Li, Shengsheng Qian, Dingwen Zhang, Qibin Hou, and Ming-Ming Cheng. Annotations as rollouts: Efficient and scalable reinforcement learning for video mllms. *arXiv preprint arXiv:2608.20492*, 2026b.

Baohao Liao, Hanze Dong, Christof Monz, Xinxing Xu, Li Dong, and Furu Wei. Multi-turn on-policy distillation with prefix replay. *arXiv preprint arXiv:2607.04763*, 2026.

Zhihang Lin, Mingbao Lin, Yuan Xie, and Rongrong Ji. Cppo: Accelerating the training of group relative policy optimization-based reasoning models. *Advances in Neural Information Processing Systems*, 38:61043–61068, 2026.

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

Zhengxi Lu, Jiabo Ye, Fei Tang, Yongliang Shen, Haiyang Xu, Ziwei Zheng, Weiming Lu, Ming Yan, Fei Huang, Jun Xiao, et al. Experience-driven multi-turn reinforcement learning for gui agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9474–9496, 2026.

Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K Qiu, and Yuqing Yang. Agent lightning: Train any ai agents with reinforcement learning. *arXiv preprint arXiv:2508.03680*, 2025.

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. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

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

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.

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. *Advances in neural information processing systems*, 36:68539–68551, 2023.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. Pmlr, 2015.

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, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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

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

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7584–7600, 2024.

Wen Sun, Arun Venkatraman, Geoffrey J Gordon, Byron Boots, and J Andrew Bagnell. Deeply aggrevated: Differentiable imitation learning for sequential prediction. In *International conference on machine learning*, pp. 3309–3318. PMLR, 2017.

Xin Sun, Jianan Xie, Zhongqi Chen, Qiang Liu, Shu Wu, Bowen Song, Weiqiang Wang, Zilei Wang, and Liang Wang. Gapd: Gold-action policy distillation for agentic reinforcement learning in knowledge base question answering. *arXiv preprint arXiv:2605.29584*, 2026.

Richard S Sutton and Andrew G Barto. Reinforcement learning: an introduction, 2nd edn. adaptive computation and machine learning, 2018.

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

Weihao Tan, Wentao Zhang, Shanqi Liu, Longtao Zheng, Xinrun Wang, and Bo An. True knowledge comes from practice: Aligning large language models with embodied environments via reinforcement learning. In *International Conference on Learning Representations*, volume 2024, pp. 54281–54328, 2024.

Daoyu Wang, Qingchuan Li, Mingyue Cheng, Jie Ouyang, Shuo Yu, Qi Liu, and Enhong Chen. Steppo: Step-aligned policy optimization for agentic reinforcement learning. *arXiv preprint arXiv:2604.18401*, 2026a.

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.

Haixin Wang, Hejie Cui, Chenwei Zhang, Xin Liu, Shuowei Jin, Shijie Geng, Xinyang Zhang, Nasser Zalmout, Zhenyu Shi, and Yizhou Sun. T2 po: Uncertainty-guided exploration control for stable multi-turn agentic reinforcement learning. *arXiv preprint arXiv:2605.02178*, 2026b.

Jiaqi Wang, Wenhao Zhang, Weijie Shi, Yaliang Li, and James Cheng. Tcod: Exploring temporal curriculum in on-policy distillation for multi-turn autonomous agents. *arXiv preprint arXiv:2604.24005*, 2026c.

Yu Wang, Yi-Kai Zhang, Wentao Shi, Ziang Ye, Yuchun Miao, Yueqing Sun, Qi Gu, Xunliang Cai, Lan-Zhe Guo, Han-Jia Ye, et al. Cast: Game solvers as turn-level teachers for llm agents. *arXiv preprint arXiv:2607.25308*, 2026d.

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.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *Advances in Neural Information Processing Systems*, 38:78500–78525, 2026.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, et al. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 7920–7939, 2025.

Muning Wen, Ziyu Wan, Jun Wang, Weinan Zhang, and Ying Wen. Reinforcing llm agents via policy optimization with action decomposition. *Advances in Neural Information Processing Systems*, 37: 103774–103805, 2024.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. *Advances in Neural Information Processing Systems*, 38:117157–117186, 2026.

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

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

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.

Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. Agent-r: Training language model agents to reflect via iterative self-training. *arXiv preprint arXiv:2501.11425*, 2025.

Yuexiang Zhai, Hao Bai, Zipeng Lin, Jiayi Pan, Shengbang Tong, Yifei Zhou, Alane Suhr, Saining Xie, Yann LeCun, Yi Ma, et al. Fine-tuning large vision-language models as decision-making agents via reinforcement learning. *Advances in neural information processing systems*, 37:110935–110971, 2024.

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

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 Additional Controlled Results

A.1 Fixed-Bank Weighting Control

The online ablation in Table 2 compares variants that continue to collect their own training trajectories. We therefore construct a frozen oracle bank from the Qwen2.5-1.5B GraphGPO checkpoint at step 50 and replay all weighting variants on exactly the same supervision data. The bank contains 271 valid roots and 5,002 oracle-continuation steps, and all variants use the same roots, continuations, optimizer settings, and step-50-to-80 replay schedule. The stored reasoning contexts, token masks, compatibility scores, prefix weights, and branch gates remain fixed throughout replay. All variants share the auxiliary sampling schedule while continuing ordinary on-policy GraphGPO updates.

The *Flat* variant distributes each branch's total supervision mass uniformly across its oracle steps. *Depth decay* replaces policy reachability with a fixed geometric decay, using $0 < \alpha_{\text{depth}} \leq 1$,

$$\widetilde{w}_{r,k} = \kappa_r \alpha_{\text{depth}}^k, \qquad \kappa_r = \frac{\sum_{j=0}^{L_r-1} w_{r,j}}{\sum_{j=0}^{L_r-1} \alpha_{\text{depth}}^j}, \tag{12}$$

while preserving the same total branch weight as SERO. Before replay, we fit $\log \alpha_{\text{depth}}$ once by least squares through the origin to $\log w_{r,k}$ versus k, using valid positive-gate branches with $k \geq 1$ and $w_{r,k} > 0$; α_{depth} is clipped to $[10^{-4}, 1]$ and then held fixed. ***Shuffled*** keeps the same set of SERO weights within each branch but permutes their positions using a fixed, reproducible permutation for that branch, breaking the correspondence between prefix compatibility and the supervised oracle step.

Table 3 shows that SERO performs best under this controlled replay setting, reaching $88.02/85.94$ seen/unseen success. It improves over Flat by $2.60/1.30$ points and over Depth decay by $0.78/3.39$ points. More importantly, shuffling the SERO weights reduces performance to $85.16/80.73$, a drop of 2.86 points on seen tasks and 5.21 points on unseen tasks. This shows that the benefit does not come only from the overall amount of supervision or from assigning smaller weights to deeper steps; the weight needs to remain aligned with the reachability of the corresponding oracle prefix.

A.2 Matched Extra-Exploration Control

We further test whether the gain of SERO can be explained by its additional environment interactions. The baseline trains GraphGPO for 15 extra steps without oracle supervision, targeting the estimated environment-transition budget of GraphGPO+$SERO$. The estimate includes primary rollouts, prefix replay for restoration, and oracle continuations, using calibration windows from early, middle, and late training; the match is approximate. As shown in Table 4, the extra exploration improves GraphGPO from $90.63/88.02$ to $93.49/90.36$ seen/unseen success. However, SERO still reaches $95.83/91.67$, outperforming the matched baseline by 2.34 and 1.31 points, respectively. The extended GraphGPO run narrows the gap but does not match SERO in this control.

Table 3: Fixed-bank weighting control on ALFWorld. All variants replay the same frozen roots and oracle continuations from the GraphGPO step-50 checkpoint and are evaluated at step 80. Shuffled preserves the SERO weights within each branch but permutes their positions along the continuation.

Weighting	Seen SR	Unseen SR
Flat	85.42 ±2.39	84.64 ±1.79
Depth decay	87.24 ±2.94	82.55 ±0.46
SERO	**88.02 ±1.22**	**85.94 ±0.75**
Shuffled	85.16 ±3.43	80.73 ±1.79

Table 4: Matched-interaction control on ALFWorld. Extra denotes GraphGPO trained for 15 additional steps to approximately match the estimated environment-transition budget of GraphGPO+$SERO$, including prefix restoration and oracle execution.

Method	Seen SR	Unseen SR
GraphGPO	90.63 ±3.40	88.02 ±3.23
GraphGPO + Extra	93.49 ±1.22	90.36 ±3.43
GraphGPO + SERO	95.83 ±1.22	91.67 ±0.92

A.3 Empirical Check of the Reachability Proxy

We test whether w_k tracks the frozen policy's ability to reproduce an oracle prefix. Using the GraphGPO step-50 checkpoint and the 271-root bank of Appendix A.1, we sample 16 commands at each recorded oracle state and run eight independent prefix trials per root. A trial proceeds along the recorded branch only while its sampled commands exactly match the oracle commands; it does not execute a new environment trajectory. The single-step compatibility z_k correlates with the sampled action-match frequency (Spearman $\rho = 0.763$). Table 5 groups the observed prefix-match frequency by w_k across root–depth pairs. The observed match frequency rises across the w_k bins. Since z_k is a geometric mean over command-token probabilities, w_k ranks prefix compatibility but is not the probability of reproducing the full command prefix.

Table 5: Empirical oracle-prefix reproduction under the frozen GraphGPO step-50 policy. Bins pool 2,869 root–depth pairs from 271 roots; each root has eight sampled prefix trials. The final column gives the fraction of trials that reproduce the full oracle prefix.

w_k interval	Root–depth pairs	Prefix match (%)
$[0, 0.05)$	2,449	0.00
$[0.05, 0.10)$	75	1.00
$[0.10, 0.25)$	127	1.08
$[0.25, 0.50)$	92	6.66
$[0.50, 1]$	126	40.97

A.4 Current-Policy Root Control

We also compare the source of the restored oracle root under a fixed-bank replay from the GraphGPO step-50 checkpoint. The reference bank selects failed-rollout states visited by that checkpoint; the alternative selects states visited by a step-10 policy. Both matched banks contain 271 roots and 78 positive-gate branches, with 5,002 and 4,961 oracle steps, respectively; 242 of 271 paired roots have the same task family. After the same 30-update replay, the step-50 roots reach $88.0/85.9$ seen/unseen success, compared with $84.4/83.9$ for step-10 roots. The more recent roots perform better in this replay, although the task instances and root horizons are not exactly matched.

A.5 ALFWorld Task-Type Breakdown

Table 6 breaks ALFWorld success into the six task families used by the benchmark. We report GraphGPO and HGPO at both 1.5B and 7B under the same 128-game final-checkpoint protocol as the main table. Improvements with SERO appear across several task families on both backbones and model scales, showing that the aggregate gains are not driven by a single category.

Table 6: ALFWorld success rate by task type. Pick2 denotes tasks requiring two objects to be placed. All reports the overall success rate on the corresponding split.

Model	Split	Method	Pick	Clean	Cool	Look	Heat	Pick2	All
1.5B	Seen	GraphGPO	92.57 ±3.65	100.00 ±0.00	81.27 ±4.41	81.73 ±8.55	87.17 ±8.04	87.27 ±8.49	90.63 ±3.40
	Seen	GraphGPO + SERO	99.17 ±1.44	96.00 ±1.13	87.33 ±4.18	95.23 ±8.26	98.13 ±3.23	96.47 ±3.07	95.83 ±1.22
	Seen	HGPO	93.27 ±1.00	97.10 ±2.69	83.77 ±3.03	77.13 ±4.97	98.33 ±2.89	92.77 ±8.57	92.19 ±0.80
	Seen	HGPO + SERO	100.00 ±0.00	97.10 ±2.69	85.33 ±1.88	91.90 ±7.34	96.50 ±6.06	89.23 ±5.59	94.53 ±1.33
	Unseen	GraphGPO	79.77 ±9.08	85.33 ±5.47	96.03 ±3.86	100.00 ±0.00	85.03 ±6.66	87.70 ±14.71	88.02 ±3.23
	Unseen	GraphGPO + SERO	83.73 ±1.70	88.60 ±7.85	100.00 ±0.00	100.00 ±0.00	94.60 ±0.35	85.87 ±5.34	91.67 ±0.92
	Unseen	HGPO	79.63 ±6.30	84.40 ±9.26	84.87 ±4.51	100.00 ±0.00	94.60 ±0.35	94.07 ±5.59	87.24 ±4.01
	Unseen	HGPO + SERO	87.07 ±9.35	93.83 ±4.27	92.43 ±2.24	96.83 ±5.48	85.90 ±7.54	88.07 ±5.38	90.10 ±1.61
7B	Seen	GraphGPO	99.17 ±1.44	97.10 ±2.69	77.57 ±3.56	100.00 ±0.00	86.20 ±7.41	87.57 ±5.98	91.67 ±0.92
	Seen	GraphGPO + SERO	100.00 ±0.00	100.00 ±0.00	85.73 ±3.15	83.80 ±10.74	93.17 ±5.92	100.00 ±0.00	95.57 ±0.46
	Seen	HGPO	99.17 ±1.44	100.00 ±0.00	85.90 ±6.63	84.67 ±5.77	87.70 ±3.08	98.13 ±3.23	94.01 ±2.05
	Seen	HGPO + SERO	100.00 ±0.00	95.37 ±5.36	91.03 ±8.71	84.90 ±7.53	96.57 ±2.98	91.10 ±2.91	94.53 ±3.15
	Unseen	GraphGPO	72.87 ±8.41	76.87 ±8.95	83.60 ±5.16	96.53 ±3.03	69.47 ±4.79	68.00 ±8.63	76.30 ±2.76
	Unseen	GraphGPO + SERO	68.50 ±4.35	83.03 ±7.20	100.00 ±0.00	100.00 ±0.00	92.77 ±3.36	89.97 ±9.65	87.50 ±2.06
	Unseen	HGPO	84.43 ±8.17	93.83 ±4.27	92.43 ±2.24	98.13 ±3.23	80.37 ±8.36	100.00 ±0.00	90.89 ±1.97
	Unseen	HGPO + SERO	85.23 ±5.50	93.83 ±4.27	92.43 ±2.24	100.00 ±0.00	90.90 ±6.59	94.43 ±5.10	91.41 ±2.12

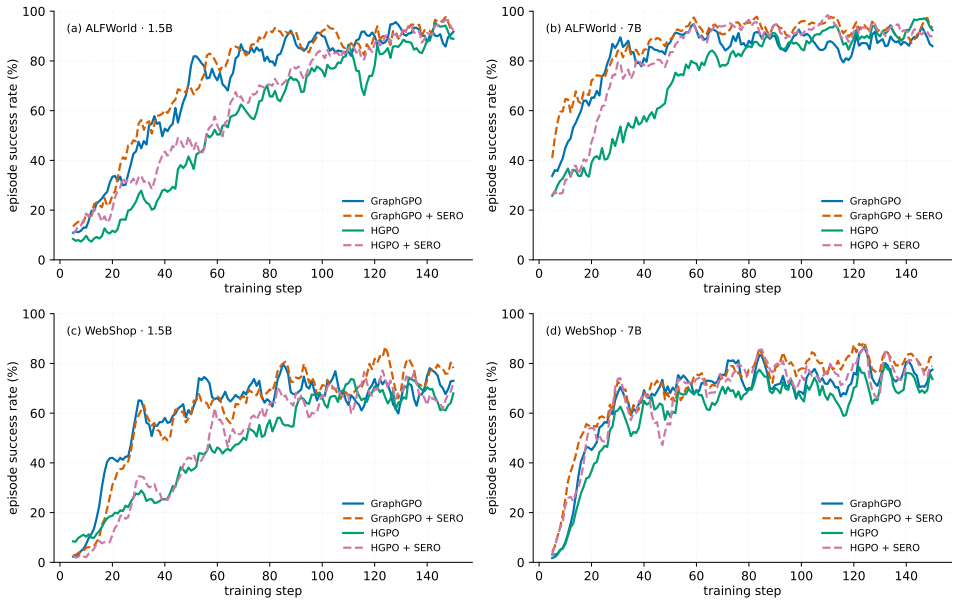


Figure 5: Training success rate on ALFWorld and WebShop. We compare GraphGPO and HGPO with and without SERO for Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct. Each curve reports the episode success rate of the primary training rollouts over training steps.

For GraphGPO, the largest 1.5B seen gains appear on Look, Heat, and Pick2, while unseen gains are strongest on Heat; at 7B, the largest unseen gains occur on Clean, Cool, Heat, and Pick2. Its few category-level regressions are not consistent across model scales. HGPO likewise shows gains across multiple categories, with some task-specific trade-offs.

A.6 Training Dynamics

Figure 5 compares the training success rates of GraphGPO and HGPO with and without SERO across both model scales and benchmarks. On ALFWorld, adding SERO generally leads to faster improvement during the early and middle stages of training, with the difference particularly clear for HGPO at 1.5B and for both backbones at 7B. The WebShop curves show more crossings: for example, the 1.5B GraphGPO baseline improves earlier during part of the initial training period, while the advantage of GraphGPO+SERO is clearer later. The curves therefore show gains before the final evaluation checkpoint, with their timing and magnitude varying across settings.

A.7 Short-Bridge Control

The short-bridge control executes one oracle action from the restored root and then returns control to the frozen behavior policy for the remaining horizon. Its gate is computed from the resulting bridge return, $g_r^{\text{bridge}} = v_r[G_r^{\text{bridge}} - G_r^{\pi}]_+$, but only the root oracle action is distilled:

$$\mathcal{L}_{\text{bridge}} = \frac{1}{\max(N_{\text{root}}, 1)} \sum_{r=1}^{N_{\text{root}}} \text{sg}[g_r^{\text{bridge}}]\, \ell_{r,0}(\theta). \tag{13}$$

The behavior-policy continuation provides no auxiliary target and does not enter the base RL statistics. This isolates whether a single corrective action followed by the current policy is sufficient, or whether learning the full oracle continuation provides additional value.

Table 7: **Short-bridge control on ALFWorld and WebShop.** Root-only supervises only the first oracle target, Short Bridge supervises that root action while using a frozen-policy continuation only to evaluate its outcome, and SERO learns the full executable oracle continuation.

	ALFWorld		WebShop	
Variant	Seen SR	Unseen SR	Task Score	Task Success
GraphGPO	90.63 ±3.40	88.02 ±3.23	87.27 ±2.83	77.93 ±2.90
+ Root-only	87.76 ±1.67	83.85 ±2.76	87.30 ±1.54	77.73 ±2.38
+ Short Bridge	91.67 ±5.83	86.72 ±2.35	87.60 ±1.47	78.77 ±1.75
+ SERO	95.83 ±1.22	91.67 ±0.92	88.90 ±1.59	79.27 ±1.72

As shown in Table 7, Short Bridge consistently improves over Root-only on both benchmarks. On ALFWorld, it raises seen/unseen success from $87.76/83.85$ to $91.67/86.72$, but still trails the full SERO by $4.16/4.95$ points. On WebShop, Short Bridge improves task score/success from $87.30/77.73$ to $87.60/78.77$, while full SERO further reaches $88.90/79.27$. These results show that a one-step oracle intervention is helpful, but learning the full executable continuation provides a stronger training signal.

A.8 Horizon-Stratified Evaluation

We further group ALFWorld episodes by expert-plan length to examine how the effect of SERO changes with task horizon. Table 8 reports results for short (≤ 5), medium (6), and long (≥ 7) plans using the same final checkpoints as the main evaluation.

Table 8: **ALFWorld success stratified by expert-plan length.** Short, medium, and long correspond to expert-plan lengths ≤ 5, 6, and ≥ 7, respectively.

Method	Split	Short (≤ 5)	Medium (6)	Long (≥ 7)
GraphGPO	Seen	90.37 ±1.59	91.65 ±4.34	90.02 ±7.14
GraphGPO + SERO	Seen	98.65 ±1.18	98.37 ±1.42	90.14 ±5.13
GraphGPO	Unseen	88.95 ±3.14	91.89 ±3.63	83.50 ±6.16
GraphGPO + SERO	Unseen	95.18 ±2.17	91.60 ±1.51	87.32 ±1.39

On the seen split, SERO improves short- and medium-horizon success by 8.28 and 6.72 points, while the long-horizon bucket remains nearly unchanged. On the unseen split, success improves by 6.23 points for short plans and 3.82 points for long plans, with similar performance on the medium bucket. These results show that the gains are not limited to short episodes: SERO also improves the more challenging long-horizon unseen tasks, where GraphGPO increases from 83.50 to 87.32.

B Notation and Action Supervision

We use s_t for the information available to the policy at decision step t: the current observation together with the interaction context exposed by the agent interface. Exact restoration may additionally use simulator state kept by the environment wrapper. Table 9 collects the notation used below.

B.1 Language Commands

At oracle step k, let the planner command contain tokens $y^\star_{k,1}, \ldots, y^\star_{k,m_k}$. If the actor produces a reasoning prefix before the command, we denote the prefix sampled from the frozen behavior policy by c_k. Both behavior-policy scoring and the auxiliary loss use the same replay context (s^O_k, c_k); for

Table 9: Notation used in the method and experiments.

Symbol	Meaning
s_t	Policy-visible state at step t, including the current observation and interaction context.
s_k^O	State on the oracle branch at step k; s_0^O is the recorded on-policy root.
π_b, π_θ	Frozen behavior policy and trainable policy.
$a_k^\star$	Executed oracle action at oracle step k.
ℓ_k, z_k	Trainable action loss and detached behavior-policy compatibility.
w_k	Policy-reachability proxy, $\prod_{j<k} z_j$.
G_r^π, G_r^O	Policy and oracle returns from the same restored root and remaining horizon.
g_r, v_r	Detached branch gate and branch-validity indicator.
L_r, N_{root}	Continuation length and number of attempted roots.
c_R	Reward scale used to normalize the branch-return gap.
λ_O	Weight of the auxiliary objective relative to the base RL loss.
$\text{sg}[\cdot]$	Stop-gradient operator.

an action-only policy, c_k is empty. We compute

$$\ell_k(\theta) = -\frac{1}{m_k} \sum_{j=1}^{m_k} \log \pi_\theta(y_{k,j}^\star \mid s_k^O, c_k, y_{k,<j}^\star), \tag{14}$$

$$z_k = \exp\left(\frac{1}{m_k} \sum_{j=1}^{m_k} \log \pi_b(y_{k,j}^\star \mid s_k^O, c_k, y_{k,<j}^\star) \right). \tag{15}$$

The same serialization and command-token mask are used for both terms. The sampled c_k is detached replay context, and no marginalization over reasoning prefixes is attempted. The geometric mean in Equation 15 therefore defines a length-normalized compatibility score rather than a calibrated marginal action probability or the probability of emitting the entire command string.

B.2 Policy-Reachability Interpretation

For categorical actions in a deterministic environment, $\prod_{j<k} \pi_b(a_j^\star \mid s_j^O)$ equals the probability of reproducing that oracle action prefix. More generally, it is an action-compatibility score along the realized oracle state sequence. For tokenized commands, $w_k = \prod_{j<k} z_j$ is the analogous length-normalized compatibility proxy, not the probability of emitting the full command prefix. The exclusive prefix is important: $w_0 = 1$, and low compatibility at step k reduces supervision only for later oracle actions through $w_{k+1} = w_k z_k$.

C Training Algorithm and Restoration

C.1 Root Records and Restoration

A root record contains the task instance, action prefix, current observation, remaining interaction budget, legal-action interface, agent-side history, and the realized return of the original trajectory from that root. Environments with serializable state are restored directly; otherwise the adapter resets the task and replays the recorded prefix. After restoration, the policy-visible prompt and admissible-action interface must match the original pre-action record before the oracle branch is executed; otherwise the branch is marked invalid.

The oracle continuation maintains its own environment history after the restored root. Later observations are produced by the oracle branch itself rather than copied from the original on-policy suffix.

C.2 Data and Gradient Separation

Primary RL data determine reward normalization, group statistics, graph construction, advantages, and policy ratios. Each backbone retains its native token-level policy-ratio computation; oracle ex-

Algorithm 1 One training iteration of SERO. Primary on-policy rollouts define the base RL update; selected roots then seed executable oracle continuations used by the auxiliary objective.

Require: Policy π_θ, base RL learner, task batch, oracle, coefficient λ_O, reward scale $c_R > 0$

Freeze behavior snapshot $\pi_b \leftarrow \pi_\theta$
Collect primary on-policy rollouts $\mathcal{D}_\pi$ and exact pre-action root records
Compute base advantages and statistics using only $\mathcal{D}_\pi$
Select eligible roots $\mathcal{R}$; set $N_{\text{root}} \leftarrow |\mathcal{R}|$ and $S \leftarrow 0$
for each attempted root $r \in \mathcal{R}$ **do**
    Restore the environment and policy-visible context in an isolated copy
    if restoration fails **then**
        Record the failed attempt and continue
    end if
    Set G_r^π to the realized return of the original on-policy suffix from this root
    Execute one oracle continuation under the original remaining horizon
    Record policy-visible states $s_{r,k}^O$, executed oracle actions $a_{r,k}^\star$, and branch outcome
    Compute detached branch gate g_r using Equation 7
    Set $w \leftarrow 1$
    for $k = 0, \ldots, L_r - 1$ **do**
        Sample $c_{r,k}$ once from π_b at $s_{r,k}^O$; use an empty prefix for action-only policies
        Store $(s_{r,k}^O, c_{r,k}, a_{r,k}^\star)$, its serialization, and the command-token mask
        Compute detached $z_{r,k}$ from the stored context and mask using Equation 15
        Compute $\ell_{r,k}(\theta)$ from the same stored context and mask using Equation 14
        $S \leftarrow S + \text{sg}[g_r]\ \text{sg}[w]\, \ell_{r,k}(\theta)$
        $w \leftarrow w z_{r,k}$
    end for
end for
$\mathcal{L}_{\text{SERO}} \leftarrow S / \max(N_{\text{root}}, 1)$
Update θ with $\mathcal{L}_{\text{base}}(\mathcal{D}_\pi) + \lambda_O \mathcal{L}_{\text{SERO}}$

amples are replayed only through the command-token loss $\ell_{r,k}(\theta)$, while g_r, $z_{r,k}$, and $w_{r,k}$ are detached. We normalize by attempted roots rather than total oracle steps so each root is one branching unit while longer valid continuations can contribute additional supervised decisions. The behavior snapshot is fixed while roots are collected and oracle actions are scored, and N_{root} is preserved across optimizer microbatches.

Implementation tests cover base-only parity at $\lambda_O = 0$, isolation between primary and oracle environments, exact root reconstruction, legal-action and horizon preservation, exclusive-prefix ordering, command-only masking, attempted-root normalization, and zero oracle access during evaluation.

D Task Protocols and Oracle Backends

D.1 ALFWorld

We use the text interface of ALFWorld (Shridhar et al., 2020). For an oracle branch, the adapter reconstructs the selected on-policy prefix and verifies the reconstructed prompt and admissible actions. It then obtains commands from the environment's expert-plan interface (`extra.expert_plan`), checks each command against the current admissible actions, and executes it through the ordinary environment interface. For each task group containing at least one failure, one failed rollout is selected and one of its pre-action turns is sampled uniformly as the root; all-success groups are skipped.

ALFWorld uses binary task success for the oracle gate, so Equation 7 becomes

$$g_r = v_r \left[\text{success}(\tau_r^O) - \text{success}(\tau_r^\pi)\right]_+, \qquad [x]_+ = \max(x, 0). \tag{16}$$

A valid oracle branch therefore receives a positive gate only when it improves the failed policy rollout from the same restored root. GraphGPO and HGPO retain their original ALFWorld wrappers;

within each backbone, the base and SERO variants share the same task manifest, prompt construction, action interface, and evaluation protocol.

D.2 WEBSHOP

WebShop (Yao et al., 2022a) requires search, navigation, option selection, and purchase actions. Root selection follows the ALFWorld rule: from each group with at least one failure, we sample one failed rollout and then one of its pre-action turns uniformly; all-success groups are skipped, and eligibility uses strict success rather than the dense task score. The training-time oracle is a deterministic rule-based policy using the target product identifier and required options from goal metadata. When search is needed, it tries the sanitized retrieval query, full product name, and its first eight words in order; it navigates to the target, selects the required options, and purchases it through the same legal `search`/`click` interface as the policy. Restoration reproduces the page/session state, navigation history, selected options, task identity, and observation renderer. We report both task score and strict success, while the oracle gate uses strict task success.

E EXPERIMENTAL CONFIGURATION

E.1 ALFWORLD TRAINING CONFIGURATION

Table 10 lists the ALFWorld settings used in the matched GraphGPO/SERO and HGPO/SERO comparisons. Within each backbone, the base and +SERO runs use the same task manifests, action interface, and decoding settings.

Table 10: ALFWorld training and evaluation configuration. Backbone-specific settings are shown explicitly; all base/+SERO comparisons are matched within backbone.

Item	Setting
Models	Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct.
Rollout batch	16 task groups $\times$ 8 rollouts per group.
GraphGPO prompt / response limit	2,048 / 512 tokens.
HGPO prompt / response limit	4,096 / 512 tokens.
Environment horizon	50 interaction steps.
Training / evaluation temperature	1.0 / 0.4.
Actor learning rate	10^{-6}.
Reference KL coefficient	0.01.
Invalid-action penalty	Coefficient 0.1.
GraphGPO credit settings	Graph discount 0.10; episode and step weights 1; mean/std normalization.
HGPO credit settings	Discount 0.95; length weighting with $\alpha_{\mathrm{HGPO}} = 1.0$; mean/std normalization; base-group disabled.
Default SERO coefficient	$\lambda_O = 0.1$.
Oracle roots	One root per eligible task group.
Compatibility score	Geometric mean of frozen behavior-policy probabilities over oracle command tokens.
Auxiliary target	Command-token NLL.
Final checkpoints	GraphGPO step 150; HGPO step 150.
Training hardware	2 GPUs for 1.5B and 4 GPUs for 7B.

E.2 WEBSHOP TRAINING CONFIGURATION

Table 11 summarizes the backbone-specific WebShop settings. Within each backbone, the base and +SERO runs keep the same horizon, prompt format, action interface, and final-checkpoint evaluation protocol.

Table 11: Backbone-specific WebShop settings used in the controlled comparisons.

Setting	GraphGPO	HGPO
Task groups × rollouts	16×8	16×8
Environment horizon	15	30
Prompt / response limit	5,120 / 512	4,096 / 512
PPO minibatch size	64	64
Backbone discount	0.20	0.95
Backbone normalization	mean/std	mean
HGPO length coefficient	–	$\alpha_{\mathrm{HGPO}} = 1.0$
Observation mode	text-rich	text
Final checkpoint	step 150	step 150
Evaluation temperature	0.4	0.4

F Ablation Definitions

F.1 Root-only and Full-Flat

The online mechanism ablation uses GraphGPO with Qwen2.5-1.5B-Instruct. Root-only keeps only the first oracle target, giving the per-root term $g_r \ell_{r,0}$. Full-Flat trains the full continuation but uses one shared branch weight,

$$\bar{w}_r = \frac{1}{L_r} \sum_{k=0}^{L_r-1} w_{r,k}, \qquad \mathcal{L}_{\mathrm{flat},r} = g_r \bar{w}_r \sum_{k=0}^{L_r-1} \ell_{r,k}. \tag{17}$$

SERO uses the same oracle branch but keeps the step-specific exclusive-prefix weights $w_{r,k}$. These three variants correspond directly to the ablation reported in Table 2.

F.2 Fixed-Bank Controls

The fixed-bank experiment in Appendix A.1 removes differences in collected oracle roots by replaying the same cached oracle branches from the same starting checkpoint. Flat, Depth decay, and SERO match the total supervision mass on every branch; their auxiliary objectives differ only in where that mass is assigned along the continuation.