GENERATE MULTIPLE ACTIONS, EXECUTE ONE: EXPERIENCE-GUIDED ACTION SELECTION FOR LLM AGENT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Group-in-Group Policy Optimization (GiGPO) improves credit assignment in reinforcement learning for LLM agents by comparing the returns of actions already executed at the same state. Comparing more candidate actions could improve action selection and provide additional learning signals. However, directly obtaining terminal feedback for each candidate requires completing a separate downstream trajectory, which involves many additional environment interactions. To avoid this cost, we introduce **E**xperience-**G**uided **A**ction **S**election (EGAS), a method that uses an experience pool of historical state–action outcomes to guide both action selection and policy learning. At the state, EGAS uses the experience pool to estimate whether an alternative action could be more likely to lead to task success than the initially generated action. When this is the case, EGAS generates additional candidates from the current policy and selects the action with the highest estimated probability of success among the initial action and candidates with sufficient historical support. Only the selected action is executed. EGAS then uses the experience pool to construct pairwise preferences between the selected action and unexecuted candidates. An auxiliary objective incorporates these preferences into the base RL objective, enabling the policy to learn from unexecuted candidates without requiring additional downstream trajectories. EGAS improves held-out task success over GiGPO by 1.7–7.9 percentage points across ALFWorld, WebShop, and Sokoban. It also reaches GiGPO's best-observed performance while using 7.3–25.0% fewer completed training trajectories. These results demonstrate that comparing and learning from more actions need not require executing more trajectories.



Figure 1: **EGAS compares multiple candidate actions but executes only one.** *Left:* A linear rollout generates and executes one action per state visit, whereas a tree rollout expands multiple environment branches. EGAS selectively generates additional candidates and uses the experience pool to select one for execution, without separate downstream trajectories for the others. *Right:* Validation success on ALFWorld with Qwen3-1.7B-Instruct. EGAS reaches 60.2% after $17{,}920$ completed training trajectories, compared with 52.3% after $18{,}560$ for GiGPO.

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.

1 INTRODUCTION

Large language models (LLMs) can operate as agents by interleaving reasoning, actions, and observations (Yao et al., 2023b), enabling them to tackle tasks in interactive environments such as ALFWorld and WebShop (Shridhar et al., 2021; Yao et al., 2022). Feedback and accumulated experience can improve subsequent decisions (Shinn et al., 2023; Zhao et al., 2024), while reinforcement learning (RL) optimizes agent policies based on task outcomes (Feng et al., 2026a;b). However, sparse terminal rewards make it difficult to assign credit to individual actions in long-horizon tasks. Group-in-Group Policy Optimization (GiGPO) addresses this credit-assignment problem by grouping actions already executed at the same state across collected trajectories and comparing their returns (Feng et al., 2026b). These comparisons yield step-level advantages that complement trajectory-level advantages.

Extending these comparisons to additional candidate actions at the current state could improve action selection and provide additional learning signals. A standard rollout, however, generates and executes only one action per state visit. Branching methods such as Tree-GRPO and SPARK illustrate the value of exploring alternative actions for policy learning (Ji et al., 2026; Wu et al., 2026). They expand multiple rollout branches from a shared state, as shown schematically in Figure 1 (left). Under sparse terminal feedback, directly obtaining a terminal outcome for each candidate requires completing a separate downstream trajectory. In long-horizon tasks, each such trajectory can involve many environment interactions, making it costly to evaluate every candidate.

RESEARCH QUESTION

Can an LLM agent compare and learn from multiple candidate actions at a given state while executing only one of them in the environment?

We introduce EGAS (Experience-Guided Action Selection), which uses an experience pool of executed state–action pairs and their terminal task outcomes to guide both action selection and policy learning. At each state, the policy first generates an initial action. The pool estimates whether an alternative could be more likely to lead to task success than this action. When the estimate indicates potential improvement, EGAS generates additional candidates from the current policy. Among the initial action and candidates with sufficient historical support, it selects the action with the highest estimated probability of task success. Only the selected action is executed, as illustrated in Figure 1 (left). The pool provides evidence for comparing candidates but does not supply the candidates themselves.

The experience pool also guides policy learning. In our main experiments, we use GiGPO as the base RL algorithm (Feng et al., 2026b) and weight its advantages according to the estimated potential improvement over the initial actions. This emphasizes learning from states where historical experience suggests more room to improve action selection. For unexecuted candidates, EGAS uses experience-pool evidence to construct pairwise preferences between each candidate and the selected action. An auxiliary pairwise preference objective supplements the base RL objective, allowing the policy to learn from these candidates without requiring their own downstream trajectories.

Across ALFWorld, WebShop, and Sokoban with 1.7B and 4B models, EGAS improves held-out success over controlled GiGPO baselines by 1.7–7.9 percentage points and reaches GiGPO's best-observed performance using 7.3–25.0% fewer completed training trajectories. Figure 1 (right) shows validation success against completed training trajectories on ALFWorld. We report both training-trajectory efficiency and rollout-collection time.

Our contributions are:

- We introduce EGAS, which uses an experience pool to decide when to generate additional candidates from the current policy and to select a single action for execution.
- We introduce an auxiliary pairwise preference objective that uses experience-pool evidence to compare unexecuted candidates with the selected action, enabling learning from these candidates without additional downstream trajectories.
- Across three benchmarks, EGAS improves held-out success over GiGPO by 1.7–7.9 percentage points and reaches GiGPO's best-observed performance using 7.3–25.0% fewer completed training trajectories.

2 Related Work

Reinforcement learning for language agents. Language agents have been studied in embodied environments (Huang et al., 2022; Yao et al., 2023b), web interaction (Yao et al., 2022; Deng et al., 2023; Zhou et al., 2024b), and tool use (Qin et al., 2024). AgentBench evaluates agents across interactive environments, while AgentTuning trains agent capabilities from interaction trajectories (Liu et al., 2024; Zeng et al., 2024). Outcome-based RL further updates policies from completed interactions (Yao et al., 2024; Feng et al., 2026a;b). GRPO estimates group-relative advantages without a value model (Shao et al., 2024); GiGPO combines trajectory-level and repeated-state step-level advantages to improve credit assignment (Feng et al., 2026b).

Rollout allocation and candidate-action selection. Deliberative methods compare or refine alternative reasoning paths through tree search, world-model planning, or self-feedback (Yao et al., 2023a; Hao et al., 2023; Zhou et al., 2024a; Madaan et al., 2023). Adaptive rollout methods allocate sampling across tasks, states, or prefixes based on estimated success or informativeness (Chen et al., 2026; Han et al., 2026; Zou et al., 2026), while Tree-GRPO and SPARK expand selected branches to compare alternatives through multiple downstream outcomes (Ji et al., 2026; Wu et al., 2026). Candidate-level methods compare actions before commitment: SAND uses self-consistency sampling and execution-guided critiques to train action deliberation, whereas VeGAS samples multiple actions at test time and selects one with a separately trained verifier (Xia et al., 2025; Singhi et al., 2026). EGAS instead uses historical state–action outcomes to select an action for execution.

Experience reuse and memory for language agents. Language agents reuse prior trials through verbal reflection, retrieved experience, long-term memory, skill libraries, retrospective models, or distilled skills (Shinn et al., 2023; Zhao et al., 2024; Park et al., 2023; Wang et al., 2024; Yao et al., 2024; Xia et al., 2026; Xie et al., 2026), and their behavior can depend strongly on how experience is stored and retrieved (Xiong et al., 2026). For action-level adaptation, ReAd learns an advantage critic from prior trajectories to refine plans, while OLIVIA updates an inference-time decision layer from action-level feedback (Zhang et al., 2025; Yu et al., 2026). These approaches typically provide textual guidance or use a learned value or decision model.

3 Method

3.1 Problem Setup and Overview

For task u, the policy conditions on $x_t = (u, h_t, o_t)$, where h_t is the interaction history and o_t is the current observation. At training iteration k, the policy samples an action $a_t^{(0)} \sim \pi_{\theta_k}(\cdot \mid x_t)$, which we call the initially generated action. A completed trajectory τ receives terminal outcome $R(\tau) \in \{0, 1\}$. A standard rollout executes this action without generating additional candidates at that state visit.

EGAS uses historical outcomes to guide action selection and policy learning. Within each rollout group, a fixed number of trajectories use EGAS, while the remaining trajectories follow standard rollouts. During rollout collection, the frozen policy π_{θ_k} consults an experience pool $\mathcal{M}_k$ constructed from previously executed state–action outcomes. The pool estimates whether an alternative could improve the probability of task success relative to the initially generated action. When the estimate indicates potential improvement, EGAS samples additional candidates from the same policy and selects the action with the highest estimated success probability among the initial action and additional candidates with sufficient historical support. Only the selected action is executed. Otherwise, the initially generated action is executed. Executed trajectories are used to update the policy through a weighted base RL objective and to update the experience pool; unexecuted candidates contribute through an auxiliary pairwise preference objective. Figure 2 summarizes this process, with full algorithmic and key-construction details deferred to Appendices A and C.4.

3.2 Experience-Guided Action Selection

The experience pool summarizes the task outcomes observed after previously executed actions. Each entry corresponds to a task u, a canonical state identifier z_t derived from the current environment observation, and an executable action a. It stores success and failure evidence only from trajectories

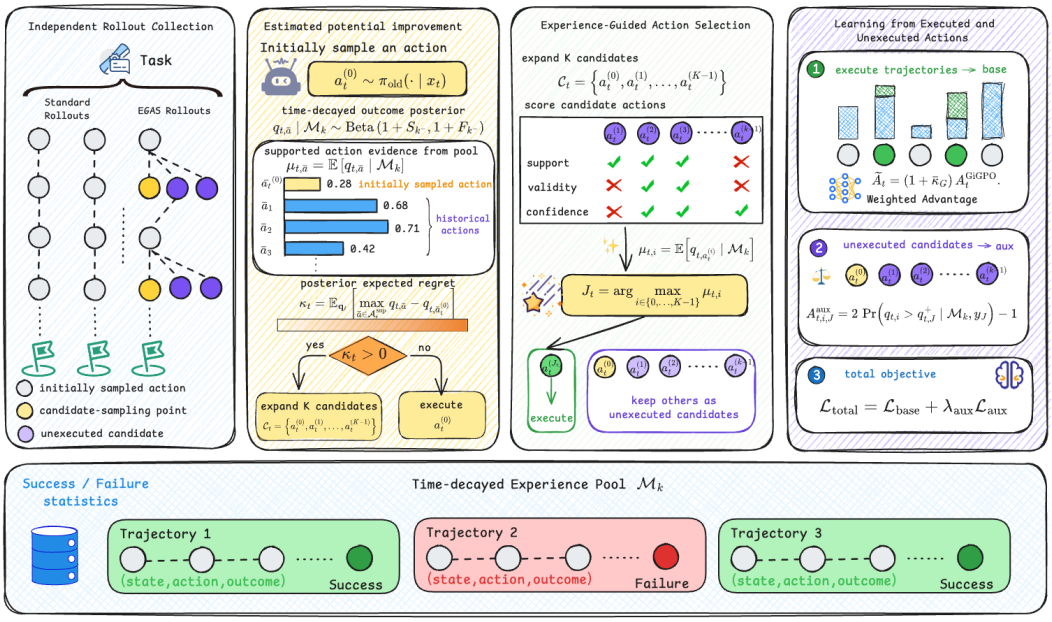

Figure 2: **Overview of EGAS.** Historical outcomes guide selective candidate generation and action selection. Only one action is executed at each state. Executed trajectories are used for weighted RL updates and experience-pool updates, while unexecuted candidates contribute through an auxiliary pairwise preference objective.

in which that action was actually executed. Because the policy evolves during training, outcomes collected under earlier policies may become less representative of its current behavior. We therefore apply exponential time decay to the stored evidence, giving greater weight to recent experience. Let $\widetilde{S}_k(u, z_t, a)$ and $\widetilde{F}_k(u, z_t, a)$ denote the time-decayed success and failure counts. We use a Beta posterior to model historical evidence about the probability of task success after executing a at state z_t for task u:

$$q(u, z_t, a) \mid \mathcal{M}_k \sim \mathrm{Beta}\Big(1 + \widetilde{S}_k(u, z_t, a),\ 1 + \widetilde{F}_k(u, z_t, a)\Big)\,. \tag{1}$$

Its posterior mean provides an experience-based success estimate for action selection. We approximate the joint posterior across distinct action keys by independent Beta distributions. Exact key construction and the decay-and-commit update are given in Appendix C.4.

Before generating additional candidates, EGAS uses historical outcomes to assess how much a supported alternative could improve task success relative to the initial action. Let $\mathcal{A}_t^{\mathrm{sup}}$ contain actions for task u at state z_t whose total time-decayed success and failure counts meet a minimum support threshold. Let $q_{t,a}$ abbreviate the corresponding task-success probability in Eq. (1). We define the potential improvement over $a_t^{(0)}$ as

$$\kappa_t = \mathbb{E}\left[\left(\max_{a \in \mathcal{A}_t^{\mathrm{sup}}} q_{t,a} - q_{t,a_t^{(0)}}\right)_+ \,\middle|\, \mathcal{M}_k\right]. \tag{2}$$

Here, $(v)_+ = \max(v, 0)$. We set $\kappa_t = 0$ when the initially generated action lacks sufficient support or no distinct supported alternative is available. Thus, κ_t measures the posterior expected positive improvement over the initial action among historically supported actions, accounting for uncertainty in their success probabilities.

When $\kappa_t = 0$, EGAS executes the initial action without generating additional candidates. At a decision with $\kappa_t > 0$, EGAS retains $a_t^{(0)}$ and samples $K - 1$ additional candidate actions independently from the current policy, forming the candidate list $\mathcal{C}_t = (a_t^{(0)}, \ldots, a_t^{(K-1)})$. The experience pool scores the generated candidates rather than supplying actions. Only candidates with sufficient historical

support are eligible to replace the initial action. Let $\mathcal{I}_t = \{0\} \cup \{i \in \{1, \ldots, K-1\} : a_t^{(i)} \in \mathcal{A}_t^{\text{sup}}\}$ index the initial action and eligible alternatives. EGAS selects by posterior mean:

$$\mu_{t,i} = \mathbb{E}\Big[q_{t,a_t^{(i)}} \mid \mathcal{M}_k\Big], \qquad J_t = \arg\max_{i \in \mathcal{I}_t} \mu_{t,i}. \tag{3}$$

If no alternative has sufficient support, $\mathcal{I}_t = \{0\}$ and the initial action is retained. Only $a_t^{(J_t)}$ is executed in the environment. The remaining candidates are not executed and receive no separate downstream rollouts. EGAS therefore broadens the set of actions considered by the agent while preserving single-path environment execution.

3.3 LEARNING FROM EXECUTED AND UNEXECUTED ACTIONS

EGAS maintains a strict distinction between executed and unexecuted actions. After a trajectory terminates, its observed outcome $y = R(\tau)$ is used to compute base RL advantages and update the experience-pool entries of its executed state–action pairs. Unexecuted candidates have no observed terminal outcomes and do not update the experience pool. Our primary instantiation uses GiGPO, which combines trajectory-level and state-level relative advantages to improve credit assignment over executed trajectories (Feng et al., 2026b). The same action-selection framework can be applied to GRPO, which we evaluate as a supplementary setting (Shao et al., 2024).

We also use the estimated improvement to weight GiGPO advantages within each state group, using the κ_t values computed before action selection. For a GiGPO state group G, let $\mathcal{S}_G$ contain the visits with sufficient experience-pool support. We define

$$\bar{\kappa}_G = \begin{cases} |\mathcal{S}_G|^{-1} \sum_{t \in \mathcal{S}_G} \kappa_t, & \mathcal{S}_G \neq \varnothing, \\ 0, & \mathcal{S}_G = \varnothing, \end{cases} \qquad \widetilde{A}_t = (1 + \bar{\kappa}_G) A_t^{\text{GiGPO}}. \tag{4}$$

Here, A_t^{GiGPO} denotes the combined GiGPO advantage before experience-based weighting. The shared positive weight preserves the signs and relative ordering of advantages within each state group, while giving greater weight to groups with larger posterior expected improvements. Appendix B.1 gives the corresponding derivation.

EGAS constructs auxiliary comparisons only when the trajectory containing the selected action succeeds, i.e., $y = R(\tau) = 1$. For the selected action $a_t^{(J)}$, where $J = J_t$, we form a temporary posterior incorporating this outcome: $q_{t,J}^+ \mid \mathcal{M}_k, y = 1 \sim \text{Beta}(\alpha_{t,J} + 1, \beta_{t,J})$, where $(\alpha_{t,J}, \beta_{t,J})$ are taken from the experience pool. Auxiliary comparisons consider candidates with $i \neq J_t$, including the initial action when it is replaced, without an additional historical-support requirement. For each such candidate $a_t^{(i)}$, write $q_{t,i} = q(u, z_t, a_t^{(i)})$. We then construct the soft pairwise preference signal

$$A_{t,iJ}^{\text{aux}} = 2 \Pr(q_{t,i} > q_{t,J}^+ \mid \mathcal{M}_k, y = 1) - 1. \tag{5}$$

Positive and negative values respectively encourage and discourage the unexecuted candidate, while values near zero provide little directional preference. The selected action serves as the comparison reference and is trained through the base RL objective.

For clarity, we present the unclipped surrogate losses:

$$\mathcal{L}_{\text{base}}(\theta) = -\mathbb{E}_{\text{base}}\Big[r_{t,\ell}(\theta)\, \widetilde{A}_t\Big], \qquad \mathcal{L}_{\text{aux}}(\theta) = -\mathbb{E}_{\text{aux}}\big[r_{t,i,\ell}(\theta)\, A_{t,iJ}^{\text{aux}}\big]. \tag{6}$$

Here, r denotes the token-level probability ratio between the current policy and the frozen policy used to generate candidates, and ℓ indexes tokens. The two expectations separately average over valid response tokens in executed trajectories and executable-action tokens of unexecuted candidates, respectively. In practice, we use a clipped policy objective (Schulman et al., 2017) with dual clipping for negative advantages (Ye et al., 2020); Appendix C.1 gives the clipped losses and parameters. The total training objective combines the two losses:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{base}} + \lambda_{\text{aux}} \mathcal{L}_{\text{aux}}, \qquad \lambda_{\text{aux}} = \frac{\lambda_0}{K}. \tag{7}$$

In the primary experiments, $\mathcal{L}_{\text{base}}$ uses the weighted GiGPO advantages defined above. Here, λ_0 controls the base strength of auxiliary learning. We use a small auxiliary coefficient to prioritize learning from observed trajectory outcomes. The two losses are normalized separately so that auxiliary tokens do not change the normalization of the base objective. Appendix B.3 specifies the coefficient scaling, and Appendix A presents the complete training procedure.

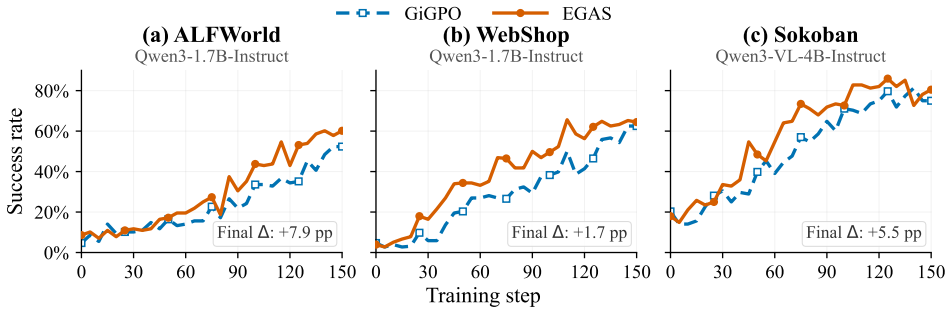

Figure 3: Held-out success for GiGPO and EGAS on ALFWorld, WebShop, and Sokoban.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Benchmarks and backbones. We evaluate on ALFWorld, WebShop, and Sokoban. ALFWorld tests long-horizon household planning through text interaction (Shridhar et al., 2021); WebShop requires search, attribute grounding, and navigation in a simulated product catalog (Yao et al., 2022); and Sokoban provides procedurally generated planning problems in which local moves can be irreversible (Racanière et al., 2017; Schrader, 2018). The text-based experiments use Qwen3-1.7B-Instruct and Qwen3-4B-Instruct-2507 (Yang et al., 2025), while Sokoban uses Qwen3-VL-4B-Instruct (Bai et al., 2025).

Baselines and controlled training. The main comparison covers prompting, group-relative RL, skill-augmented RL, and self-distillation baselines. GiGPO is our primary baseline. EGAS extends it with experience-guided action selection, advantage weighting, and auxiliary preference learning. Each controlled comparison uses the same backbone, training budget, and evaluation protocol; GRPO+EGAS additionally tests compatibility with another base RL objective.

Training and evaluation protocol. We evaluate frozen checkpoints on held-out tasks, measuring the learned policy without test-time action selection. The primary metric is task success, with normalized reward additionally reported for WebShop. We measure trajectory efficiency by completed training trajectories entering the main policy update. Three-seed results and full protocol details are provided in Appendices G and C.

4.2 EGAS IMPROVES THE LEARNED POLICY

EGAS improves learned-policy performance across environments and model scales. Table 1 and Figure 4 show that, relative to GiGPO, EGAS improves success by 6.3–7.9 percentage points on ALFWorld, 1.7–2.5 points on WebShop, and 5.5 points on Sokoban. GRPO+EGAS also improves the aggregate metrics over GRPO, demonstrating compatibility with another base RL objective.

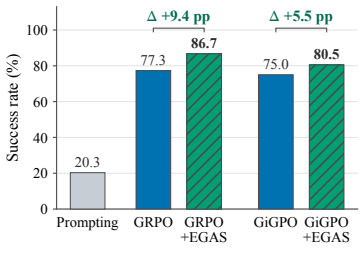

Figure 4: Sokoban success with Qwen3-VL-4B-Instruct.

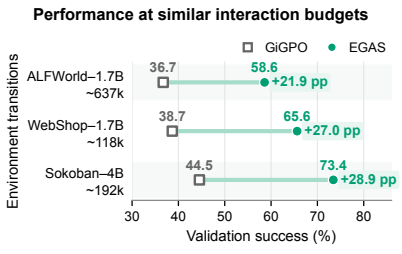

Figure 5: Validation success at nearly matched environment-transition budgets.

Table 1: ALFWorld success (%) by task type and on average, and WebShop normalized score and success (%). Pink bold and blue underlining denote the best and second-best results within each model block; colored values beside aggregate EGAS results give absolute percentage-point changes over the corresponding base objective.

Type	Method	ALFWorld success (%)							WebShop	
		Pick	Look	Clean	Heat	Cool	Pick2	Avg.	Score	Succ.
Qwen3-1.7B-Instruct										
Prompting	Vanilla	25.0	22.2	3.1	0.0	21.4	4.2	12.5	46.5	4.7
	Skill-Prompt* (Lu et al., 2026)	10.3	50.0	16.1	0.0	0.0	5.0	9.4	23.0	2.3
RL Training	GRPO (Shao et al., 2024)	60.0	9.1	33.3	15.0	17.4	20.8	30.5	69.9	47.7
	Skill-GRPO (Lu et al., 2026)	27.6	54.5	22.7	27.3	0.0	19.2	21.1	73.4	46.1
	GRPO+OPSD (Lu et al., 2026)	38.2	50.0	30.8	28.6	30.0	21.1	32.0	70.7	38.3
	Skill-SD (Wang et al., 2026)	52.9	37.5	69.2	42.9	**60.0**	**36.8**	52.3	**81.8**	53.9
	RLSD (Yang et al., 2026)	50.0	37.5	61.5	19.0	50.0	21.1	42.2	74.0	50.8
	SDAR (Lu et al., 2026)	73.5	25.0	**76.9**	33.3	40.0	**36.8**	53.9	76.8	58.6
	GiGPO (Feng et al., 2026b)	75.0	**55.6**	62.5	7.7	48.4	35.7	52.3	80.6	62.8
	GRPO+EGAS	57.1	9.1	73.3	45.0	26.1	25.0	41.4 +10.9	72.4 +2.5	48.1 +0.4
	GiGPO+EGAS	**80.0**	45.5	73.9	**47.4**	53.3	30.0	**60.2** +7.9	80.2 -0.4	**64.5** +1.7
Qwen3-4B-Instruct-2507										
Prompting	Vanilla	53.3	0.0	0.0	15.4	3.7	0.0	14.8	30.6	8.6
RL Training	GRPO (Shao et al., 2024)	75.0	27.8	25.0	15.4	25.8	28.6	35.9	58.8	49.6
	SkillRL (Xia et al., 2026)	90.0	63.6	**92.3**	42.9	52.0	40.9	67.2	89.4	75.7
	GiGPO (Feng et al., 2026b)	82.8	**69.2**	40.0	**68.8**	**62.5**	65.5	64.8	86.1	78.0
	GRPO+EGAS	78.6	33.3	41.7	46.2	41.9	57.1	48.4 +12.5	83.1 +24.3	65.6 +16.0
	GiGPO+EGAS	**100.0**	53.9	56.5	58.3	61.3	**76.2**	**71.1** +6.3	**89.6** +3.5	**80.5** +2.5

EGAS reaches GiGPO's best-observed performance in fewer training steps. Figure 3 also shows higher final success across all three environments.

4.3 EGAS USES TRAINING TRAJECTORIES MORE EFFECTIVELY

EGAS learns more effectively from completed training trajectories. Table 2 reports normalized area under the curve (AUC) over a common training-trajectory budget and the number of trajectories needed to first reach GiGPO's best-observed success. Across all five settings, EGAS improves normalized AUC by 2.9–10.7 percentage points and reaches the target with 7.3–25.0% fewer trajectories. Over the same number of training updates, cumulative rollout-collection time increases by a median of 10.8% across settings (range: 2.7–20.4%).

EGAS achieves higher success at similar interaction budgets. Because trajectories can differ in length, Figure 5 also compares checkpoints whose cumulative executed environment transitions differ by less than 1%. At approximately 637k ALFWorld, 118k WebShop, and 192k Sokoban

Table 2: Training-trajectory efficiency. Normalized AUC is the area under the held-out success curve plotted against cumulative completed training trajectories, divided by the shared trajectory budget of each GiGPO–EGAS pair. $s^\star$ denotes GiGPO's best-observed success; Reduction is the relative decrease in completed training trajectories required to first reach $s^\star$; and Rollout-time Δ is the relative change in cumulative rollout-collection time over the same number of training updates.

Setting	$s^\star$ (%)	Normalized AUC (%)			Trajectories to $s^\star$ (k)			Rollout-time Δ (%)
		GiGPO	EGAS	Δ (pp)	GiGPO	EGAS	Reduction (%)	
ALFWorld–1.7B	52.3	23.4	31.4	+8.0	18.6	14.1	+24.1	+2.7
ALFWorld–4B	68.0	40.3	43.2	+2.9	17.9	16.6	+7.3	+10.8
WebShop–1.7B	62.8	27.9	38.6	+10.7	18.6	14.1	+24.1	+13.0
WebShop–4B	82.4	61.1	67.2	+6.1	16.6	13.4	+19.2	+5.3
Sokoban–4B	81.3	49.5	57.9	+8.3	35.8	26.9	+25.0	+20.4

transitions, EGAS improves validation success over GiGPO by 21.9, 27.0, and 28.9 percentage points, respectively.

4.4 HISTORICAL EXPERIENCE SUPPORTS SELECTIVE ACTION REPLACEMENT

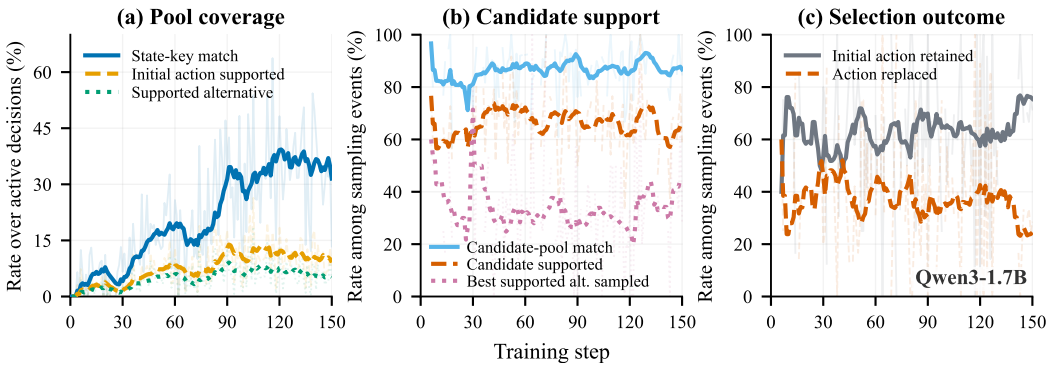

Figure 6: Experience-guided selection dynamics on ALFWorld with Qwen3-1.7B-Instruct. (a) Coverage of active decisions by the experience pool. (b) Candidate support among decisions where additional candidates are sampled. (c) The proportions of these decisions that retain or replace the initially sampled action.

Experience-pool coverage increases during training. Figure 6(a) shows that the state-key match rate generally increases, along with the fractions of decisions with historical support for the initial action or a distinct alternative. Among decisions where additional candidates are generated, candidate matching and support rates remain high (panel b). Thus, newly generated candidates often have historical outcome evidence available for comparison.

Generating additional candidates does not always lead to action replacement. Figure 6(c) shows both retention and replacement of the initial action after additional candidates are generated. This is consistent with selecting among the initial action and historically supported alternatives, rather than replacing the initial action whenever sampling is expanded.

4.5 ABLATIONS AND SENSITIVITY ANALYSIS

Both experience-guided action selection and auxiliary learning contribute to the gains. In Figure 7(a), random candidate selection causes the largest drop relative to full EGAS, at 16.4 percentage points, while randomly choosing candidate-sampling states reduces success by 5.5 points. Removing the auxiliary preference objective lowers success by 10.9 points, showing its contribution beyond action selection and advantage weighting. Removing advantage weighting or time decay reduces success by 7.8 and 7.0 points, respectively. These results support using historical outcomes both to guide action selection and to construct learning signals for unexecuted candidates.

Applying EGAS to more trajectories does not consistently improve success. Figure 7(b) shows that applying EGAS to four of the eight trajectories reaches 71.1% success with 10.8% rollout-collection overhead, whereas six reach the highest success of 73.4% but raise the overhead to 24.4%. Applying EGAS to all eight lowers success to 60.9% with 24.2% overhead. Applying experience-guided selection to every trajectory may concentrate executed actions on choices favored by the pool, reducing the diversity of collected experience. Retaining standard rollouts may mitigate this effect. We use four EGAS-enabled trajectories by default, as increasing this number to six gains 2.3 percentage points in success but more than doubles the rollout-collection overhead.

$K = 8$ and $\lambda_0 = 0.05$ achieve the highest success in their respective sweeps. Figure 7(c) shows that $K = 8$ reaches 60.2% success, compared with 45.3% at $K = 4$ and 51.6% at $K = 16$. This sweep fixes $\lambda_0 = 0.05$, so increasing K changes both the candidate count and the auxiliary coefficient $\lambda_{\text{aux}} = \lambda_0/K$. The decline at $K = 16$ may reflect a weaker auxiliary learning signal that is not

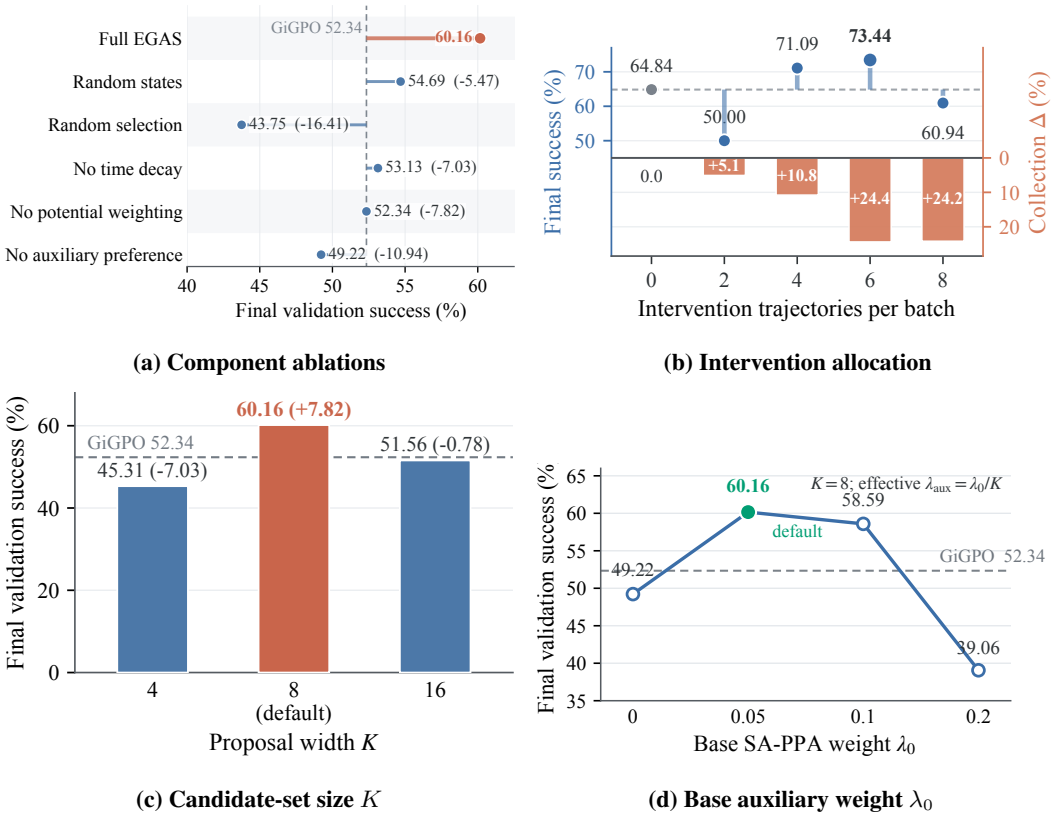

Figure 7: Ablations and sensitivity analysis on ALFWorld. (a) Component ablations. (b) Final success and rollout-collection overhead as the number of EGAS-enabled trajectories varies. (c) Candidate-set-size sensitivity. (d) Sensitivity to the base auxiliary weight, with $\lambda_{\text{aux}} = \lambda_0/K$. Panels (a), (c), and (d) use Qwen3-1.7B-Instruct, while Panel (b) uses Qwen3-4B-Instruct-2507. Parenthesized changes are relative to full EGAS in (a) and GiGPO in (c).

offset by gains from additional candidates. At $K = 8$, increasing λ_0 from 0.05 to 0.2 reduces success from 60.2% to 39.1% (Figure 7(d)). A larger auxiliary coefficient may amplify errors in preferences inferred from historical outcomes, reducing the benefit of auxiliary learning.

5 Conclusion

We introduced EGAS to compare and learn from multiple candidate actions while executing only one at each state visit. Historical state–action outcomes guide action selection and advantage weighting, while an auxiliary preference objective enables learning from unexecuted candidates. Across ALFWorld, WebShop, and Sokoban, EGAS improves held-out task success over GiGPO by 1.7–7.9 percentage points and reaches GiGPO's best-observed performance with fewer completed training trajectories. These findings show that historical experience can extend policy learning beyond executed actions without requiring a separate downstream trajectory for every candidate.

Reproducibility Statement

Section 3 specifies the EGAS method and learning objectives, and Section 4.1 describes the benchmarks, baselines, and evaluation protocol. The appendix provides the complete training algorithm (Appendix A), implementation settings and dataset splits (Appendix C), experience-pool key construction and normalization (Appendix C.4), environment prompts (Appendix D), and multi-seed results (Appendix G). An anonymized code repository is available at `https://anonymous.4open.science/r/EGAS-2E9F`.

AI USE STATEMENT

Generative AI tools were used to assist with language editing and literature search, and to generate parts of the code. All AI-assisted material was manually reviewed and verified by the authors, who take full responsibility for the technical content, experiments, and reported results.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Yuhan Chen, Yuxuan Liu, Long Zhang, Pengzhi Gao, Jian Luan, and Wei Liu. Step: Success-rate-aware trajectory-efficient policy optimization. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 30681–30692, 2026.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. In *International Conference on Learning Representations*, volume 2026, pp. 37909–37926, 2026a.

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, 2026b.

Yu Han, Kailing Li, Yang Jiao, Yulin Dai, Yuqian Fu, Linhai Zhuo, and Tianwen Qian. 3spo: State-score-supervised policy optimization for llm agents. *arXiv preprint arXiv:2606.09961*, 2026.

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

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pp. 9118–9147. PMLR, 2022.

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for llm agent reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 87362–87388, 2026.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, et al. Self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2605.15155*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, volume 2024, pp. 9695–9717, 2024.

Sébastien Racanière, Théophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

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

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. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. URL `https://arxiv.org/abs/2010.03768`.

Nishad Singhi, Christian Bialas, Snehal Jauhri, Vignesh Prasad, Georgia Chalvatzaki, Marcus Rohrbach, and Anna Rohrbach. Think twice, act once: Verifier-guided action selection for embodied agents. *arXiv preprint arXiv:2605.12620*, 2026. URL `https://arxiv.org/abs/2605.12620`.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL `https://openreview.net/forum?id=ehfRiF0R3a`.

Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, and Honggang Qi. Skill-sd: Skill-conditioned self-distillation for multi-turn llm agents, 2026. URL `https://arxiv.org/abs/2604.10674`.

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

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, et al. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026.

Yu Xia, Yiran Jenny Shen, Junda Wu, Tong Yu, Sungchul Kim, Ryan A Rossi, Lina Yao, and Julian McAuley. Sand: Boosting llm agents with self-taught action deliberation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 3062–3077, 2025.

Can Xie, Yuyi Zhou, Wen Yang, Ziyi zhang, Siyao Song, Yingzhuo Deng, Shuo Ren, and Jiajun Zhang. Edge: Experience-distillation for guided exploration in agentic reinforcement learning, 2026. URL `https://arxiv.org/abs/2608.21946`.

Zidi Xiong, Yuping Lin, Wenya Xie, Pengfei He, Zirui Liu, Jiliang Tang, Himabindu Lakkaraju, and Zhen Xiang. How memory management impacts llm agents: An empirical study of experience-following behavior. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 623–645, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled rlvr. *arXiv preprint arXiv:2604.03128*, 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, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023a.

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

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Ramapura Narasimha Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, et al. Retroformer: Retrospective large language agents with policy gradient optimization. In *International Conference on Learning Representations*, volume 2024, pp. 10091–10111, 2024.

Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, et al. Mastering complex control in moba games with deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 6672–6679, 2020.

Sheldon Yu, Junda Wu, Xintong Li, Nikki Lijing Kuang, Sizhe Zhou, Tong Yu, Jiawei Han, Jingbo Shang, and Julian McAuley. Olivia: Online learning via inference-time action adaptation for decision making in llm react agents. *arXiv preprint arXiv:2605.11169*, 2026.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3053–3077, 2024.

Yang Zhang, Shixin Yang, Chenjia Bai, Fei Wu, Xiu Li, Zhen Wang, and Xuelong Li. Towards efficient llm grounding for embodied multi-agent collaboration. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 1663–1699, 2025.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *Forty-first International Conference on Machine Learning*, 2024a. URL https://openreview.net/forum?id=njwv9BsGHF.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024b.

Heming Zou, Qi Wang, Yun Qu, Yuhang Jiang, Lizhou Cai, Yixiu Mao, Ru Peng, Xin Xu, Weijie Liu, Kai Yang, et al. Trace: A unified rollout budget allocation framework for efficient agentic reinforcement learning. *arXiv preprint arXiv:2606.11119*, 2026.

TABLE OF CONTENTS

A Complete Training Algorithm **15**

B Properties and Design Analysis **16**

B.1 Properties of Potential-Improvement Weighting 16

B.2 Posterior Interpretation of the Auxiliary Preference Signal 17

B.3 Auxiliary Weight Scaling 17

B.4 What Time Decay Controls 17

C Implementation Details **18**

C.1 Training Details 18

C.2 Evaluation Protocol 19

C.3 Dataset Splits 19

C.4 Experience-Pool Key Construction and Normalization 19

D Environment Prompts **20**

D.1 ALFWorld 20

D.2 WebShop 20

D.3 Sokoban 21

E Illustrative Task and Rollout Trace **21**

F Current-Policy Candidates versus Retrieved Historical Actions **22**

G Multi-Seed Results **24**

A COMPLETE TRAINING ALGORITHM

Algorithm 1 summarizes the complete training loop for the primary GiGPO+EGAS instantiation. It specifies how standard and EGAS-enabled rollouts are collected, how one candidate action is selected for execution, how unexecuted candidates are retained for auxiliary learning, and when executed outcomes update the experience pool.

Algorithm 1 Primary GiGPO+EGAS Training Procedure

Require: Training tasks $\mathcal{D}$; initial policy π_{θ_0}; experience pool $\mathcal{M}_0$; updates T; candidate-set size K; minimum support m; maximum action replacements B_{int} per EGAS-enabled trajectory; base auxiliary weight λ_0

Ensure: Trained policy π_{θ_T}

for $k = 0, \dots, T-1$ **do**
  $\pi_{\text{old}} \leftarrow \pi_{\theta_k}$; freeze π_{old} and keep $\mathcal{M}_k$ read-only
  $\lambda_{\text{aux}} \leftarrow \lambda_0 / K$
  1. Standard rollout collection
  $(\mathcal{B}_{\text{exec}}, \mathcal{P}_k) \leftarrow \text{COLLECTSTANDARD}(\pi_{\text{old}}, \mathcal{D})$ ▷ execute each initially sampled action
  $\mathcal{U}_k \leftarrow \varnothing$ ▷ candidate records for auxiliary learning
  2. EGAS-enabled rollout collection
  for all EGAS-enabled trajectories sampled from $\mathcal{D}$ **do**
    initialize trajectory $\tau \leftarrow \varnothing$ and used budget $b \leftarrow 0$
    while τ is active **do**
      observe x_t and its environment anchor o_t^{anchor}; compute $z_t = \phi(o_t^{\text{anchor}})$
      sample $r_t^{(0)} \sim \pi_{\text{old}}(\cdot \mid x_t)$ and extract $a_t^{(0)} = \text{Exec}(r_t^{(0)})$
      form $\mathcal{A}_t^{\text{sup}}$ and compute κ_t from $\mathcal{M}_k$ using Eq. equation 2
      $J_t \leftarrow 0$ and $\mathcal{C}_t \leftarrow \{a_t^{(0)}\}$
      if $\kappa_t > 0$ and $b < B_{\text{int}}$ **then**
        sample $r_t^{(1)}, \dots, r_t^{(K-1)}$ from $\pi_{\text{old}}(\cdot \mid x_t)$
        extract and deduplicate executable actions to form $\mathcal{C}_t$
        compute candidate posterior means and select J_t using Eq. equation 3
      end if
      execute only $a_t^{(J_t)}$ and append the observed transition to τ
      for all i such that $a_t^{(i)} \in \mathcal{C}_t$ and $i \neq J_t$ **do**
        add $(x_t, z_t, r_t^{(i)}, a_t^{(i)}, a_t^{(J_t)}, \text{id}(\tau))$ to $\mathcal{U}_k$
      end for
      if $J_t \neq 0$ **then**
        $b \leftarrow b + 1$
      end if
    end while
    assign $y = R(\tau)$ to the executed transitions in τ
    add τ to $\mathcal{B}_{\text{exec}}$ and stage its executed state–action outcomes in $\mathcal{P}_k$
  end for
  3. Policy learning and memory update
  compute A^{GiGPO} from $\mathcal{B}_{\text{exec}}$ and apply the group-mean weight in Eq. equation 4
  construct $\mathcal{L}_{\text{aux}}$ using Eq. equation 5 from records in $\mathcal{U}_k$ whose associated executed trajectory succeeds ($y = 1$)
  update θ_k with $\mathcal{L}_{\text{total}}$ in Eq. equation 7 to obtain θ_{k+1}
  if the optimizer step succeeds **then**
    $\mathcal{M}_{k+1} \leftarrow \text{DECAYANDCOMMIT}(\mathcal{M}_k, \mathcal{P}_k)$ ▷ executed outcomes only
  else
    $\mathcal{M}_{k+1} \leftarrow \mathcal{M}_k$
  end if
end for
return π_{θ_T}

Standard and EGAS-enabled trajectories occupy separate rollout subsets in our implementation. Within an EGAS-enabled trajectory, additional candidate sampling may recur until B_{int} selected

actions differ from their initially sampled actions; retaining the initially sampled action leaves the budget unchanged. The experience pool remains fixed during rollout collection and loss construction. Staged outcomes from executed state–action pairs are committed only after a successful optimizer step; unexecuted candidates never update the pool. For the supplementary GRPO+EGAS setting, only the base RL objective is replaced; action selection, auxiliary learning, and experience-pool updates remain unchanged.

B PROPERTIES AND DESIGN ANALYSIS

We analyze potential-improvement weighting, the auxiliary preference signal, coefficient scaling, and time decay.

B.1 PROPERTIES OF POTENTIAL-IMPROVEMENT WEIGHTING

Proposition 1 (Range and sign preservation)**.** *For any experience-pool snapshot and supported action set,* $0 \le \kappa_t \le 1$. *Consequently,* $0 \le \bar{\kappa}_G \le 1$ *for every GiGPO state group* G*, including the convention* $\bar{\kappa}_G = 0$ *when no supported visit is available. The group weight therefore satisfies* $1 \le 1 + \bar{\kappa}_G \le 2$*, and* $\operatorname{sign}(\widetilde{A}_t) = \operatorname{sign}(A_t^{\mathrm{GiGPO}})$ *whenever the base advantage is nonzero.*

Proof. If the initially sampled action lacks sufficient support or no distinct supported alternative is available, $\kappa_t = 0$ by definition. Otherwise, $\mathcal{A}_t^{\sup}$ contains $a_t^{(0)}$, and every Beta draw lies in $[0, 1]$. Hence $\max_{a \in \mathcal{A}_t^{\sup}} q_{t,a} - q_{t,a_t^{(0)}}$ is nonnegative and at most one. Taking the positive part and expectation therefore gives $0 \le \kappa_t \le 1$. Any average $\bar{\kappa}_G$ of supported scores lies in the same interval, and the empty-set convention also gives zero. Thus $1 \le 1 + \bar{\kappa}_G \le 2$. Because this group weight is strictly positive, multiplication preserves the sign of every nonzero base advantage. □

The shared multiplier preserves advantage signs and increases their magnitudes by at most a factor of two.

Consider a GiGPO state group G with n_G visits and base advantages $\{A_t^{\mathrm{GiGPO}} : t \in G\}$. Whenever the base within-group normalization is centered,

$$\frac{1}{n_G} \sum_{t \in G} A_t^{\mathrm{GiGPO}} = 0. \tag{8}$$

The group-mean rule in equation 4 applies the same factor $w_G = 1 + \bar{\kappa}_G$ to every visit. It therefore retains this property:

$$\frac{1}{n_G} \sum_{t \in G} \widetilde{A}_t = w_G \frac{1}{n_G} \sum_{t \in G} A_t^{\mathrm{GiGPO}} = 0. \tag{9}$$

It also preserves the ordering and ratios of nonzero advantages within the group. By contrast, per-visit weighting gives $\widetilde{A}_t^{\mathrm{visit}} = (1 + \kappa_t) A_t^{\mathrm{GiGPO}}$ and hence

$$\begin{aligned} \frac{1}{n_G} \sum_{t \in G} \widetilde{A}_t^{\mathrm{visit}} &= \frac{1}{n_G} \sum_{t \in G} \kappa_t A_t^{\mathrm{GiGPO}} \\ &= \operatorname{Cov}_G\left(\kappa_t, A_t^{\mathrm{GiGPO}}\right), \end{aligned} \tag{10}$$

where the second equality uses equation 8 and the empirical covariance with denominator n_G. This term is generally nonzero when the potential-improvement estimate varies within the group and correlates with its realized advantages. Per-visit weighting can therefore alter the comparison constructed by GiGPO's group normalization, rather than only changing the strength assigned to that group.

Group-mean weighting preserves within-group advantage ratios and, when the base advantages are centered, their zero mean. It intentionally changes the relative emphasis across groups.

Empirical sanity check. Figure 8 compares group-mean and per-visit potential-improvement weighting under the same ALFWorld Qwen3-4B training setup, with GiGPO as the unweighted reference. Group-mean weighting reaches 71.1% best-observed and final success. Per-visit weighting reaches 57.8% and ends at 56.3%, compared with 68.0% and 64.8% for GiGPO. This comparison supports using a shared group weight, which preserves the signs and relative proportions of within-group advantages rather than rescaling each visit independently.

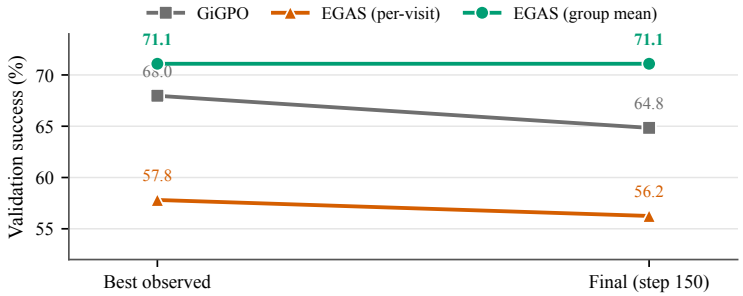

Figure 8: Group-mean versus per-visit potential-improvement weighting on ALFWorld with Qwen3-4B. The plot reports both best-observed and final validation success; GiGPO is the unweighted reference. Group-mean weighting achieves the highest final success in this comparison.

B.2 Posterior Interpretation of the Auxiliary Preference Signal

For a fixed decision, suppress the time index and consider a successful executed trajectory with $y = R(\tau) = 1$. Define a latent pairwise label $Y_{iJ} = +1$ if $q_i > q_J^+$ and $Y_{iJ} = -1$ otherwise. Equation equation 5 then satisfies

$$A_{iJ}^{\text{aux}} = \mathbb{E}[Y_{iJ} \mid \mathcal{M}_k, y = 1]. \tag{11}$$

The signal is positive when the experience-pool posterior favors the unexecuted candidate and negative when it favors the selected action; values near zero provide little directional preference. It is therefore a posterior mean of a latent pairwise comparison, not an observed counterfactual return for the unexecuted action.

B.3 Auxiliary Weight Scaling

We set the auxiliary coefficient as

$$\lambda_{\text{aux}} = \frac{\lambda_0}{K}, \tag{12}$$

where λ_0 controls the base strength of the auxiliary objective and K is the candidate-set size. The two losses are normalized separately. The $1/K$ scaling further reduces the auxiliary coefficient as the candidate count increases, limiting the weight assigned to experience-derived preferences. At fixed λ_0, varying K changes both the candidate count and the auxiliary coefficient.

B.4 What Time Decay Controls

Under the update rule in Eq. equation 15, one unit of success or failure evidence contributes pseudo-count mass $\rho^d = 2^{-d/H}$ after d successful updates. Its cumulative mass over future experience-pool snapshots is the geometric sum $\sum_{d=0}^{\infty} \rho^d = (1-\rho)^{-1}$. Thus, H is the evidence half-life and determines how quickly historical evidence decays.

C Implementation Details

C.1 Training Details

Table 3 reports the environment-specific settings for EGAS and the controlled GiGPO runs. Each task group contains eight trajectories for the same task: four standard and four EGAS-enabled trajectories for EGAS, or eight standard trajectories for GiGPO. Each setting is trained for 150 optimizer updates.

Table 3: Training and rollout configuration. Micro-batch sizes are reported per GPU. "Mean–std" and "mean" denote the GiGPO within-group advantage normalization modes.

Setting	ALFWorld	WebShop	Sokoban
Backbone	Qwen3-1.7B-Instruct Qwen3-4B-Instruct-2507	Qwen3-1.7B-Instruct Qwen3-4B-Instruct-2507	Qwen3-VL-4B-Instruct
Task groups / update	16	16	32
Trajectories / group	8	8	8
Training updates	150	150	150
Max. environment steps	50	15	15
Observation history	2	2	2
Max. prompt length	2048	4096	1024
Max. response length	512	512	512
PPO mini-batch	256	64	64
Actor micro-batch	32	8	8
Log-prob. micro-batch	32	16	16
GiGPO normalization	Mean–std	Mean	Mean

Across environments, we use AdamW with an actor learning rate of 10^{-6}, weight decay 0.01, and gradient clipping at 1.0. The token-level PPO objective uses a clip ratio of 0.2, one optimization epoch per update, discount $\gamma = 0.95$, and KL coefficient 0.01 (Schulman et al., 2017). Training actions are sampled at temperature 1.0, and an invalid-action penalty of 0.1 is applied consistently across controlled runs. Training uses two NVIDIA B300 GPUs with tensor parallelism of two. All Qwen3 models are run with thinking mode disabled. Validation is performed before training and every five updates.

For EGAS, we set $K = 8$, use $H = 50$ in the decay rule of Eq. equation 15, and set $m = 3$ and $B_{\text{int}} = 2$. Candidate actions are selected by their posterior mean under the time-decayed experience pool. We use 64 Monte Carlo samples to estimate κ_t, set $\lambda_0 = 0.05$, and scale the auxiliary coefficient as $\lambda_{\text{aux}} = \lambda_0/K$.

Clipped policy losses. We apply PPO clipping and dual clipping to both policy losses in Eq. equation 6 (Schulman et al., 2017; Ye et al., 2020). For a token-level probability ratio r and a fixed advantage A, define

$$\begin{aligned} \ell_{\text{PPO}}(r, A) &= \max\{-rA, -\operatorname{clip}(r, 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}})A\}\,, \\ \ell_{\text{dual}}(r, A) &= \begin{cases} \min\{\ell_{\text{PPO}}(r, A), -cA\}\,, & A < 0, \\ \ell_{\text{PPO}}(r, A), & A \geq 0. \end{cases} \end{aligned} \tag{13}$$

These are loss-minimization forms. We use $\epsilon_{\text{low}} = \epsilon_{\text{high}} = 0.2$ and $c = 3$. Here, clip truncates the ratio to $[0.8, 1.2]$ in the clipped term; the additional dual clip caps the token loss at $-cA$ when $A < 0$. Using Eq. equation 13, the implemented policy losses are

$$\begin{aligned} \mathcal{L}_{\text{base}}(\theta) &= \mathbb{E}_{\text{base}}\Big[\ell_{\text{dual}}\Big(r_{t,\ell}(\theta), \widetilde{A}_t\Big)\Big]\,, \\ \mathcal{L}_{\text{aux}}(\theta) &= \mathbb{E}_{\text{aux}}\big[\ell_{\text{dual}}\big(r_{t,i,\ell}(\theta), A^{\text{aux}}_{t,iJ}\big)\big]\,. \end{aligned} \tag{14}$$

The expectations in Eq. equation 14 use the same token masks as the unclipped losses: valid response tokens for the base loss and executable-action tokens of eligible unexecuted candidates for the auxiliary loss. Each loss is normalized by its own valid-token count over the PPO mini-batch; the auxiliary loss is zero when no eligible tokens are present. We combine the two losses using $\lambda_{\text{aux}} = \lambda_0/K$ as in Eq. equation 7.

C.2 Evaluation Protocol

At each checkpoint, we freeze the policy, disable EGAS action selection, and sample one response per decision at temperature 0.4. Evaluation uses 128 episodes for ALFWorld and Sokoban and 256 for WebShop, processed as two batches of 128. Episode horizons follow Table 3. ALFWorld and Sokoban report binary task success, whereas WebShop reports both binary purchase success and normalized task score. The ALFWorld average weights episodes equally, with task-type columns computed on their corresponding subsets.

C.3 Dataset Splits

Table 4 summarizes how each environment separates training from evaluation. The data files initialize environment instances rather than fixing task identities; tasks are sampled when the environment is reset. We therefore enforce separation through the environment-level splits shown below. For WebShop, N denotes the total number of available shopping goals.

Table 4: Training and evaluation data usage. Counts in the last column are evaluation episodes per checkpoint.

Environment	Training source	Evaluation source	Episodes
ALFWorld	Official `train` games	Official in-distribution `valid_seen` games	128
WebShop	Goal indices $500, \ldots, N-1$ in the 1,000-product catalog	Held-out goal indices $0, \ldots, 499$ from the same catalog	256
Sokoban	Procedurally generated 6×6 one-box rooms with reset seeds in $[0, 2^{16}-1)$	Procedurally generated rooms with disjoint reset seeds in $[2^{16}, 2^{32}-1)$	128

C.4 Experience-Pool Key Construction and Normalization

Each experience-pool entry is indexed by a task, state, and action tuple $(u, z_t, \bar{a}_t)$. Here, $\bar{a}_t$ is the normalized key corresponding to the executable action a used in Section 3. Its construction follows three principles.

Task isolation. The task key u prevents outcomes from different tasks from being pooled. Thus, identical observations and actions in two tasks still correspond to different experience-pool entries.

Prompt-independent state identity. The state key z_t is constructed from the environment-provided observation anchor rather than the complete model prompt. Prompt templates and appended action history therefore do not create different keys for the same observable state.

Executable-action identity. The action key $\bar{a}_t$ is constructed from the action accepted by the environment projection, rather than the complete model response. Responses that execute the same action therefore share their outcome evidence.

After these projections, text keys are lowercased, stripped, and reduced to single spaces; structured observations are first serialized deterministically. These operations define $z_t = \phi(o_t^{\text{anchor}})$ and $\bar{a}_t = \psi(a_t)$.

For ALFWorld, u is the game-file identifier, z_t is the current textual observation, and $\bar{a}_t$ is an admissible textual command. WebShop uses the goal index, current formatted page observation, and a projected `search[...]` or `click[...]` action. Sokoban uses the reset seed, the current board observation, and the projected movement action.

For example, an ALFWorld observation ``` ``You are in the kitchen. You see a Fridge 1.'' ``` is normalized to ``` ``you are in the kitchen. you see a fridge 1.'' ```. A response `<action> OPEN Fridge 1 </action>` is first projected by the environment and then stored as the action key ``` ``open fridge 1'' ```. Responses that project to this same executable command therefore share one action key. This procedure removes formatting variation but does not perform semantic matching; differently worded equivalent observations may still form different keys.

Time-decayed evidence update. Let $e = (u, z_t, \bar{a}_t)$ denote an experience-pool entry, and let $\Delta S_k(e)$ and $\Delta F_k(e)$ denote the success and failure evidence committed after training update k. On each successful optimizer update, we apply

$$\begin{aligned} \widetilde{S}_{k+1}(e) &= \rho\, \widetilde{S}_k(e) + \Delta S_k(e), \\ \widetilde{F}_{k+1}(e) &= \rho\, \widetilde{F}_k(e) + \Delta F_k(e), \end{aligned} \qquad \rho = 2^{-1/H}. \tag{15}$$

Thus, evidence that is d successful updates old has weight $\rho^d = 2^{-d/H}$. We implement this update lazily by storing each entry's most recent update index.

The auxiliary loss is applied only to tokens corresponding to the executable action; a candidate is used only when this span can be mapped reliably.

D Environment Prompts

EGAS and GiGPO use the same environment templates, controlling for prompt design so that the comparison isolates differences in data collection and learning. Placeholders in braces are filled from the current environment state. For ALFWorld and WebShop, the first decision omits the history block; later decisions include the two most recent observation–action pairs, as specified by Table 3. The visual Sokoban prompt contains the current image without textual action history. Although the prompts request brief task reasoning in the response, all models are run with their built-in thinking mode disabled.

D.1 ALFWorld

ALFWorld provides the current text observation and admissible-action set. The task appears in the reset observation and is inserted explicitly as {`task_description`} at subsequent decisions.

ALFWorld Interaction Prompt

You are an expert agent operating in the ALFRED Embodied Environment. Your task is to:
{task_description}
Prior to this step, you have already taken {step_count} step(s). Below are the most recent
{history_length} observations and the corresponding actions you took: {action_history}
You are now at step {current_step} and your current observation is: {current_observation}
Your admissible actions of the current situation are: [{admissible_actions}].

Now it's your turn to take an action.
You should first reason step−by−step about the current situation.
Once you've finished your reasoning, you should choose an admissible action for current step and
present it within <action> </action> tags.

D.2 WebShop

WebShop exposes the shopping instruction, current page observation, and legal search or click actions. Both model scales use the same prompt body:

WebShop Prompt Body

You are an expert autonomous agent operating in the WebShop e−commerce environment.
Your task is to: {task_description}.
Prior to this step, you have already taken {step_count} step(s). Below are the most recent
{history_length} observations and the corresponding actions you took: {action_history}
You are now at step {current_step} and your current observation is: {current_observation}.
Your admissible actions of the current situation are:
[
{available_actions}
].

WebShop Output Instruction — Qwen3-1.7B-Instruct

Now it's your turn to take one action for the current step.
You should first reason step–by–step about the current situation, then think carefully which admissible action best advances the shopping goal.
Once you've finished your reasoning, you should choose an admissible action for current step and present it within <action> </action> tags.

WebShop Output Instruction — Qwen3-4B-Instruct-2507

Now it's your turn to take one action for the current step.
Reason briefly step–by–step about the current situation and which admissible action best advances the shopping goal. Explain the key evidence for your choice in a short, focused paragraph without copying the task, observation, or action list verbatim.
After your reasoning, output exactly one admissible action within <action> </action> tags.

D.3 SOKOBAN

The Sokoban experiments use Qwen3-VL-4B-Instruct and therefore provide the board as an image rather than a text grid. The prompt defines the visual entities, irreversible-push constraint, and four-action interface:

Sokoban Visual Interaction Prompt

You are an expert agent operating in the Sokoban environment. Your goal is to push all the boxes onto the target spots. Once all boxes are on the targets, you win!

Rules
You can only push boxes. You can't pull them, so plan ahead to avoid getting stuck.
You can't walk through or push boxes into walls.
To avoid traps, do not push boxes into corners or against walls where they can't be moved again.

Visual Elements in the Image
Character: A small, green alien–like figure with two antennae and black eyes. It represents you.
Box: A yellow crate marked with an orange "X" across its front. It is the box you need to push.
Target: A black tile outlined in red, with a small red diamond shape in the center. It marks the destination where a box should be pushed.

Current Step
Your current observation is shown in the image: <image>
Your admissible actions are ["up", "down", "left", "right"].

Now it's your turn to take one action for the current step.
Briefly consider the current board and choose the admissible action that best advances the boxes toward the targets while avoiding deadlocks. Keep your reasoning concise and focused, and do not restate the board, rules, or action list.
Then immediately output exactly one admissible action within <action> </action> tags.

E ILLUSTRATIVE TASK AND ROLLOUT TRACE

We use a constructed ALFWorld-style walkthrough to illustrate experience-guided action selection and auxiliary learning. For clarity, the example uses $K = 3$, whereas the experiments use $K = 8$. All candidate actions are freshly sampled from the current policy, and only the selected action is executed in the environment.

Worked Example: Clean an Apple and Put It on the Dining Table

[1. Task]
Put a clean apple on diningtable 1.

[2. Executed trajectory prefix]

Action: <action>go to countertop 1</action>
Observation: On countertop 1, you see apple 1.

Action: <action>take apple 1 from countertop 1</action>
Observation: You pick up apple 1 from countertop 1.

Action: <action>go to sinkbasin 1</action>
Observation: You arrive at sinkbasin 1.

[3. A state with potential improvement]
Current state: At the sink, holding apple 1; the apple is not yet clean.
Initially sampled action: <action>go to diningtable 1</action>

Experience−pool evidence: Earlier executed trajectories provide sufficient support for the initially sampled action and for cleaning the apple at this state. The resulting potential improvement is positive, so EGAS retains the initially sampled action and samples two additional candidate actions.

[4. Candidate actions from the same state −− candidate−set size K = 3]
Candidate 0 | Initially sampled action
<action>go to diningtable 1</action>
Move toward the destination before cleaning the apple.

Candidate 1 | Additional candidate action −− SELECTED
<action>clean apple 1 with sinkbasin 1</action>
Satisfy the cleaning requirement before moving to the destination.

Candidate 2 | Additional candidate action
<action>go to fridge 1</action>
Move to another location while still holding the apple.

Selection: The experience pool assigns Candidate 1 the highest posterior mean success estimate, so EGAS selects it for execution. Candidates 0 and 2 remain unexecuted.

[5. Continue the single environment trajectory]
Action: <action>clean apple 1 with sinkbasin 1</action>
Observation: You clean apple 1 using sinkbasin 1.

Action: <action>go to diningtable 1</action>
Observation: You arrive at diningtable 1.

Action: <action>put apple 1 in/on diningtable 1</action>
Observation: You put the clean apple on diningtable 1. Task completed.

[6. Outcome accounting]
In this illustrative continuation, the executed trajectory succeeds. Only executed state−action pairs receive that terminal outcome and enter the experience−pool update. The two unexecuted candidates receive no return and never update the experience pool. Because the executed trajectory succeeds, they may nevertheless contribute to the auxiliary pairwise preference objective constructed with the experience pool.

The executed trajectory provides the observed outcome for weighted RL updates and experience-pool updates. The unexecuted candidates provide additional learning signals through experience-based pairwise preferences, without separate downstream trajectories.

F Current-Policy Candidates versus Retrieved Historical Actions

EGAS uses the experience pool to evaluate candidate actions sampled from the current policy rather than retrieving a stored action for execution. We examine this choice on ALFWorld with Qwen3-4B-

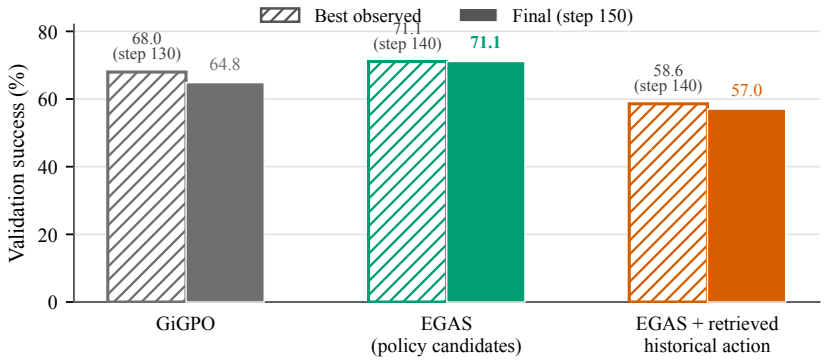

Figure 9: Retrieved-action ablation on ALFWorld with Qwen3-4B-Instruct-2507. Hatched bars show best-observed validation success, with the corresponding training step in parentheses; solid bars show final success at step 150. GiGPO is the controlled baseline without EGAS. Standard EGAS restricts selection to candidates sampled from the current policy. Appending a retrieved historical action reduces both best-observed and final success in this controlled run.

Instruct-2507. Both EGAS variants use group-mean potential-improvement weighting, $K = 8$, four standard trajectories, and four EGAS-enabled trajectories. Standard EGAS selects only among the K current-policy candidates. The retrieval variant appends at most one supported admissible historical action with the highest posterior mean, yielding up to $K + 1$ candidates before deduplication. GiGPO is included as the controlled baseline without EGAS.

Despite allowing an additional retrieved candidate, the retrieval variant performs worse. Standard EGAS reaches 71.1% best-observed and final success, whereas the retrieval variant reaches 58.6% and 57.0%, respectively. Its final success is 14.1 percentage points below standard EGAS and 7.8 points below GiGPO.

This controlled run supports using the experience pool as an evaluator of current-policy candidates rather than as an action-retrieval policy. One possible explanation is that retrieving a historical action bypasses the current policy distribution. Even when the action remains admissible and has a high posterior mean in the experience pool, it may be less compatible with the current policy or with distinctions omitted by the normalized state key.

G MULTI-SEED RESULTS

Tables 5–7 report GiGPO+EGAS results across three random seeds for ALFWorld, WebShop, and Sokoban. All summary statistics are computed over seeds 0–2; SD denotes the sample standard deviation.

Table 5: GiGPO+EGAS held-out success (%) on ALFWorld across random seeds.

Model	Seed/Stat.	ALFWorld success (%)						
		Pick	Look	Clean	Heat	Cool	Pick2	Avg.
Qwen3-1.7B	0	80.0	45.5	73.9	47.4	53.3	30.0	60.2
	1	72.4	61.5	68.0	68.8	43.8	65.5	64.8
	2	72.7	27.3	87.5	65.0	56.5	36.0	59.4
	Mean	75.0	44.8	76.5	60.4	51.2	43.8	61.5
	SD	4.3	17.1	10.0	11.4	6.6	19.0	2.9
Qwen3-4B	0	100.0	53.9	56.5	58.3	61.3	76.2	71.1
	1	89.3	53.9	52.2	66.7	74.2	76.2	70.3
	2	89.3	61.5	59.4	84.6	63.6	85.0	73.4
	Mean	92.9	56.4	56.0	69.9	66.4	79.1	71.6
	SD	6.2	4.4	3.6	13.4	6.9	5.1	1.6

Table 6: GiGPO+EGAS normalized score and success (%) on WebShop across random seeds.

Model	Seed/Stat.	Score (%)	Success (%)
Qwen3-1.7B	0	80.2	64.5
	1	80.9	65.2
	2	77.1	61.8
	Mean	79.4	63.8
	SD	2.0	1.8
Qwen3-4B	0	89.6	80.5
	1	88.4	79.3
	2	88.2	79.7
	Mean	88.7	79.8
	SD	0.8	0.6

Table 7: GiGPO+EGAS success (%) on Sokoban across random seeds with Qwen3-VL-4B.

Seed/Stat.	Success (%)
0	80.5
1	85.9
2	89.8
Mean	85.4
SD	4.7