COMPETERL: FORESIGHT–HINDSIGHT COMPETITIVE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Experiential learning is becoming tightly integrated with agentic reinforcement learning (RL), which uses task outcomes to directly optimize experience utilization and extraction. However, existing methods separate experience extraction from utilization across rollouts, delaying feedback of newly extracted experience and leaving retrieved historical experience poorly adapted to the current task. To tackle the challenge, we introduce **Experience-in-the-Loop Learning**, a formulation that turns experience extraction and utilization into trainable decisions within the policy rollout. Building on this formulation, we propose COMPETERL , an online RL framework that integrates experience generation, utilization, and validation within a unified rollout, while improving experience quality through foresight–hindsight competition. Specifically, COMPETERL adapts retrieved knowledge into foresight experience before acting and distills trajectory-grounded hindsight experience afterward. The two experiences compete through their guided execution returns, forming a closed loop of task-specific adaptation and immediate feedback. Extensive experiments across four agent benchmarks using Qwen2.5-7B and Qwen3-4B show that COMPETERL **(I) delivers strong task performance**, exceeding Skill1 by 6.1% on average with Qwen3-4B, and **(II) lies on the Pareto-optimal frontier** of cost and performance across four datasets, outperforming SkillRL by 7.4% on ALFWorld at comparable token cost.

1 INTRODUCTION

Large language model (LLM) agents have demonstrated remarkable progress in complex reasoning, coding, and long-horizon tasks (Yang et al., 2024; Guo et al., 2025; Wang et al., 2026b). Pivotal to this success is the concept of experience memory, which allows LLM agents to acquire skills or experience memory from past trajectories and achieve recursive self-improvement (Zhang et al., 2025a; Yang et al., 2025b; Ouyang et al., 2026). Existing experience memory systems typically retrieve relevant experiences to guide agent execution and update the memory bank by extracting knowledge from completed trajectories (Zhao et al., 2024; Yang et al., 2025c; Ferraz et al., 2026). However, these approaches require agents to manually identify transferable patterns from long-horizon trajectories and accurately reuse them in new contexts, placing increasing demands on the underlying model capabilities (Zhang et al., 2026a; 2025b; Ma et al., 2026). Motivated by this challenge, reinforcement learning (RL) has recently been explored to jointly optimize experience utilization and extraction policies (Yan et al., 2026; Xia et al., 2026; Shi et al., 2026).

Despite this progress, existing RL methods (Muhtar et al., 2026; He et al., 2026) still face two limitations: post-hoc experience generation lacks reliable and immediate feedback on its utility, while historical experience use lacks an explicit mechanism for generalizing to the current task. On the generation side, experience distilled after a source rollout (Shinn et al., 2023; Yang et al., 2026; Wu et al., 2026) does not contribute to that rollout's outcome, so the source return cannot directly evaluate its guidance value. Evaluating it indirectly through future tasks (Zhang et al., 2026b; Yao et al., 2026; Shen et al., 2026) instead delays utility feedback and entangles experience quality with target-task difficulty and experience–task compatibility. On the utilization side, retrieved experience enters the target context as a static entry whose source-specific assumptions and strategies may not fit the current task (Zhao et al., 2026; Liang et al., 2026). Together, these limitations reflect the cross-rollout misalignment illustrated in Figure 1: experience is generated after a source rollout but

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.

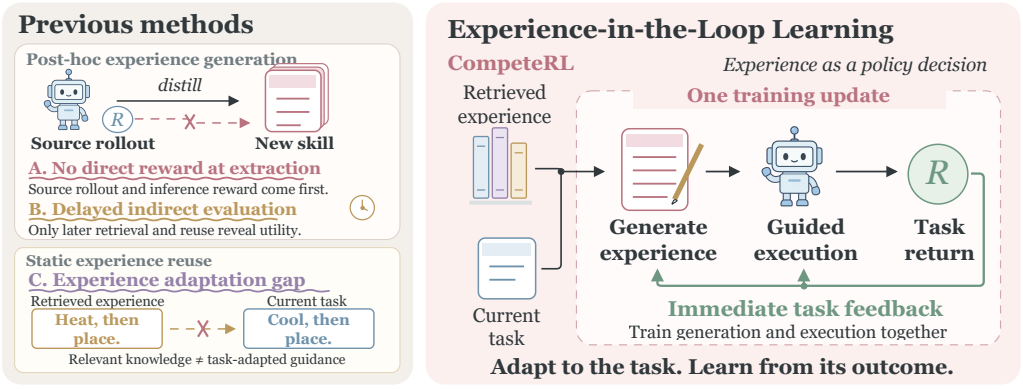

Figure 1: Previous methods (left) place skill generation and its subsequent use and validation in separate rollouts, while reusing static skills. Experience-in-the-Loop Learning (right) jointly optimizes task-conditioned experience and the execution it guides within one policy update.

is only retrieved, used, and evaluated as static knowledge in a later rollout (Liu et al., 2026; Wang et al., 2024b; Ju et al., 2026).

To resolve this misalignment, experience generation and experience utilization should be incorporated into a unified rollout and optimized as trainable policy decisions (Shi et al., 2026). Specifically, pre-execution experience adaptation transforms retrieved knowledge into task-specific **foresight experience**, which the actor then uses to solve the current task. Because the generated experience causally precedes the execution it guides, the resulting task outcome directly optimizes both foresight experience generation and experience utilization, establishing a direct end-to-end credit path. We call this formulation **Experience-in-the-Loop Learning**: an online end-to-end training paradigm that jointly optimizes experience generation and utilization.

Building on this formulation, we propose COMPETERL , which couples foresight experience with a trajectory-informed hindsight counterpart through competitive online RL. COMPETERL first generates task-specific foresight experience from retrieved historical knowledge and uses it to guide actor execution. To complement foresight experience with trajectory information unavailable before execution, the policy further distills the completed trajectory into **hindsight experience**. The hindsight experience guides a task-matched re-execution, providing immediate task feedback for post-execution experience generation within the same update. The two task-matched returns serve as moving baselines for experience generation, reducing interference from task difficulty and forming a dynamic competition. Under this competition, foresight experience learns to anticipate potential problems and narrow its performance gap with hindsight experience, while hindsight experience uses newly observed evidence to explore improvements beyond the current guidance. The resulting experiences are stored back into memory, forming a closed-loop cycle of experience foresight adaptation, utilization, and hindsight extraction. Our contributions are summarized as follows:

- **Experience-in-the-Loop Formulation.** We introduce a formulation that redefines experience adaptation, utilization, and extraction as learnable decisions within the policy rollout loop, enabling end-to-end optimization of the experience lifecycle through task feedback.
- **Foresight-Hindsight Co-Improvement.** We propose COMPETERL , an online RL algorithm that integrates task-conditioned foresight experience adaptation, trajectory-grounded hindsight experience extraction, and competitive optimization between the two, enabling the joint evolution of experience learning and agent execution.
- **Comprehensive Empirical Evaluation.** We evaluate COMPETERL across four agent tasks and two backbone models, showing consistent improvements in task performance and inference efficiency. Specifically, COMPETERL improves over Skill1 by 6.1% on average with Qwen3-4B and achieves Pareto-optimal cost-performance trade-offs across four datasets, outperforming SkillRL by 7.4% on ALFWorld at comparable token costs.

2 Related Work

Skill-Based Agent Self-Evolution. Experience memory, often instantiated as reusable skills, enables agents to improve beyond individual episodes by transforming knowledge accumulated from past interactions into actionable guidance for future tasks (Wang et al., 2024b; Ouyang et al., 2026; Ju et al., 2026). Early systems such as Reflexion (Shinn et al., 2023) and ExpeL (Zhao et al., 2024) summarize successful or failed trajectories into textual feedback that can be retrieved in subsequent episodes. Later approaches represent experience as executable programs, workflows, or structured skills, enabling agents to reuse procedural knowledge across tasks, as demonstrated by Voyager (Wang et al., 2024a) and AWM (Wang et al., 2024b). Recent studies further explore structured skill representations, task-aware composition, lifecycle management, and reflection mechanisms for maintaining an evolving collection of reusable behaviors (Si et al., 2026; Liang et al., 2026; Zhao et al., 2026; Ju et al., 2026; Shen et al., 2026). Despite these advances, existing systems typically inject retrieved experience as static context, leaving source-specific knowledge insufficiently adapted to the current task (Wang et al., 2024b; Ouyang et al., 2026). Our work builds on this experience-centric view but further integrates experience adaptation and construction into the agent's rollout, allowing retrieved knowledge to be transformed into task-specific guidance during execution.

Reinforcement Learning for Skill Evolution. Recent work has introduced reinforcement learning to optimize how agents use and construct reusable skills (Xia et al., 2026; Shi et al., 2026; Muhtar et al., 2026). SkillRL (Xia et al., 2026) and Skill1 (Shi et al., 2026) train the policy to improve skill selection and utilization, whereas SLIM (Shen et al., 2026) dynamically manages the active set of skills during training. Another line focuses on learning experience extraction or skill distillation from interaction trajectories, including OPID (Yang et al., 2026), SEED (Wu et al., 2026), Skill-SD (Wang et al., 2026a), and ReSkill (He et al., 2026). More recent approaches, such as ComplementaryRL (Muhtar et al., 2026), Skill1 (Shi et al., 2026), further study the co-evolution of the task-solving policy and the experience extractor. In many of these systems, however, experience construction and utility evaluation remain temporally separated: experience distilled after one rollout is evaluated through its use in subsequent trajectories (Wang et al., 2024b; Ouyang et al., 2026; Xia et al., 2026; Shi et al., 2026). In contrast, CompeteRL integrates skill adaptation, utilization, and extraction into a single rollout, enabling direct feedback from task outcomes to assess the utility of the experience and optimize the policy that produces it.

3 Method

3.1 Foresight Skill Adaptation

Training and skill acquisition proceed iteratively. At iteration t, CompeteRL is given an interactive task instance $x = (q, o_0)$, where $q \sim \mathcal{D}$ denotes the task and o_0 its initial observation, together with the current skill library $\mathcal{M}_t = \{m_{t,j}\}_{j=1}^{|\mathcal{M}_t|}$.

The lifecycle begins with a lightweight retrieval step in which CompeteRL uses embedding function $f(\cdot)$ and similarity function $s(\cdot,\cdot)$ to retrieve the top-K skill set $\mathcal{C}_t(x)$ from $\mathcal{M}_t$.

$$\mathcal{C}_t(x) = \underset{m \in \mathcal{M}_t}{\arg\,\mathrm{topK}}\, s\big(f(x), f(m)\big). \tag{1}$$

For each rollout $i \in \{1, \ldots, G\}$ in a group of G parallel rollouts for the same task, the policy π_θ is conditioned on the foresight instruction ω_{F} to adapt the retrieved skills $\mathcal{C}_t(x)$ to the current task, yielding a task-specific foresight skill e_i^{F}:

$$e_i^{\mathrm{F}} = (e_{i,1}^{\mathrm{F}}, \ldots, e_{i,L_i^{\mathrm{F}}}^{\mathrm{F}}) \sim \prod_{\ell=1}^{L_i^{\mathrm{F}}} \pi_\theta\big(e_{i,\ell}^{\mathrm{F}} \mid x, C(x), e_{i,<\ell}^{\mathrm{F}}, \omega_{\mathrm{F}}\big), \qquad L_i^{\mathrm{F}} = \big|e_i^{\mathrm{F}}\big|. \tag{2}$$

The resulting foresight skill e_i^{F} consolidates the potentially stale and contextually misaligned skills $\mathcal{C}_t(x)$ into guidance tailored to the current task. Guided by e_i^{F}, the policy π_θ, now conditioned on the acting instruction ω_{A}, executes the task through interaction with the environment $\mathcal{E}$:

$$\tau_i^{\mathrm{F}} = \Big(o_0, a_{i,1}^{\mathrm{F}}, o_{i,1}^{\mathrm{F}}, \ldots, a_{i,T_i^{\mathrm{F}}}^{\mathrm{F}}, o_{i,T_i^{\mathrm{F}}}^{\mathrm{F}}\Big) \sim P_{\mathcal{E},\theta}\big(\cdot \mid x, e_i^{\mathrm{F}}, \omega_{\mathrm{A}}\big), \qquad R_i^{\mathrm{F}} = \mathcal{R}\big(x, \tau_i^{\mathrm{F}}\big). \tag{3}$$

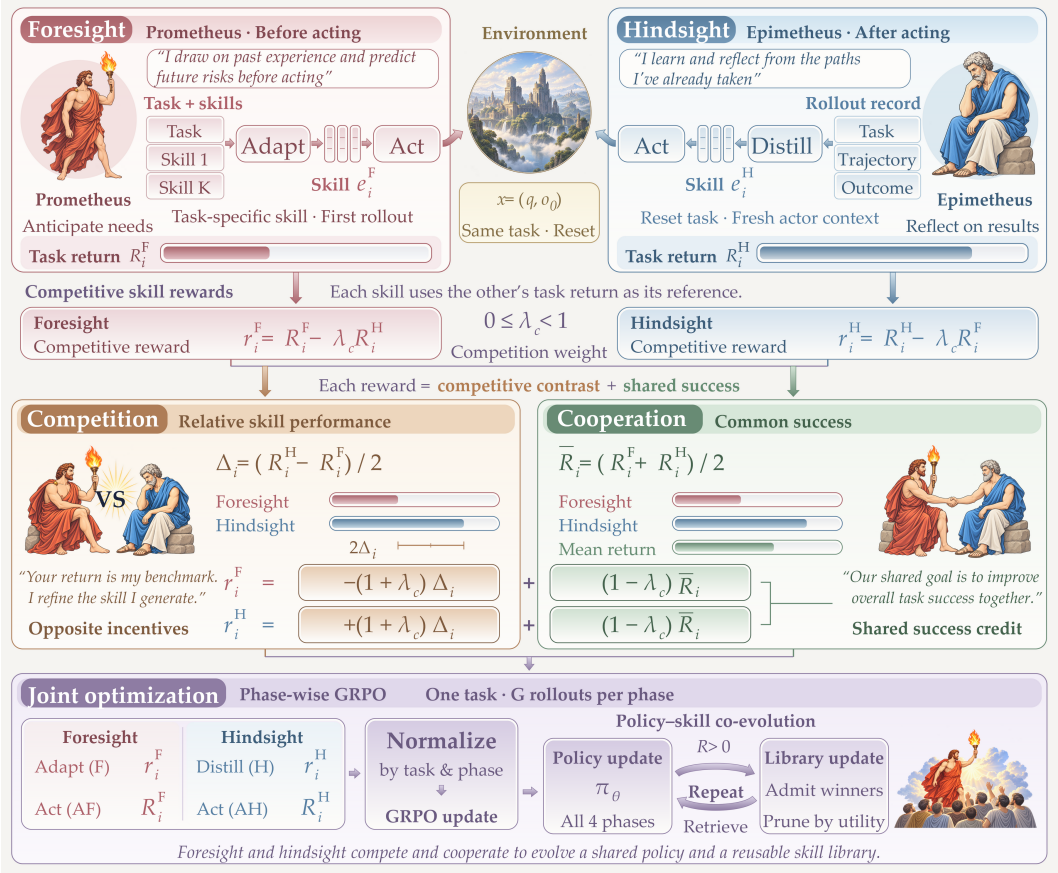

Figure 2: CompeteRL overview: the competition and cooperation between foresight and hindsight.

Here each $o_{i,s}^{\mathrm{F}}$ is the environment feedback elicited by action $a_{i,s}^{\mathrm{F}}$, and $P_{\mathcal{E},\theta}$ denotes the resulting closed-loop interaction between the actor and the environment. The return R_i^{F} therefore evaluates both the generated foresight skill and the behavior it induces, while their tokens remain identifiable as two separate phases for later credit assignment.

3.2 Hindsight Skill Distillation

The first execution not only evaluates the foresight skill but also reveals successes, failures, and environment feedback that were unavailable before acting. To iteratively refine its task-solving experience through interaction, CompeteRL collects this newly available evidence into a hindsight context $d_i^{\mathrm{F}} = \mathrm{Pack}\left(x, \tau_i^{\mathrm{F}}, R_i^{\mathrm{F}}, \chi_i\right)$, where χ_i denotes an optional reference trajectory from the same task with the opposite outcome. Conditioned on d_i^{F} and the hindsight instruction ω_{H}, the policy distills a trajectory-grounded hindsight skill e_i^{H}:

$$\pi_\theta\left(e_i^{\mathrm{H}} \mid d_i^{\mathrm{F}}, \omega_{\mathrm{H}}\right) = \prod_{\ell=1}^{L_i^{\mathrm{H}}} \pi_\theta\left(e_{i,\ell}^{\mathrm{H}} \mid d_i^{\mathrm{F}}, e_{i,<\ell}^{\mathrm{H}}, \omega_{\mathrm{H}}\right), \qquad L_i^{\mathrm{H}} = \left|e_i^{\mathrm{H}}\right|. \tag{4}$$

Because e_i^{H} is produced only after the first execution, R_i^{F} cannot evaluate the behavior that this new skill would induce. To obtain direct utility feedback for hindsight distillation, CompeteRL brings e_i^{H} back into the rollout as guidance for a second execution of the same task.

Before this execution, the environment is restored to its initial state $x_i^{\mathrm{H}} = \mathrm{Reset}\left(x_i^{\mathrm{F}}\right) = x$, and the actor context is rebuilt from x_i^{H} and e_i^{H} alone, exposing neither the first actor history nor the foresight skill e_i^{F}. The hindsight-guided execution then yields:

$$\tau_i^{\mathrm{H}} = \left(o_0, a_{i,1}^{\mathrm{H}}, o_{i,1}^{\mathrm{H}}, \ldots, a_{i,T_i^{\mathrm{H}}}^{\mathrm{H}}, o_{i,T_i^{\mathrm{H}}}^{\mathrm{H}}\right) \sim P_{\mathcal{E},\theta}\left(\cdot \mid x_i^{\mathrm{H}}, e_i^{\mathrm{H}}, \omega_{\mathrm{A}}\right), \qquad R_i^{\mathrm{H}} = \mathcal{R}\left(x, \tau_i^{\mathrm{H}}\right). \tag{5}$$

In this way, hindsight distillation refines the skill from prior interaction to iteratively improve execution and receives immediate feedback through R_i^{H} from this rollout, providing a trainable approach to recursive self-improvement.

3.3 Foresight–Hindsight Competitive Reinforcement Learning

The above foresight adaptation and hindsight distillation are both performed within the rollout, providing each with direct and immediate feedback. Considering that each outcome jointly reflects the quality of skill generation and actor execution, CompeteRL compares the two outcomes to reduce confounding from actor execution, thereby better attributing their performance difference to skill generation. Accordingly, CompeteRL combines the two returns into the competitive rewards:

$$r_i^{\mathrm{F}} = R_i^{\mathrm{F}} - \lambda_{\mathrm{c}} R_i^{\mathrm{H}}, \qquad r_i^{\mathrm{H}} = R_i^{\mathrm{H}} - \lambda_{\mathrm{c}} R_i^{\mathrm{F}}, \tag{6}$$

where r_i^{F} and r_i^{H} are assigned to foresight adaptation and hindsight distillation, respectively, and λ_{c} controls the strength of skill competition. To make their complementary roles explicit, we summarize the paired returns by the common-success signal $\bar{R}_i$, which measures their average task performance, and the skill-competition signal Δ_i, which captures their relative advantage on the matched task. They are defined as:

$$\bar{R}_i = \frac{R_i^{\mathrm{F}} + R_i^{\mathrm{H}}}{2}, \qquad \Delta_i = \frac{R_i^{\mathrm{H}} - R_i^{\mathrm{F}}}{2}. \tag{7}$$

The paired rewards then separate into a common-success and a skill-competition component

$$\begin{bmatrix} r_i^{\mathrm{F}} \\ r_i^{\mathrm{H}} \end{bmatrix} = \underbrace{(1-\lambda_{\mathrm{c}})\bar{R}_i \begin{bmatrix} 1 \\ 1 \end{bmatrix}}_{\text{common-success reward}} + \underbrace{(1+\lambda_{\mathrm{c}})\Delta_i \begin{bmatrix} -1 \\ 1 \end{bmatrix}}_{\text{skill-competition reward}}. \tag{8}$$

For $0 \le \lambda_{\mathrm{c}} < 1$, the two experiences reinforce each other rather than engage in a zero-sum game. The common-success component jointly steers both foresight and hindsight experience toward higher overall task success. Meanwhile, the skill-competition component encourages the two to improve through mutual competition, with foresight skills better anticipating critical task information and hindsight skills better reflecting on the completed trajectory.

Above competitive rewards r_i^{F} and r_i^{H} supervise foresight adaptation and hindsight distillation, respectively, while foresight-guided and hindsight-guided acting are directly rewarded by their execution returns, with $r_i^{\mathrm{AF}} = R_i^{\mathrm{F}}$ and $r_i^{\mathrm{AH}} = R_i^{\mathrm{H}}$. For each $p \in \{\mathrm{F}, \mathrm{AF}, \mathrm{H}, \mathrm{AH}\}$, CompeteRL compares the corresponding rewards among the G rollouts and the same phase to obtain advantages:

$$\mu_x^p = \frac{1}{G}\sum_{i=1}^{G} r_i^p, \qquad (\sigma_x^p)^2 = \frac{1}{G}\sum_{i=1}^{G} (r_i^p - \mu_x^p)^2, \qquad A_i^p = \frac{r_i^p - \mu_x^p}{\sigma_x^p + \varepsilon_{\mathrm{num}}}. \tag{9}$$

After the responses from all four phases are collected, let $y_{i,k}$ denote generated token k, $y_{i,<k}$ its autoregressive prefix within the current response, and $\mathcal{H}_{i,k}$ the task, phase instruction, applicable skill, and environment history available before that response. The token-level importance ratio is

$$\rho_{i,k}(\theta) = \frac{\pi_\theta\left(y_{i,k} \mid \mathcal{H}_{i,k}, y_{i,<k}\right)}{\pi_{\theta_{\mathrm{old}}}\left(y_{i,k} \mid \mathcal{H}_{i,k}, y_{i,<k}\right)}. \tag{10}$$

The binary mask $m_{i,k}^p$ equals one for tokens generated in phase p and zero otherwise, so each token is optimized only with its corresponding phase advantage A_i^p. The resulting policy objective is

$$\mathcal{J}_{\mathrm{CompeteRL}}(\theta) = \frac{1}{Z}\sum_{p,i,k} m_{i,k}^p \min\left(\rho_{i,k}A_i^p, \mathrm{clip}\left(\rho_{i,k}, 1-\epsilon, 1+\epsilon\right) A_i^p\right) - \beta_{\mathrm{KL}}\widehat{\mathrm{KL}}, \tag{11}$$

where $Z = \sum_{p,i,k} m_{i,k}^p$. Through this objective, CompeteRL jointly optimizes the policy for foresight adaptation, hindsight distillation, and task execution.

3.4 Policy and Skill Library Co-Evolution

Our algorithm optimizes not only its internal policy but also an external persistent skill library. For each matched foresight–hindsight pair, CompeteRL selects the skill whose execution return is both higher and positive and admits it to the library

$$\mathcal{W}_i = \left\{ e_i^p \mid p \in \{\mathrm{F}, \mathrm{H}\},\ R_i^p = \max\left(R_i^{\mathrm{F}}, R_i^{\mathrm{H}}\right) > 0 \right\}, \qquad \mathcal{M}_{t+1} = \mathcal{M}_t \cup \bigcup_{i=1}^{G} \mathcal{W}_i. \tag{12}$$

Table 1: Qwen2.5-7B-Instruct results on ALFWorld, WebShop, MiniHack (MH), and ScienceWorld (SW). ALFWorld reports six task-family success rates and their micro-average; WebShop reports normalized score and success rate; MH and SW report success rate. The final Avg. is the macro-average of ALFWorld Avg., WebShop SR, MH SR, and SW SR. Best values are bold; dark- and light-green cells indicate the best and second-best results, respectively.

	ALFWorld OOD							WebShop		MH	SW	Avg.
Method	Pick	Look	Clean	Heat	Cool	Pick2	Avg.	Score	SR	SR	SR	Avg.
Training-free and Skill-free methods												
Vanilla (Direct)	25.0	16.7	3.2	17.4	0.0	11.8	11.9	2.3	2.3	4.7	8.6	6.9
ReAct	79.2	83.3	54.8	60.9	66.7	0.0	59.0	25.4	6.3	17.2	18.8	25.3
Reflexion	83.3	88.9	77.4	73.9	81.0	23.5	73.1	23.0	14.1	42.2	25.0	38.6
Skill-free RL methods												
PPO	92.3	64.0	92.5	89.5	80.3	68.8	80.4	81.4	68.7	59.3	24.0	58.1
GRPO	90.8	66.1	89.3	74.7	72.5	64.7	77.6	79.3	66.1	67.8	49.1	65.2
GiGPO	97.7	82.7	93.5	83.7	89.3	88.2	90.8	84.4	72.8	80.5	53.4	74.4
Training-free skill methods												
Mem0	75.0	88.9	61.3	47.8	57.1	0.0	56.7	18.5	3.9	14.1	14.1	22.2
ExpeL	62.5	83.3	51.6	21.7	38.1	11.8	45.5	27.4	10.9	15.6	10.9	20.7
SimpleMem	66.7	66.7	41.9	56.5	47.6	5.9	48.5	12.2	5.5	3.1	11.7	17.2
RL-based skill methods												
MemRL	62.8	38.5	22.2	12.5	8.0	0.0	21.4	29.5	9.2	25.8	16.4	18.2
SkillRL	95.8	72.2	90.3	91.3	95.2	88.2	89.6	85.2	72.7	77.3	39.1	69.7
Skill0	**100.0**	83.3	93.5	82.6	85.7	82.4	88.8	84.8	71.9	79.7	41.4	70.4
Complementary RL	91.7	77.8	87.1	87.0	76.2	88.2	85.1	87.0	80.9	88.2	47.6	75.5
SkillRise	**100.0**	88.9	90.3	87.0	85.7	82.4	89.6	85.8	82.4	85.2	50.8	77.0
Skill1	**100.0**	94.4	93.5	91.3	95.2	94.1	94.8	86.7	82.9	87.5	56.3	80.4
Our method												
COMPETERL	91.7	**100.0**	**96.8**	**95.2**	**100.0**	**100.0**	**97.0**	**89.2**	**84.4**	**91.4**	**60.9**	**83.4**

As training progresses, the growing skill library requires pruning redundant skills. Following the utility-based retirement principle of Skill1 (Shi et al., 2026), COMPETERL updates the utility of every retrieved skill as

$$U_{t+1}(e_i^p) \leftarrow (1-\alpha)U_t(e_i^p) + \alpha R_i^p, \qquad \forall e_i^p \in \mathcal{M}_t. \tag{13}$$

When the library exceeds its capacity, skills with the lowest utility are retired.

Competition determines which new skills enter, while recent utility determines which skills remain. The improved policy produces stronger skills, while the enhanced skill library, in turn, enables more effective policy execution, forming a closed loop of policy-library self-improvement.

4 EXPERIMENTS

4.1 EXPERIMENTAL PROTOCOL

Benchmarks and metrics. We evaluate COMPETERL on ALFWorld (Shridhar et al., 2021), WebShop (Yao et al., 2022), MiniHack (Samvelyan et al., 2021), and ScienceWorld (Wang et al., 2022), using Qwen2.5-7B-Instruct (Qwen et al., 2025) and Qwen3-4B (Yang et al., 2025a) as the main policy backbone. For ALFWorld, we evaluate a fixed set of 134 tasks from `valid_unseen` and report success rates for each task family and across all episodes. For WebShop, we report exact success rate and normalized task score on a scale from 0 to 100. For MiniHack, we evaluate `MiniHack-Room-Ultimate-5x5-v0` and report success rate. For ScienceWorld, success is defined as reaching the maximum task score of 100.

Baselines. Table 1 compares COMPETERL with fifteen baselines covering prompting, reinforcement learning without skills, skill-based prompting methods, and RL-based skill methods. The skill learning baselines include SkillRL (Xia et al., 2026), Skill0 (Lu et al., 2026), Complementary RL (Muhtar et al., 2026), SkillRise (Yao et al., 2026), and Skill1 (Shi et al., 2026). Results taken from prior work retain their original models and evaluation protocols and should not be interpreted as comparisons under identical settings.

Implementation details. Unless otherwise specified, we train COMPETERL with GRPO using a group size of $G = 8$, a batch size of 16 task prompts, a learning rate of 10^{-6}, and a KL coefficient of 0.01. We retrieve $K = 3$ candidate skills using the Qwen3 embedding encoder (Zhang et al., 2025c) and set the default competition coefficient to $\lambda_c = 0.5$. Each execution is limited to 50, 20,

Table 2: Qwen3-4B results under the same metrics and highlighting convention as Table 1.

	ALFWorld OOD							WebShop		MH	SW	Avg.
Method	Pick	Look	Clean	Heat	Cool	Pick2	Avg.	Score	SR	SR	SR	Avg.
Training-free and Skill-free methods												
Vanilla (Direct)	29.2	5.6	0.0	21.7	9.5	29.4	14.9	7.0	1.6	9.4	2.3	7.0
ReAct	33.3	5.6	12.9	26.1	9.5	23.5	18.7	28.6	17.2	7.8	3.9	11.9
Reflexion	54.2	55.6	22.6	26.1	33.3	52.9	38.8	30.3	20.3	21.9	4.7	21.4
Skill-free RL methods												
PPO	82.1	70.0	57.9	52.9	50.0	72.0	67.2	73.3	68.5	75.9	27.9	59.9
GRPO	94.1	75.0	95.2	66.7	75.0	52.6	79.7	77.4	71.6	77.2	31.2	64.9
GiGPO	90.3	88.2	**96.4**	83.3	75.0	60.0	83.6	78.6	73.7	79.7	35.1	68.0
Training-free skill methods												
Mem0	20.8	5.6	6.5	8.7	0.0	23.5	10.4	17.4	10.2	7.8	4.7	8.3
ExpeL	50.0	0.0	9.7	0.0	4.8	17.6	14.2	22.5	7.0	7.8	3.1	8.0
SimpleMem	33.3	11.1	29.0	4.3	9.5	35.3	20.9	16.8	7.0	10.9	3.1	10.5
RL-based skill methods												
MemRL	41.7	22.2	19.4	30.4	38.1	11.8	27.6	7.1	5.4	10.2	1.6	11.2
SkillRise	**100.0**	87.5	88.5	81.0	80.8	68.4	85.9	80.4	74.8	71.1	40.6	68.1
Skill1	91.7	83.3	83.9	82.6	85.7	70.6	83.6	77.0	72.5	80.4	38.3	68.7
Our method												
COMPETERL	87.5	**94.4**	90.3	**87.0**	**90.5**	**76.4**	**88.1**	**82.4**	**76.6**	**92.2**	**42.5**	**74.8**

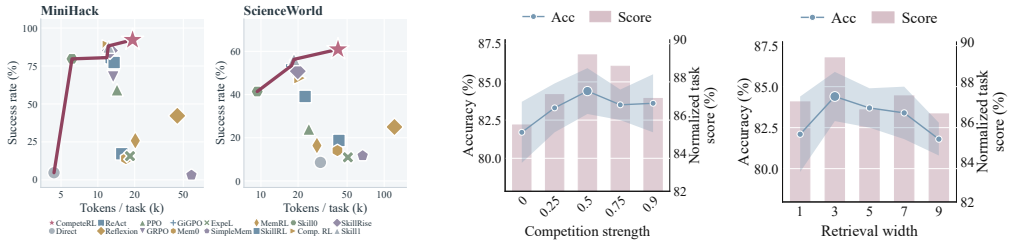

Figure 3: Inference efficiency and WebShop sensitivity. **Left:** performance–cost frontiers on MiniHack and ScienceWorld. **Right:** WebShop competition strength λ_c and retrieval width K; blue lines show success rate, rose bars show normalized task score, and shaded bands show variability.

30, and 30 interaction steps on ALFWorld, WebShop, MiniHack, and ScienceWorld, respectively, with further implementation details in Appendix B.2.

4.2 OVERALL PERFORMANCE

Main results. In Table 1, COMPETERL achieves the highest reported aggregate success rates on all four benchmarks: 97.0% on ALFWorld, 84.4% on WebShop, 91.4% on MiniHack, and 60.9% on ScienceWorld. These results exceed the strongest listed baseline on each benchmark by 2.2, 1.5, 3.2, and 4.6 percentage points, respectively. On WebShop, COMPETERL also obtains a normalized score of 89.2, compared with 87.0 for Complementary RL, improving both task score and exact success. Especially on the most challenging ScienceWorld dataset, COMPETERL achieved a 4.6% improvement compared to others, showing our competition mechanism can distill sharper insights.

The Qwen3-4B results in Table 2 show the same overall advantage, with COMPETERL obtaining the highest aggregate success rate among the evaluated methods on each benchmark and an average improvement of 6.1%. We noticed that the updated qwen3-4b model performs worse than the qwen2.5-7b-instruct on several benchmarks. This might be due to the smaller model size and the model not being an instruct version. This is consistent with the results found in other papers (Yao et al., 2026; Tan et al., 2026). Together with the Qwen2.5-7B results, this second-backbone evaluation shows that the observed advantage is not confined to a single policy model.

Inference performance and cost. Figure 3 (left) compares success rate against generated tokens per task of Qwen2.5-7b-instruct on MiniHack and ScienceWorld, with ALFWorld and WebShop results provided in Appendix C. COMPETERL reaches the Pareto front across multiple datasets, achieving an optimal trade-off between performance and overhead. As shown in Figure 7, there was 2% more inference overhead compared to SkillRise, but COMPETERL gained a 7.4% performance boost on ALFWorld.

Table 3: Component ablations on ALFWorld and WebShop. Relative to the complete COMPETERL configuration, green downward arrows report drops, and red upward arrows report improvements.

	ALFWorld							WebShop	
Method	Pick	Look	Clean	Heat	Cool	Pick2	Avg.	Score	SR
2× actor GRPO (w/o skill)	79.2 $_{\downarrow 12.5}$	83.3 $_{\downarrow 16.7}$	83.9 $_{\downarrow 12.9}$	82.6 $_{\downarrow 12.6}$	81.0 $_{\downarrow 19.0}$	70.6 $_{\downarrow 29.4}$	80.6 $_{\downarrow 16.4}$	83.0 $_{\downarrow 6.2}$	76.6 $_{\downarrow 7.8}$
Static retrieval (w/o adaptation)	87.5 $_{\downarrow 4.2}$	94.4 $_{\downarrow 5.6}$	96.8 $_{\downarrow 0.0}$	91.3 $_{\downarrow 3.9}$	95.2 $_{\downarrow 4.8}$	76.5 $_{\downarrow 23.5}$	91.0 $_{\downarrow 6.0}$	77.8 $_{\downarrow 11.4}$	74.2 $_{\downarrow 10.2}$
Foresight only (w/o hindsight)	83.3 $_{\downarrow 8.4}$	77.8 $_{\downarrow 22.2}$	87.1 $_{\downarrow 9.7}$	87.0 $_{\downarrow 8.2}$	90.5 $_{\downarrow 9.5}$	82.4 $_{\downarrow 17.6}$	85.1 $_{\downarrow 11.9}$	84.7 $_{\downarrow 4.5}$	78.1 $_{\downarrow 6.3}$
Hindsight only (w/o foresight)	87.5 $_{\downarrow 4.2}$	94.4 $_{\downarrow 5.6}$	90.3 $_{\downarrow 6.5}$	91.3 $_{\downarrow 3.9}$	85.7 $_{\downarrow 14.3}$	88.2 $_{\downarrow 11.8}$	89.6 $_{\downarrow 7.4}$	85.9 $_{\downarrow 3.3}$	79.7 $_{\downarrow 4.7}$
No competition ($\lambda_c = 0$)	91.7 $_{\downarrow 0.0}$	88.9 $_{\downarrow 11.1}$	93.5 $_{\downarrow 3.3}$	95.7 $_{\uparrow 0.5}$	95.2 $_{\downarrow 4.8}$	88.2 $_{\downarrow 11.8}$	92.5 $_{\downarrow 4.5}$	85.5 $_{\downarrow 3.7}$	81.3 $_{\downarrow 3.1}$
COMPETERL	91.7	100.0	96.8	95.2	100.0	100.0	97.0	89.2	84.4

4.3 FRAMEWORK ANALYSIS

Component ablations. In Table 3, *w/o adaptation* retains retrieval and supplies the retrieved experience directly to the actor, but does not generate adapted experience for the current task. The *w/o foresight* variant removes the entire foresight branch, including experience retrieval, adaptation, and execution guided by foresight experience. The *w/o hindsight* variant removes hindsight distillation and the subsequent execution guided by the distilled experience. The *no competition* variant retains both branches but sets $\lambda_c = 0$, so each experience generator receives only the return from its own guided execution. The 2× actor GRPO control performs two actor executions per task without skills.

As shown in Table 3, hindsight distillation is the most critical component of COMPETERL : removing it produces the largest average drop across the two benchmarks, reducing success by 11.9 percentage points on ALFWorld and 6.3 points on WebShop. Static retrieval without task-specific adaptation is also brittle, particularly on WebShop, where success drops by 10.2 points. This suggests that irrelevant retrieved experience can contaminate the actor context and severely degrade performance, highlighting the importance of task-specific adaptation for effective experience transfer. Overall, the consistent performance degradation across all ablations demonstrates that every component contributes meaningfully to COMPETERL .

Hyperparameter sensitivity. We analyze the two key hyperparameters of COMPETERL : the competition coefficient $\lambda_c \in \{0, 0.25, 0.5, 0.75, 0.9\}$ and the retrieval width $K \in \{1, 3, 5, 7, 9\}$. We report the WebShop results in Figure 3 and the remaining benchmarks in Appendix D. Overall, $\lambda_c = 0.5$ and $K = 3$ provide the strongest shared operating point across benchmarks. For λ_c, moderate competition generally outperforms either no competition or overly strong competition, while retrieving three candidates achieves the highest observed success rate on all four benchmarks. The optimal competition strength nevertheless varies slightly across environments: ALFWorld performs similarly at $\lambda_c = 0.25$ and 0.5, whereas ScienceWorld peaks at 0.25. This variation is consistent with their different foresight–hindsight dynamics. On ALFWorld, hindsight initially holds a pronounced advantage and foresight gradually catches up during training (Figure 4 and Figure 10). An excessively large λ_c may therefore over-penalize foresight skill generation, weakening the complementary learning between the two branches and ultimately reducing task performance.

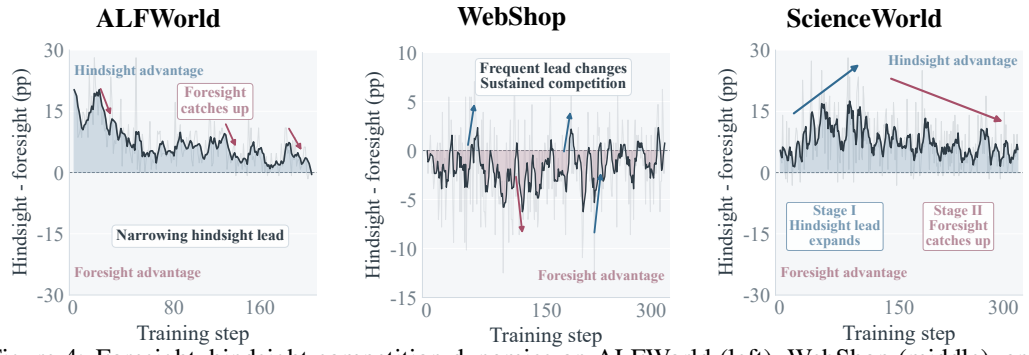

Figure 4: Foresight–hindsight competition dynamics on ALFWorld (left), WebShop (middle), and ScienceWorld (right). The vertical axis is the hindsight-minus-foresight success-rate gap; positive values favor hindsight. Dark curves show trailing five-step means and faint curves show raw values.

Foresight and hindsight competition dynamics. Figure 4 tracks the difference between hindsight and foresight success rates during training. We identify three major patterns of competitive evolu-

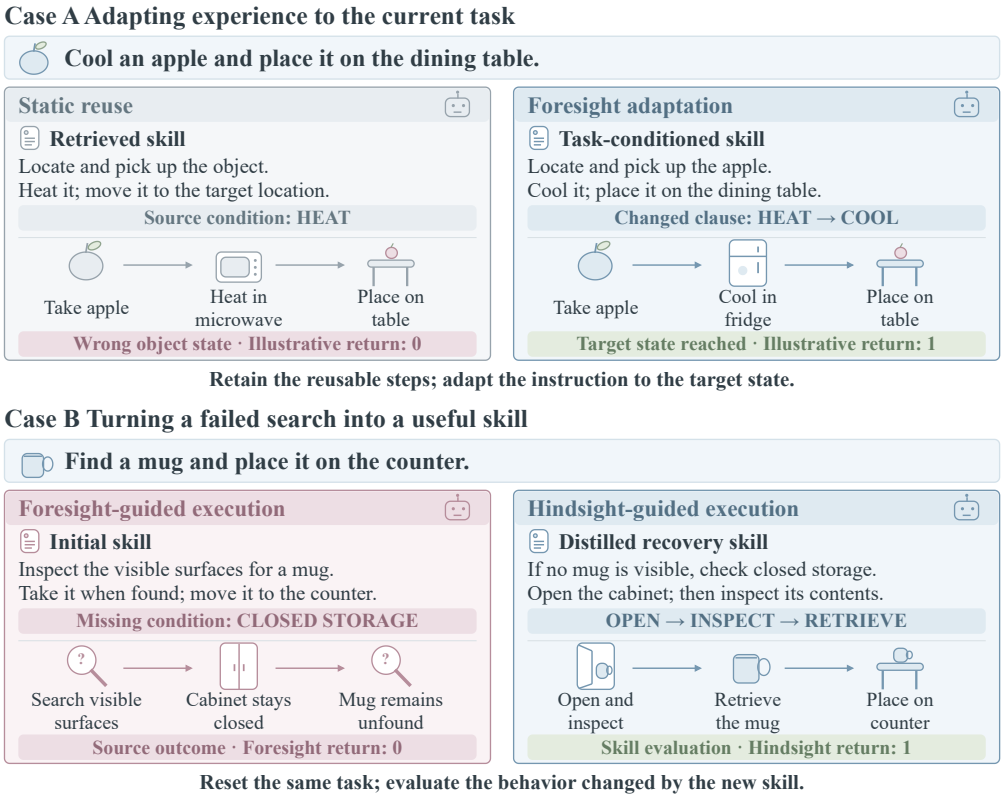

Figure 5: Case studies on ALFWorld.

tion: (1) foresight experience consistently trails and gradually approaches hindsight experience; (2) foresight and hindsight experiences improve in an alternating manner; and (3) hindsight experience first expands its advantage, after which foresight experience gradually catches up. Despite these different dynamics, all three patterns share a common trend as shown in Figure 10: the joint success rate of the two experience sources steadily increases, indicating their synergistic co-evolution.

4.4 Case Study

Figure 5 presents two real case studies from ALFWorld. Case (a) demonstrates how retrieved experience is adapted to the current task. The adaptation transforms a retrieved heating procedure into guidance for a cooling task while preserving relevant object handling steps. Case (b) shows a recorded evaluation trajectory of hindsight experience. The distilled experience from a failed search is reused to guide a new execution of the same task. In the initial attempt, the actor overlooks the closed storage, whereas the distilled hindsight experience instructs the actor to open and inspect the cabinet before retrieving the mug.

5 Conclusion

We introduced Experience-in-the-Loop Learning, a formulation that places experience adaptation, utilization, and distillation within the policy rollout so that generated experience can be optimized by the execution it guides. Based on this formulation, CompeteRL couples task-conditioned foresight experience with trajectory-grounded hindsight experience and trains the two through task-matched competitive feedback. Across four interactive agent benchmarks and two policy backbones, CompeteRL achieves strong task performance, while the analyses illustrate the complementary evolution of foresight and hindsight guidance. These results suggest that jointly learning how to act and how to construct the experience that guides action is critical for continually improving agents.

AI USE STATEMENT

We used OpenAI Codex to assist with manuscript language refinement. Codex also supported parts of the code implementation, which multiple authors subsequently reviewed and verified. Additionally, it assisted in generating certain graphical elements for the method overview figure 2 and in reviewing the logical consistency of the algorithmic workflow and the rigor of theoretical derivations. For experiments, AI was employed only as a secondary verification tool after human researchers had collected and initially validated the results, helping assess the consistency and rigor of the experimental findings and analyses.

ETHICS STATEMENT

We have not identified any distinctive ethical risks arising from this work beyond those generally associated with LLM-based agents, including unreliable outputs, bias, privacy concerns, and potential misuse. These considerations remain important when generating and deploying agent skills or experiences in real-world systems. Our approach does not introduce additional ethical risks beyond these existing concerns.

REPRODUCIBILITY STATEMENT

Our code is anonymously available at https://anonymous.4open.science/r/CompeteRLICLR27-CC5F. The implementation details can be found in B.2.

REFERENCES

Thomas Palmeira Ferraz, Romain Deffayet, Vassilina Nikoulina, Hervé Déjean, and Stéphane Clinchant. Retrieval-augmented llm agents: Learning to learn from experience, 2026. URL https://arxiv.org/abs/2603.18272.

Yifu Guo, Jiaye Lin, Huacan Wang, Yuzhen Han, Sen Hu, Ziyi Ni, Licheng Wang, and Mingguang Chen. Se-agent: Self-evolution trajectory optimization in multi-step reasoning with llm-based agents. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 116314–116341. Curran Associates, Inc., 2025. doi: 10.52202/085713-3877. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/a911e543a95493ae5004fdc01909043e-Paper-Conference.pdf.

Zelin He, Haotian Lin, Boran Han, Wei Zhu, Haoyang Fang, Bernie Wang, Xuan Zhu, Runze Li, and Matthew Reimherr. Reskill: Reconciling skill creation with policy optimization in agentic rl, 2026. URL https://arxiv.org/abs/2606.01619.

Ruofei Ju, Xinrui Wang, Xin Ding, Yifan Yang, Hao Wu, Shiqi Jiang, Qianxi Zhang, Hao Wen, Xiangyu Li, Weijun Wang, Kun Li, Yunxin Liu, Haipeng Dai, Wei Wang, and Ting Cao. Embodiskill: Skill-aware reflection for self-evolving embodied agents, 2026. URL https://arxiv.org/abs/2605.10332.

Qiliang Liang, Hansi Wang, Zhong Liang, and Yang Liu. From skill text to skill structure: The scheduling-structural-logical representation for agent skills, 2026. URL https://arxiv.org/abs/2604.24026.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Simplemem: Efficient lifelong memory for llm agents, 2026. URL https://arxiv.org/abs/2601.02553.

Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Chengcheng Han, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Skill0: In-context agentic reinforcement learning for skill internalization, 2026. URL https://arxiv.org/abs/2604.02268.

Wenquan Ma, Jiayan Nan, and WenLong Wu. What deserves memory: Adaptive memory distillation for LLM agents. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume*

1: Long Papers), pp. 34789–34812, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1607. URL https://aclanthology.org/2026.acl-long.1607/.

Dilxat Muhtar, Jiashun Liu, Wei Gao, Weixun Wang, Shaopan Xiong, Ju Huang, Siran Yang, Wenbo Su, Jiamang Wang, Ling Pan, and Bo Zheng. Complementary rl: Towards efficient experience-driven agent learning, 2026. URL https://arxiv.org/abs/2603.17621.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory, 2026. URL https://arxiv.org/abs/2509.25140.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Mikayel Samvelyan, Robert Kirk, Vitaly Kurin, Jack Parker-Holder, Minqi Jiang, Eric Hambro, Fabio Petroni, Heinrich Küttler, Edward Grefenstette, and Tim Rocktäschel. Minihack the planet: A sandbox for open-ended reinforcement learning research. In Joaquin Vanschoren and Sai-Kit Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/fa7cdfad1a5aaf8370ebeda47a1ff1c3-Abstract-round1.html.

Junhao Shen, Teng Zhang, Xiaoyan Zhao, and Hong Cheng. Dynamic skill lifecycle management for agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2605.10923.

Yaorui Shi, Yuxin Chen, Zhengxi Lu, Yuchun Miao, Shugui Liu, Qi GU, Xunliang Cai, Xiang Wang, and An Zhang. Skill1: Unified evolution of skill-augmented agents via reinforcement learning, 2026. URL https://arxiv.org/abs/2605.06130.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Shuzheng Si, Haozhe Zhao, Yu Lei, Qingyi Wang, Dingwei Chen, Zhitong Wang, Zhenhailong Wang, Kangyang Luo, Zheng Wang, Gang Chen, Fanchao Qi, Minjia Zhang, and Maosong Sun. From context to skills: Can language models learn from context skillfully?, 2026. URL https://arxiv.org/abs/2604.27660.

Qitai Tan, Zefang Zong, Mo Li, Yipeng Shi, Yang Li, and Peng Chen. Atod: Annealed turn-aware on-policy distillation for multi-turn agentic tasks, 2026. URL https://arxiv.org/abs/2606.27814.

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. *Trans. Mach. Learn. Res.*, 2024, 2024a. 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, 2026a. URL https://arxiv.org/abs/2604.10674.

Ruoyao Wang, Peter A. Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 11279–11298. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.EMNLP-MAIN.775. URL https://doi.org/10.18653/v1/2022.emnlp-main.775.

Zixuan Wang, Yuchen Yan, Hongxing Li, Teng Pan, Dingming Li, Ruiqing Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Milestone-guided policy learning for long-horizon language agents, 2026b. URL https://arxiv.org/abs/2605.06078.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory, 2024b. URL https://arxiv.org/abs/2409.07429.

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. Seed: Self-evolving on-policy distillation for agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2607.14777.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning, 2026. URL https://arxiv.org/abs/2602.08234.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schuetze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12805–12825, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.583. URL https://aclanthology.org/2026.acl-long.583/.

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, 2025a. URL https://arxiv.org/abs/2505.09388.

Cheng Yang, Xuemeng Yang, Licheng Wen, Daocheng Fu, Jianbiao Mei, Rong Wu, Pinlong Cai, Yufan Shen, Nianchen Deng, Botian Shi, Yu Qiao, and Haifeng Li. Learning on the job: An experience-driven self-evolving agent for long-horizon tasks, 2025b. URL https://arxiv.org/abs/2510.08002.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering, 2024. URL https://arxiv.org/abs/2405.15793.

Shuo Yang, Jinyang Wu, Zhengxi Lu, Yuhao Shen, Fan Zhang, Lang Feng, Shuai Zhang, Haoran Luo, Zheng Lian, Zhengqi Wen, and Jianhua Tao. Opid: On-policy skill distillation for agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2606.26790.

Wei Yang, Jinwei Xiao, Hongming Zhang, Qingyang Zhang, Yanna Wang, and Bo Xu. Coarse-to-fine grounded memory for LLM agent planning. In Christos Christodoulopoulos, Tanmoy

Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 13029–13056, Suzhou, China, November 2025c. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.659. URL https://aclanthology.org/2025.emnlp-main.659/.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

Zhiyuan Yao, Yuxin Chen, Zhengxi Lu, Zishan Xu, Yueqing Sun, Yifu Guo, Yuquan Lu, Zhengzhou Cai, Kangning Zhang, Zhuowen Han, Zi-Han Wang, Ziang Ye, Qi Gu, Xunliang Cai, Weiwen Liu, and Yongliang Shen. Skillrise: Agentic reinforcement learning for cross-task skill evolution, 2026. URL https://arxiv.org/abs/2607.26784.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. Memgen: Weaving generative latent memory for self-evolving agents, 2025a. URL https://arxiv.org/abs/2509.24704.

Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. Memevolve: Meta-evolution of agent memory systems, 2025b. URL https://arxiv.org/abs/2512.18746.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents, 2026a. URL https://arxiv.org/abs/2602.02474.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, Bo Tang, and Muning Wen. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory, 2026b. URL https://arxiv.org/abs/2601.03192.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models, 2025c. URL https://arxiv.org/abs/2506.05176.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: LLM agents are experiential learners. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, pp. 19632–19642. AAAI Press, 2024. doi: 10.1609/AAAI.V38I17.29936. URL https://doi.org/10.1609/aaai.v38i17.29936.

Xinyu Zhao, Zhen Tan, Vaishnav Tadiparthi, Nakul Agarwal, Kwonjoon Lee, Ehsan Moradi Pari, Hossein Nourkhiz Mahjoub, and Tianlong Chen. Generative skill composition for llm agents, 2026. URL https://arxiv.org/abs/2606.32025.

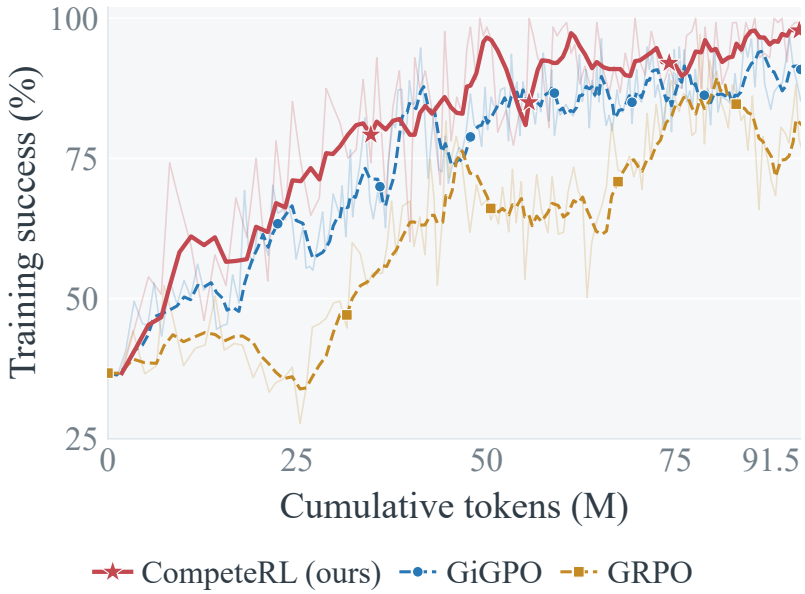

Figure 6: ALFWorld training success versus cumulative generated tokens for COMPETERL , GRPO, and GiGPO.

A LIMITATIONS

Due to the availability of open model weights and the computational resources required for reinforcement learning, our experiments are limited to open-weight policy backbones and do not evaluate the powerful proprietary models currently used in production systems. In addition, our evaluation focuses on text-based interactive benchmarks, leaving the generalization of Experience-in-the-Loop Learning to multimodal environments and real-world agent deployments for future study.

B EXPERIMENTAL PROTOCOL DETAILS

B.1 EVALUATION MANIFESTS AND BENCHMARK DETAILS

ALFWorld OOD evaluation. Our primary ALFWorld evaluation uses the complete official OOD `valid_unseen` split rather than a sampled subset. The resulting frozen 134-instance manifest is identified as `LOCAL-ALF-OOD-134`. It contains 24 Pick, 18 Look, 31 Clean, 23 Heat, 21 Cool, and 17 Pick-Two episodes. The aggregate ALFWorld metric is the micro success rate over all 134 episodes; Table 1 additionally reports the success rate for each task family. All matching local methods will be evaluated on the same dataset with the same interaction limits, prioritising the use of officially reported metrics, model parameters and code

WebShop evaluation. Following the evaluation configurations of SkillRL (Xia et al., 2026) and Skill1 (Shi et al., 2026), we use a fixed set of 128 tasks from the standard WebShop evaluation split and allow at most 20 environment interactions per task. We report both the benchmark's normalized task score on a 0–100 scale, which preserves partial credit, and exact success rate, which requires the purchased item and selected options to satisfy the complete instruction.

MiniHack evaluation. Following Complementary RL (Muhtar et al., 2026), we evaluate `MiniHack-Room-Ultimate-5x5-v0` with fixed held-out environment seeds and a 30-step interaction limit. An episode is counted as correct only when the environment reports a win within this budget; otherwise it is counted as a failure.

ScienceWorld evaluation. We follow BEACON's L0 ScienceWorld configuration (Wang et al., 2026b): the released 3,322/1,661 train/test variation membership, and a 30-step interaction limit. Evaluation uses a deterministic 128-instance selection from the L0 test set following BEACON. We

adopt a stricter correctness criterion than positive partial progress: an episode is counted as correct only if its final official ScienceWorld score is exactly 100; all lower scores are failures.

B.2 Implementation Details

Training infrastructure. We implement CompeteRL in the verl framework with Fully Sharded Data Parallel training in BFloat16 precision and gradient checkpointing. Each run uses one four-GPU node; rollout generation is served by vLLM with tensor parallelism of 4. For each paired foresight–hindsight sample, both executions use the same task, while the environment is reset before the hindsight-conditioned execution. Policy checkpoints are stored together with the corresponding skill-library snapshots so that evaluation restores the matched policy–library state.

Shared hyperparameters. Unless otherwise specified, all environments use GRPO, a task batch size of 16, group size $G = 8$, learning rate 1×10^{-6}, low-variance KL regularization with coefficient 0.01, and a PPO mini-batch size of 256. Training rollouts use temperature 1.0 and top-p 1.0. The retriever returns the top $K = 3$ candidates, the utility exponential-moving-average coefficient is $\alpha = 0.05$, and the competition coefficient is $\lambda_c = 0.5$. Table 4 lists the settings that vary by environment; in particular, training length is not shared across benchmarks.

Table 4: Environment-specific training and evaluation settings. "Training steps" denotes the maximum number of optimizer updates, not environment interaction steps.

Environment	Training steps	Max env. steps	Max prompt tokens	Max response tokens	PPO micro-batch/GPU	Eval. temperature
ALFWorld	200	50	12,288	2,048	8	0.0
WebShop	300	20	12,288	2,048	8	0.0
MiniHack	200	30	12,288	2,048	8	0.0
ScienceWorld	300	30	15,000	512	1	0.0

ALFWorld and MiniHack are therefore trained for at most 200 optimization steps, whereas WebShop and ScienceWorld use at most 300 steps. Environment-specific interaction limits are enforced by the corresponding wrappers and apply separately to each foresight- or hindsight-conditioned execution.

C Additional Inference Performance–Cost Frontiers

Figure 7 adds ALFWorld and WebShop to the main-text performance–cost comparison. On each panel, CompeteRL has the highest displayed success rate and lies on the displayed frontier, while lower-cost configurations remain non-dominated at lower success rates. The plotted costs therefore characterize a trade-off rather than a uniform reduction in token use.

D Additional Hyperparameter Sensitivity Results

Figure 8 extends the one-factor sweeps to ALFWorld, MiniHack, and ScienceWorld. All three environments improve over the zero-competition setting at a moderate coefficient. ALFWorld reaches a plateau across 0.25 and 0.5, MiniHack peaks at 0.5, and ScienceWorld peaks at 0.25. The lower point estimates at 0.9 show that stronger competition does not monotonically improve task success. The retrieval-width comparison is more consistent across environments: $K = 3$ gives the highest displayed value, while larger candidate sets yield lower point estimates. Together with the WebShop results, these sweeps support the chosen defaults as a shared operating point, without implying that $\lambda_c = 0.5$ is optimal for every environment.

E Additional Foresight–Hindsight Competition Dynamics

Figure 9 shows repeated changes in the sign of the foresight–hindsight success gap on MiniHack. Figure 10 decomposes paired executions into shared failure, hindsight-only success, foresight-only success, and shared success. If p_{ab} denotes the frequency of outcome (a, b), then foresight and hindsight success rates are $p_{10} + p_{11}$ and $p_{01} + p_{11}$; their difference is $p_{01} - p_{10}$. Thus, the gap plot records only the imbalance between the two one-sided success outcomes, while the paired-outcome plot also exposes changes in joint success and failure. On ALFWorld and ScienceWorld, the growing shared-success region provides context for the narrowing late-training gap. These are

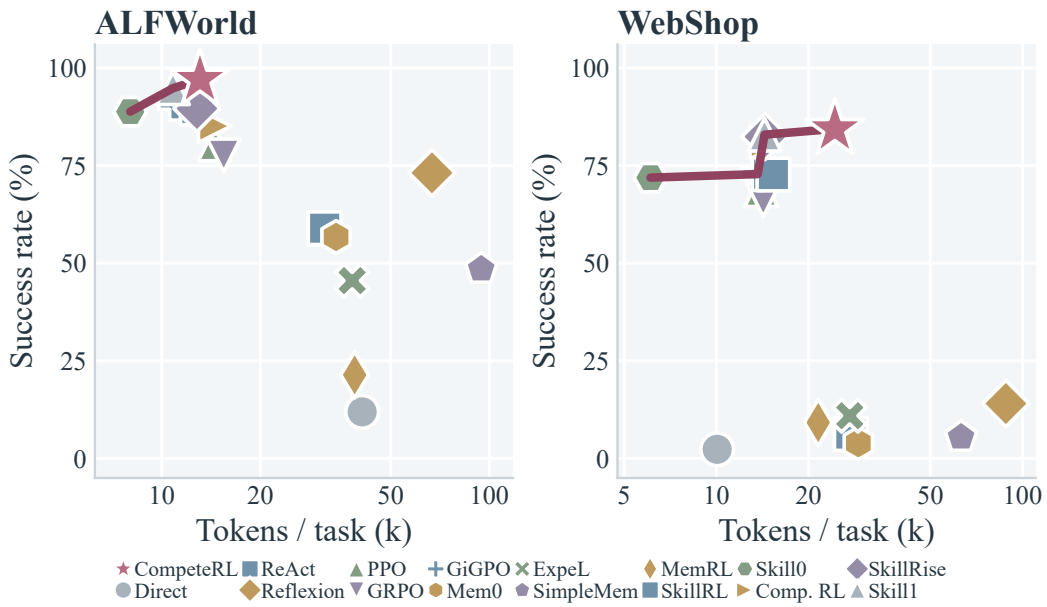

Figure 7: Inference performance–cost frontiers on ALFWorld (left) and WebShop (right).

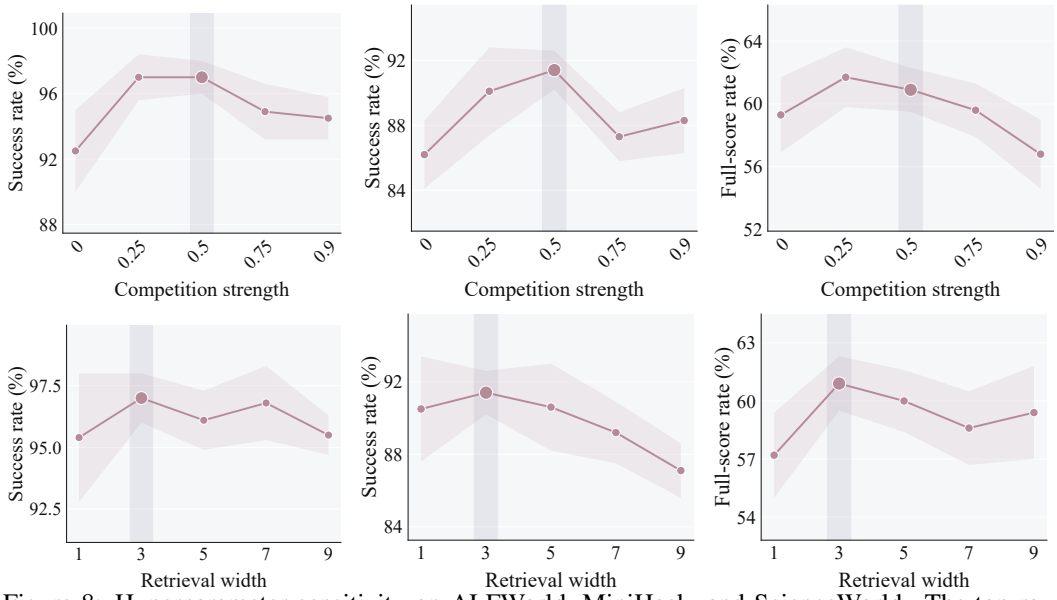

Figure 8: Hyperparameter sensitivity on ALFWorld, MiniHack, and ScienceWorld. The top row varies competition strength and the bottom row varies retrieval width. Shaded strips mark the default settings, and bands show variability.

descriptive within-run observations, with the smoothing and interpolation conventions stated in the captions.

F Additional Training Analysis

Training progress. On ALFWorld and ScienceWorld, CompeteRL reaches higher success rates toward the end of training than GRPO and GiGPO in Figure 11. The ALFWorld advantage is also visible when training progress is measured by cumulative generated tokens in Figure 6. This shows that our algorithm can achieve better results even with the same training budget

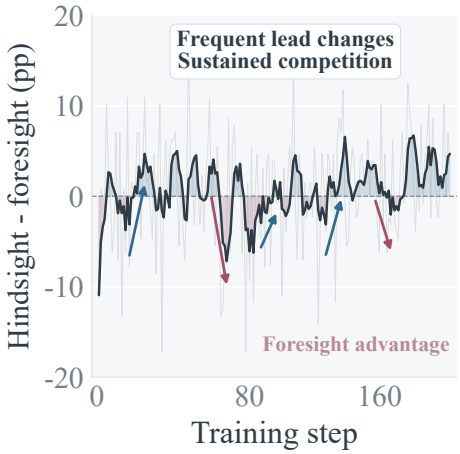

Figure 9: Foresight–hindsight competition dynamics on MiniHack. The vertical axis is the hindsight-minus-foresight success-rate gap; the dark curve shows a trailing five-step mean and the faint curve shows raw values.

This appendix analyzes the competitive rewards in Eq. (6) under the notation of the main method. Uppercase R_i^{F} and R_i^{H} denote the two task returns, lowercase r_i^{F} and r_i^{H} denote the competitive training rewards, and $0 \le \lambda_{\mathrm{c}} < 1$. Our analysis establishes four properties of foresight–hindsight competition. First, the matched competitive reward preserves the operational-utility objective of skill generation. Second, same-task comparison removes the additive influence of task difficulty. Third, the resulting rewards retain a non-zero-sum incentive for joint success. Finally, phase masking and the shared-policy geometry preserve local credit attribution and align competitive optimization with joint execution success.

F.1 Operational Skill Utility and Task-Difficulty Cancellation

We first establish that the competitive reward continues to optimize the behavioral utility of the generated skill, rather than replacing it with a purely relative objective. We define the operational utility of a hindsight skill by the behavior it induces after the environment is reset

$$U_\theta\big(e_i^{\mathrm{H}}; d_i^{\mathrm{F}}\big) = \mathbb{E}_{\tau_i^{\mathrm{H}} \sim P_{\mathcal{E},\theta}(\cdot \mid x, e_i^{\mathrm{H}}, \omega_{\mathrm{A}})}\big[\mathcal{R}\big(x, \tau_i^{\mathrm{H}}\big)\big]\,. \tag{14}$$

The corresponding hindsight-skill objective is

$$J_{\mathrm{H}}^{\mathrm{skill}}(\theta) = \mathbb{E}_{e_i^{\mathrm{H}} \sim \pi_\theta(\cdot \mid d_i^{\mathrm{F}}, \omega_{\mathrm{H}})}\big[U_\theta\big(e_i^{\mathrm{H}}; d_i^{\mathrm{F}}\big)\big]\,. \tag{15}$$

The following theorem shows that subtracting the matched foresight return does not change the expected operational-utility gradient of the hindsight skill generator.

Theorem F.1 (Operational-utility preservation)**.** *For the masked score-function gradient assigned to the hindsight-skill tokens, replacing R_i^{H} by $r_i^{\mathrm{H}} = R_i^{\mathrm{H}} - \lambda_{\mathrm{c}} R_i^{\mathrm{F}}$ leaves the expected gradient of Eq. (15) unchanged. Specifically,*

$$\mathbb{E}\big[r_i^{\mathrm{H}} \nabla_\theta \log \pi_\theta\big(e_i^{\mathrm{H}} \mid d_i^{\mathrm{F}}, \omega_{\mathrm{H}}\big)\big] = \mathbb{E}\big[R_i^{\mathrm{H}} \nabla_\theta \log \pi_\theta\big(e_i^{\mathrm{H}} \mid d_i^{\mathrm{F}}, \omega_{\mathrm{H}}\big)\big]\,. \tag{16}$$

Proof. The first execution and R_i^{F} are observed before e_i^{H} is sampled. Conditioned on d_i^{F}, the foresight return is therefore independent of the hindsight-skill sampling action, and

$$\mathbb{E}\big[R_i^{\mathrm{F}} \nabla_\theta \log \pi_\theta\big(e_i^{\mathrm{H}} \mid d_i^{\mathrm{F}}, \omega_{\mathrm{H}}\big) \mid d_i^{\mathrm{F}}\big] = R_i^{\mathrm{F}} \nabla_\theta \int \pi_\theta\big(e^{\mathrm{H}} \mid d_i^{\mathrm{F}}, \omega_{\mathrm{H}}\big)\, \mathrm{d}e^{\mathrm{H}} = 0. \tag{17}$$

Subtracting λ_{c} times this zero-expectation term from the hindsight return proves Eq. (16). Thus R_i^{F} acts as a task-matched baseline for hindsight-skill generation, while the actor phase continues to receive its own return through the separate phase mask. □

Therefore, hindsight competition changes the reference used for credit assignment without changing the expected operational-utility objective optimized by the hindsight skill generator.

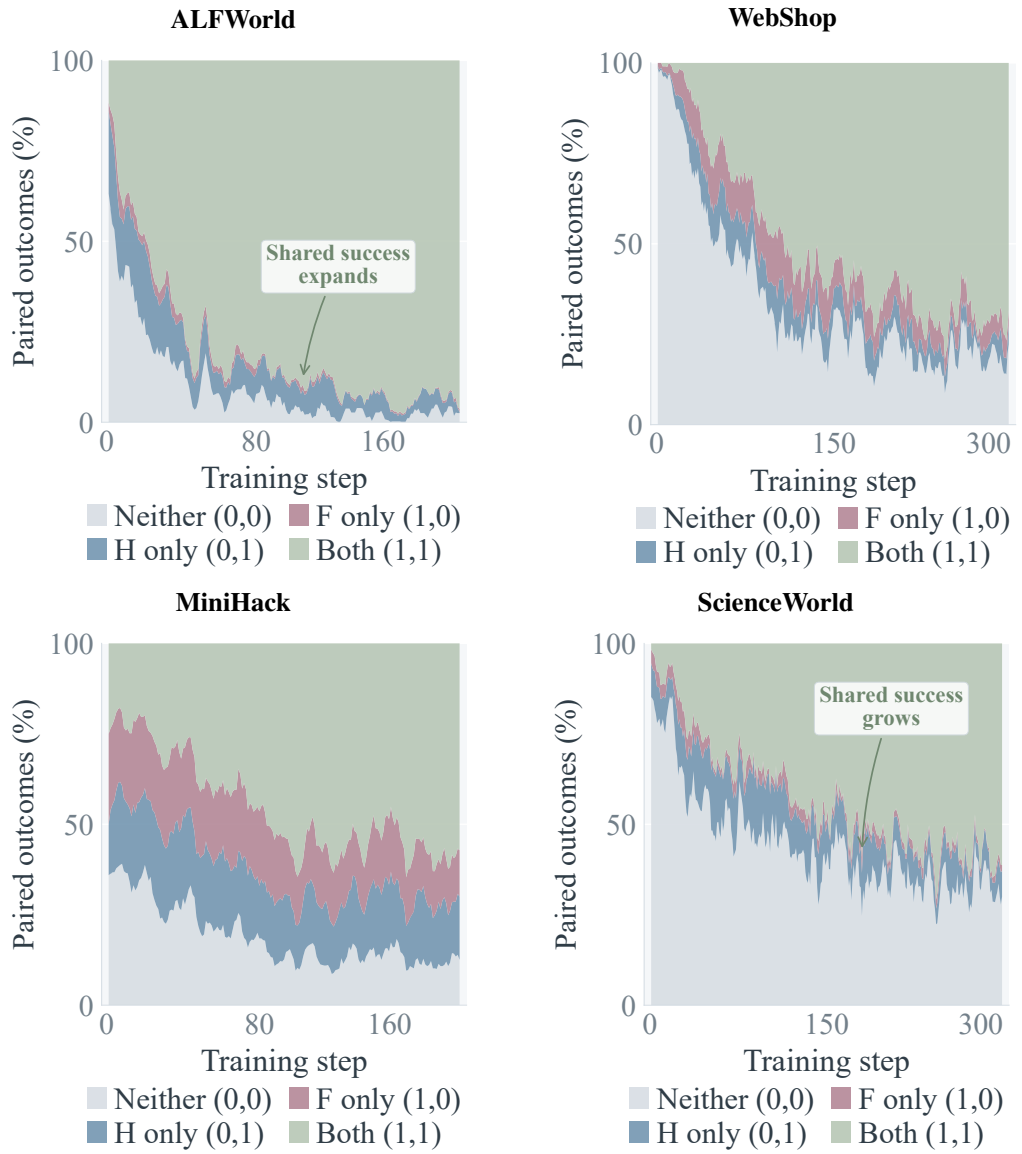

Figure 10: Paired foresight–hindsight outcomes on ALFWorld, WebShop, MiniHack, and ScienceWorld. Outcomes $(0,0)$, $(0,1)$, $(1,0)$, and $(1,1)$ denote shared failure, hindsight-only success, foresight-only success, and shared success, respectively. Stacked frequencies show trailing five-step means.

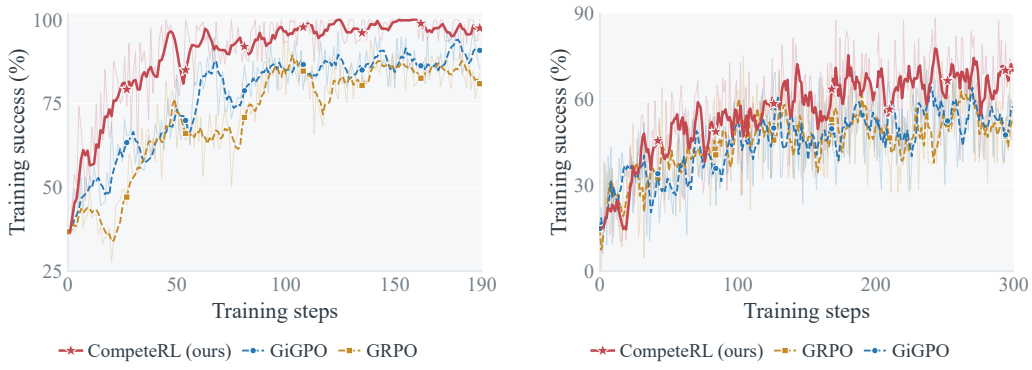

Figure 11: Training success versus optimization steps for COMPETERL , GRPO, and GiGPO on ALFWorld (left) and ScienceWorld (right).

Raw task returns also reflect task difficulty and actor execution noise. We next establish that the additive task-level component shared by the matched executions is removed by the same-task normalization used by COMPETERL .

Lemma F.2 (Cancellation of additive task difficulty)**.** *Suppose the two returns on task x admit the decomposition*

$$R_i^p = b(x) + v_i^p + \varepsilon_i^p, \qquad p \in \{\mathrm{F}, \mathrm{H}\}, \tag{18}$$

where $b(x)$ is an additive task-level difficulty effect, v_i^p is the skill-conditioned utility, and ε_i^p is execution noise. Then the phase-wise advantage in Eq. (9) is invariant to $b(x)$.

Proof. Let $\bar{p}$ denote the competing skill phase. The competitive reward satisfies

$$r_i^p = (1-\lambda_{\mathrm{c}})b(x) + v_i^p - \lambda_{\mathrm{c}} v_i^{\bar{p}} + \varepsilon_i^p - \lambda_{\mathrm{c}}\varepsilon_i^{\bar{p}}. \tag{19}$$

Because all G rollouts in the comparison group share task x, subtracting $\mu_x^p = G^{-1}\sum_{j=1}^{G} r_j^p$ removes $(1-\lambda_{\mathrm{c}})b(x)$ exactly. The standard deviation σ_x^p is also invariant to this additive shift, so A_i^p is unchanged. Equivalently, under independence between $b(x)$ and the remaining terms,

$$\mathrm{Var}(r^p) = (1-\lambda_{\mathrm{c}})^2 \,\mathrm{Var}_x[b(x)] + \mathrm{Var}\left(v^p - \lambda_{\mathrm{c}} v^{\bar{p}} + \varepsilon^p - \lambda_{\mathrm{c}}\varepsilon^{\bar{p}}\right), \tag{20}$$

and same-task centering removes the first term. □

Hence the resulting advantage is invariant to the additive difficulty shared by the matched task and attributes within-task performance variation more directly to the skill conditions and their induced executions.

F.2 POSITIVE COOPERATIVE POTENTIAL OF COMPETITIVE REWARDS

We next determine whether foresight–hindsight competition forms a zero-sum objective that could reward improving one skill condition by sacrificing the other. Let $\mathbf{R}_i = [R_i^{\mathrm{F}}, R_i^{\mathrm{H}}]^\top$ and $\mathbf{r}_i = [r_i^{\mathrm{F}}, r_i^{\mathrm{H}}]^\top$. The competitive reward in Eq. (6) is equivalently

$$\mathbf{r}_i = \mathbf{K}_{\lambda_{\mathrm{c}}}\mathbf{R}_i, \qquad \mathbf{K}_{\lambda_{\mathrm{c}}} = \begin{bmatrix} 1 & -\lambda_{\mathrm{c}} \\ -\lambda_{\mathrm{c}} & 1 \end{bmatrix}. \tag{21}$$

The following result shows that the reward transformation preserves a positive joint-success component while strengthening the relative signal between the matched skill conditions.

Theorem F.3 (Positive cooperative potential)**.** *For $0 \le \lambda_{\mathrm{c}} < 1$, the competition operator is positive definite and preserves a positive joint-success component while amplifying the matched performance difference. In particular, for every $\mathbf{R}_i \neq \mathbf{0}$,*

$$\mathbf{R}_i^\top \mathbf{r}_i = \frac{1-\lambda_{\mathrm{c}}}{2}\left(R_i^{\mathrm{F}} + R_i^{\mathrm{H}}\right)^2 + \frac{1+\lambda_{\mathrm{c}}}{2}\left(R_i^{\mathrm{H}} - R_i^{\mathrm{F}}\right)^2 > 0. \tag{22}$$

Proof. With

$$\mathbf{Q} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}, \tag{23}$$

the symmetric operator has the eigendecomposition

$$\mathbf{Q}^\top \mathbf{K}_{\lambda_{\mathrm{c}}} \mathbf{Q} = \begin{bmatrix} 1-\lambda_{\mathrm{c}} & 0 \\ 0 & 1+\lambda_{\mathrm{c}} \end{bmatrix}. \tag{24}$$

Both eigenvalues are positive for $0 \le \lambda_{\mathrm{c}} < 1$. Expanding the resulting quadratic form gives Eq. (22). □

Thus the competitive transformation does not replace joint improvement with a zero-sum game. It preserves a positive common-success incentive while assigning additional credit according to the matched performance difference.

The same property can be stated directly at the aggregate reward level.

Corollary F.4 (Non-zero-sum aggregate reward)**.** *The two skill-construction rewards satisfy*

$$r_i^{\mathrm{F}} + r_i^{\mathrm{H}} = (1 - \lambda_{\mathrm{c}}) \left(R_i^{\mathrm{F}} + R_i^{\mathrm{H}} \right). \tag{25}$$

Including the two actor rewards yields the aggregate four-phase reward

$$\begin{aligned} \Phi_i^{\mathrm{all}} &= r_i^{\mathrm{F}} + R_i^{\mathrm{F}} + r_i^{\mathrm{H}} + R_i^{\mathrm{H}} \\ &= (2 - \lambda_{\mathrm{c}}) \left(R_i^{\mathrm{F}} + R_i^{\mathrm{H}} \right), \end{aligned} \tag{26}$$

and hence

$$\frac{\partial \Phi_i^{\mathrm{all}}}{\partial R_i^{\mathrm{F}}} = \frac{\partial \Phi_i^{\mathrm{all}}}{\partial R_i^{\mathrm{H}}} = 2 - \lambda_{\mathrm{c}} > 0. \tag{27}$$

Therefore, increasing either execution return strictly increases the aggregate four-phase reward.

Corollary F.5 (No incentive to sacrifice either execution)**.** *For binary task returns, joint success strictly dominates either one-sided success outcome*

$$\Phi^{\mathrm{all}}(1,1) - \Phi^{\mathrm{all}}(0,1) = 2 - \lambda_{\mathrm{c}} > 0, \qquad \Phi^{\mathrm{all}}(1,1) - \Phi^{\mathrm{all}}(1,0) = 2 - \lambda_{\mathrm{c}} > 0. \tag{28}$$

Thus intentionally degrading either execution cannot improve the aggregate four-phase reward.

Together, these results establish that foresight–hindsight competition is cooperative rather than zero-sum at the reward level. The aggregate objective strictly prefers joint success and provides no reward-level incentive to deliberately degrade either execution.

F.3 Phase-Local Attribution and Positive Alignment

The competitive rewards are constructed by comparing outcomes across phases, but this comparison should not mix the token-level attribution of the shared policy. We first establish that the phase masks change only the credit assigned to each phase and preserve the location to which that credit is applied.

Lemma F.6 (Phase-local token attribution)**.** *Let $\mathcal{I}_{i,p} = \{k : m_{i,k}^p = 1\}$ denote the token support of phase p. The masks in Eq. (11) satisfy*

$$\mathcal{I}_{i,p} \cap \mathcal{I}_{i,p'} = \emptyset \quad (p \neq p'), \qquad \sum_{p \in \mathcal{P}} m_{i,k}^p = 1. \tag{29}$$

Consequently, every generated token receives the advantage of exactly one phase even though the rewards are constructed by cross-phase comparison.

Proof. For a token $k \in \mathcal{I}_{i,p}$, disjointness gives $m_{i,k}^{p'} = 0$ for every $p' \neq p$. Therefore

$$\sum_{p' \in \mathcal{P}} m_{i,k}^{p'} A_i^{p'} = A_i^p, \tag{30}$$

and all other phase terms vanish from the token's clipped surrogate. Outcome comparison is cross-phase, whereas gradient attribution remains phase-local. □

Therefore, competition changes the magnitude of the learning signal assigned to each phase without assigning that signal to tokens generated by another phase.

Phase-local attribution alone does not yet establish that the resulting shared-policy update promotes joint success. We analyze this property through the local geometry of the four phase gradients.

Definition F.7 (Phase-whitened local geometry)**.** For a fixed task, define centered returns $\widehat{R}_i^p = R_i^p - G^{-1} \sum_{j=1}^G R_j^p$. Centering Eq. (6) gives $\widehat{r}_i^{\mathrm{F}} = \widehat{R}_i^{\mathrm{F}} - \lambda_{\mathrm{c}} \widehat{R}_i^{\mathrm{H}}$ and $\widehat{r}_i^{\mathrm{H}} = \widehat{R}_i^{\mathrm{H}} - \lambda_{\mathrm{c}} \widehat{R}_i^{\mathrm{F}}$. Absorb the positive phase-wise normalization factors in Eq. (9) into the four local masked score directions and collect them as

$$\mathbf{G}_\theta = [g_{\mathrm{F}} \quad g_{\mathrm{AF}} \quad g_{\mathrm{H}} \quad g_{\mathrm{AH}}]. \tag{31}$$

When $\mathbf{G}_\theta$ has full column rank, its phase-whitened representation is

$$\widetilde{\mathbf{G}}_\theta = \mathbf{G}_\theta \left(\mathbf{G}_\theta^\top \mathbf{G}_\theta \right)^{-1/2}, \qquad \widetilde{\mathbf{G}}_\theta^\top \widetilde{\mathbf{G}}_\theta = \mathbf{I}_4. \tag{32}$$

This defines the local geometry used for analysis and does not introduce an additional operation into the training algorithm.

The final theorem determines whether the competitive update remains aligned with simultaneously improving foresight- and hindsight-guided execution. It shows that competition is an ascent direction for joint success in the phase-whitened shared-policy geometry.

Theorem F.8 (Positive alignment with joint success)**.** *In the phase-whitened local geometry, the competitive update is positively aligned with the joint-success update whenever* $(\widehat{R}_i^{\mathrm{F}}, \widehat{R}_i^{\mathrm{H}}) \neq (0, 0)$.

Proof. The independent and competitive four-phase coefficients are

$$\mathbf{z}_i^0 = \begin{bmatrix} \widehat{R}_i^{\mathrm{F}} \\ \widehat{R}_i^{\mathrm{F}} \\ \widehat{R}_i^{\mathrm{H}} \\ \widehat{R}_i^{\mathrm{H}} \end{bmatrix}, \qquad \mathbf{z}_i^\lambda = \begin{bmatrix} \widehat{R}_i^{\mathrm{F}} - \lambda_{\mathrm{c}} \widehat{R}_i^{\mathrm{H}} \\ \widehat{R}_i^{\mathrm{F}} \\ \widehat{R}_i^{\mathrm{H}} - \lambda_{\mathrm{c}} \widehat{R}_i^{\mathrm{F}} \\ \widehat{R}_i^{\mathrm{H}} \end{bmatrix}. \tag{33}$$

Let $g_0 = \widetilde{\mathbf{G}}_\theta \mathbf{z}_i^0$ and $g_\lambda = \widetilde{\mathbf{G}}_\theta \mathbf{z}_i^\lambda$. Orthonormality in Eq. (32) gives

$$\begin{aligned} g_0^\top g_\lambda &= (\mathbf{z}_i^0)^\top \mathbf{z}_i^\lambda \\ &= 2\left[(\widehat{R}_i^{\mathrm{F}})^2 + (\widehat{R}_i^{\mathrm{H}})^2 - \lambda_{\mathrm{c}} \widehat{R}_i^{\mathrm{F}} \widehat{R}_i^{\mathrm{H}}\right] \\ &= \left(1 - \frac{\lambda_{\mathrm{c}}}{2}\right)\left(\widehat{R}_i^{\mathrm{F}} + \widehat{R}_i^{\mathrm{H}}\right)^2 + \left(1 + \frac{\lambda_{\mathrm{c}}}{2}\right)\left(\widehat{R}_i^{\mathrm{H}} - \widehat{R}_i^{\mathrm{F}}\right)^2 > 0. \end{aligned} \tag{34}$$

Both coefficients in the final expression are positive for $0 \leq \lambda_{\mathrm{c}} < 1$, which proves that the two update directions form an acute angle. □

Consequently, the competitive gradient is positively aligned with the gradient that jointly increases foresight- and hindsight-guided execution success.

Corollary F.9 (First-order ascent for joint success)**.** *Let* $\mathcal{J}_0$ *denote the locally linearized joint-success objective whose phase-whitened gradient is* g_0*. For sufficiently small* $\eta > 0$*,*

$$\mathcal{J}_0(\theta + \eta g_\lambda) - \mathcal{J}_0(\theta) = \eta g_0^\top g_\lambda + O(\eta^2) > 0. \tag{35}$$

Hence competitive optimization is a first-order ascent direction for joint execution success in the phase-whitened local geometry of the shared policy.

Overall, these results show that foresight–hindsight competition preserves the operational skill objective, removes additive task difficulty from within-task credit, retains a cooperative incentive for joint success, and routes the resulting positively aligned update to the correct phase tokens.