EVOGAP: WEAKNESS-GRAPH-GUIDED SKILL EVOLUTION FOR FROZEN LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Frozen agents can improve through external skills, but what they learn is constrained by the tasks they encounter. Generating additional tasks can broaden practice, but may not sufficiently target diagnosed behavioral weaknesses. Consequently, agents may accumulate more experience without systematically repairing the weaknesses that actually limit their performance. To overcome this limitation, we propose EvoGap, a weakness-graph-guided framework that turns diagnosed failures into executable practice and uses validated outcomes to guide skill evolution and subsequent practice. An updatable weakness graph records behavioral gaps and repair evidence to select the next target. Depending on a weakness's repair state, EvoGap runs breadth-oriented or depth-oriented practice to generate executable tasks with verifiable outcomes. Environment feedback informs the extraction of environment knowledge and behavioral skills. Candidate skills are admitted only after targeted validation, and their measured effects are written back to the same graph. This feedback loop connects practice selection, skill updates, and the next round of practice. Across AppWorld, BFCL-v3, and the three τ^2-Bench domains, EvoGap consistently improves task success over prior memory- and skill-based baselines with a frozen backbone, by 5.8 points on average and up to 12.5 points.

1 INTRODUCTION

In real enterprise deployments, an agent often operates for long periods with a fixed model. Whether accessed through a third-party API or hosted in-house under security and compliance constraints, the underlying model weights are typically updated infrequently. Such agents, including customer-service assistants, operations helpers, and internal enterprise tools, therefore cannot rely on frequent gradient-based updates for improvement. Instead, they must improve by accumulating and reusing external experience through practice. However, deployment continually exposes new failure modes, and enterprises expect agents to learn from these failures rather than repeat them. Consider a customer-service agent handling telecom plan changes and international roaming activation. The agent may complete most steps correctly, but end the conversation after the user casually reports that the service "seems to be working now," without verifying whether the tool return from a critical write operation confirms that the change actually took effect. When a similar but not identical situation arises later, the agent may repeat the same premature termination. For a frozen agent, addressing such recurring behavioral weaknesses requires learning from subsequent practice. This motivates skill self-evolution: how can an agent continuously identify and repair capability gaps exposed during interaction without updating its model parameters?

Existing skill self-evolution methods, e.g., EvoSkill, Trace2Skill, and SkillOpt (Alzubi et al., 2026; Ni et al., 2026; Yang et al., 2026), can discover or refine skills from execution traces, but what they harvest remains bounded by the fixed distribution of tasks they happened to practice, and behavioral weaknesses that are never triggered lack practice evidence. Recent work (e.g., SkillX's tool-level guided expansion and Skill Self-Play's skill-conditioned challenge generation (Wang et al., 2026; Huang et al., 2026)) can actively expand the task set, but more tasks do not reliably trigger the target error, nor do they necessarily land on the variations the agent has yet to master. Even when a task does hit the target error, abstract strategic advice (such as "check the state before closing") hardly specifies which fields to inspect or which authorization condition is still unmet; this kind of evidence

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.

mostly comes from the tool execution itself rather than a summary verdict, which is one motivation behind dual-track proposals that separate environment knowledge from execution strategy.

Making practice truly directed, executable, and trustworthy is not easy. First, the same task can fail for different reasons—the repair was never started, a parameter was changed incorrectly, a permission was not met, or the agent closed too early—yet a single pass/fail outcome treats all of them alike. Second, superficially different tasks do not necessarily share the same target error mechanism: a new task may fail to reproduce the triggering condition of the original error, may still be passed by guessing despite the missing key behavior, or may lose comparability because its initial state and tool-chain setting diverge from the original task. Third, a success in one practice attempt may be sampling noise rather than a real repair; abstract strategic advice tends to be vacuous, knowledge fabricated from the environment may be merely plausible detail, and inappropriate or excessive knowledge injection may in turn disturb capabilities the agent already had.

We therefore propose EvoGap, a weakness-graph-guided framework that turns diagnosed failures into executable practice while using validated results to update both the agent's skills and the direction of the next round of practice. For the first challenge, EvoGap represents recurring failures as weakness nodes: each node records a recognizable trigger condition, an observable error signature, and dynamic "mastered / not-yet-mastered" evidence. This lets different failure causes map to different nodes instead of collapsing into one pass/fail signal. For the second challenge, each node's repair state tells the weakness graph which of two practice modes to run: breadth-oriented practice expands along triggering variations to consolidate a repair that already works on some cases, while depth-oriented practice targets the decision point in the failure trajectory of a persistently hard weakness. The resulting tasks reproduce the original trigger condition and keep the initial state and tool-chain setting comparable to the source case, so environment feedback shows whether the targeted behavior actually occurred—rather than the task being passed by guessing—and which operational constraints remain unmet. For the third challenge, EvoGap distills environment-grounded knowledge and behavioral strategy from these practices at an appropriate granularity; candidate experience is admitted only if it shows a net improvement on the targeted practice without regressions, unseen variations and unrelated tasks are audited separately, and the measured effects are written back to the weakness graph to drive the next round of selection.

We evaluate this framework on three long-horizon, user-interactive benchmarks—AppWorld (Trivedi et al., 2024), BFCL-v3 (Patil et al., 2025), and τ^2-Bench (Barres et al., 2026): holding the agent, environment feedback, generator, skill extractor, admission gate, and initial skill library fixed while changing only what gets practiced, we compare weakness-graph-guided selection against a fixed task pool, random weakness selection, failure-frequency priority, and tool-level expansion, and we ablate the contributions of environment feedback versus strategy skills, whether the admission gate matters, and whether hierarchical experience retrieval is necessary. Core metrics include the average success rate on frozen held-out tasks, the disappearance rate of target error signatures, the generalization rate on unseen facets, the consistent-resolution rate on persistently failing hard tasks, and the regression rate on original tasks after admission (HARM). Main results are reported in Table 2.

This paper makes three contributions. (1) Weakness-driven practice control: we formalize skill self-evolution for frozen agents as the problem of selecting what to practice next, and cast prior selection heuristics—random and failure-frequency priority—as special cases of a single weakness-conditioned selection score; this unified view casts repairing a weakness and probing a persistently failing one as two modes of the same decision problem. (2) EvoGap: we introduce a weakness-graph-guided framework that turns diagnosed failures into executable, verifiable practice and writes validated results back to the same graph to guide skill updates and the next round of practice. (3) Extensive evaluation: we evaluate EvoGap on AppWorld, BFCL-v3, and τ^2-Bench, comparing weakness-guided practice selection against memory- and skill-based baselines; with a frozen Qwen3-Coder-Plus backbone, EvoGap improves task success over these baselines by 5.8 pp on average across the five evaluation domains ($+12.5$ pp on Telecom and $+10.6$ pp on BFCL-v3).

2 Problem Formulation

We study a single question: *for a frozen agent, what should it practice next so that its external skills keep narrowing the capability gaps that practice has exposed?* A frozen agent cannot be improved

by gradient updates; its only mutable state is an *external skill library*—retrievable experience entries injected into its context at inference time. Skill self-evolution builds up and edits this library over rounds of *practice*, and, because every other part of the loop can be held fixed, its outcome turns on a single decision: at each round, which behavioral weakness to target and how to practice it. We formalize this decision as a *weakness-conditioned practice policy* and take it as the sole object of study; the rest of this section makes it precise. Extensions beyond this problem—acquiring knowledge from external resources, or transferring across domains—are deferred to **??**.

Setup. Let π_θ be the agent with parameters frozen at $\theta = \theta_0$ throughout ($\Delta\theta = 0$); its only mutable state is a skill library $\mathcal{S}$, a set of retrievable entries. On a task q in an executable environment E, the agent retrieves a subset $R(\mathcal{S}, q) \subseteq \mathcal{S}$, injects it into context, and rolls out a trajectory $\tau \sim \pi_\theta(\cdot \mid E, q, R(\mathcal{S}, q))$, which the environment scores with a verifier $v(\tau, q, E) \in \{0, 1\}$. Performance on a task set $\mathcal{D}$ is the expected success rate

$$\mathrm{Perf}(\mathcal{S}; \mathcal{D}) = \mathbb{E}_{(E,q)\sim\mathcal{D}}\, \mathbb{E}_\tau \big[\, v(\tau, q, E) \,\big]. \tag{1}$$

Because θ is frozen, self-evolution is a sequence of library updates $\mathcal{S}_0 \to \mathcal{S}_1 \to \cdots \to \mathcal{S}_T$ (with $\theta_t = \theta_0$ for all t) that starts from a seed $\mathcal{D}_{\text{seed}}$ and an initial library $\mathcal{S}_0$ and aims to maximize performance on a frozen, held-out test set $\mathcal{D}_{\text{test}}$ never touched during evolution.

Weakness. A behavioral weakness is a machine-checkable pair $w = (c_w, \sigma_w)$ with:

- **Trigger** $c_w(E, q) \in \{0, 1\}$—a predicate over the environment/task state for *when* the error may occur;
- **Error signature** $\sigma_w(\tau) \in \{0, 1\}$—a predicate over trajectories detecting the erroneous *behavior*;
- **Facets** $\mathcal{F}_w$—variation dimensions (trigger strength, number of distractors, user behavior);
- **Mastery** $m_w \in [0, 1]$—a repair estimate from train-side probes, not from $\mathcal{D}_{\text{test}}$.

A run exhibits w iff $c_w(E, q) = 1 \wedge \sigma_w(\tau) = 1$; a failed task alone is *not* a weakness, since without a detectable (c_w, σ_w) failures cannot be attributed, aggregated, or selected. Weaknesses form a graph $\mathcal{G}_t = (\mathcal{W}_t, \mathcal{E}_t)$ whose edges encode prerequisite relations (master A before practicing B).

Practice policy and objective. A practice policy ρ reads the current graph and returns a target and a mode, $(w_t, k_t) = \rho(\mathcal{G}_t)$ with $k_t \in \{\textsc{breadth}, \textsc{depth}\}$, where BREADTH consolidates a repair that already works on some facets and DEPTH probes a weakness no intervention has yet resolved. A generator instantiates an executable, isolated task family $\mathcal{D}_t^{\text{prac}} = \mathrm{Gen}(w_t, k_t)$ with $\mathcal{D}_t^{\text{prac}} \cap \mathcal{D}_{\text{test}} = \varnothing$, each item carrying its own verifier; an extractor Φ turns the resulting trajectories and environment feedback into candidate skills $\Delta\mathcal{S}_t$; an admission gate $\mathcal{A}$ keeps them only when accepted ($\mathcal{S}_{t+1} = \mathcal{S}_t \cup \Delta\mathcal{S}_t$, otherwise $\mathcal{S}_{t+1} = \mathcal{S}_t$); and the measured evidence is written back, $\mathcal{G}_{t+1} = U(\mathcal{G}_t, \text{evidence}_t)$, updating m_w and the remaining facets and thereby conditioning the next call to ρ. Fixing everything except ρ—the agent π_θ, generator Gen, extractor Φ, gate $\mathcal{A}$, retrieval R, seed, and initial library $\mathcal{S}_0$—and writing the harm of an evolved library as the audit-set regression $\mathrm{HARM}(\mathcal{S}_T) = \Pr_{(E,q)\sim\mathcal{C}}[\, v_{\mathcal{S}_0}(E, q) = 1 \wedge v_{\mathcal{S}_T}(E, q) = 0 \,]$, the problem is to select the practice policy that maximizes frozen held-out success without regressing what the agent already knew:

$$\rho^\star = \arg\max_\rho \ \mathrm{Perf}(\mathcal{S}_T; \mathcal{D}_{\text{test}}) \quad \text{s.t. } \ \theta_t = \theta_0, \ \ \mathcal{D}_t^{\text{prac}} \cap \mathcal{D}_{\text{test}} = \varnothing, \ \ \mathrm{HARM}(\mathcal{S}_T) \le \varepsilon. \tag{2}$$

This isolates *what to practice* as the only lever and unifies prior selection signals as special cases: a fixed task pool never expands, random selection ignores m_w and $\mathcal{F}_w$, failure-frequency priority ranks by observed failure counts alone, and tool-level expansion conditions on tool usage rather than on (c_w, σ_w).

Scope. This paper solves the single problem in Equation (2): selecting weakness-conditioned practice for a frozen agent under fixed machinery. Related but distinct problems—compiling knowledge from external documents into skills, transferring skills across unseen domains, and jointly evolving parameters and skills—are treated as extensions and discussed in **??**, not as competing objectives here.

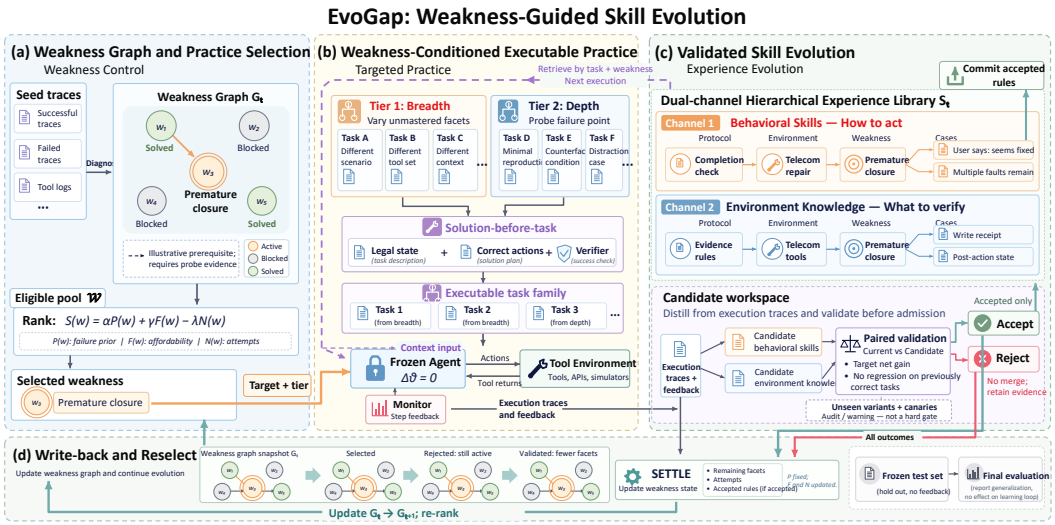

Figure 1: Overview of EvoGap. An updatable weakness graph sits at the control center: it selects the next practice target, routes it to breadth-oriented consolidation or depth-oriented breakthrough by repair state, and writes validated repair evidence back to condition the next selection. Each round runs the closed loop SELECT–GENERATE–RUN–DIAGNOSE–PROPOSE–ACCEPT–SETTLE, with the agent, environment, generator, extractor, and admission gate held fixed.

3 EvoGap Overview

Framework. We propose EvoGap, a weakness-graph-guided framework that turns diagnosed failures into executable practice and uses validated outcomes to update both the agent's skills and the direction of the next round. Its core idea is to recast skill self-evolution from "optimizing skill text on a fixed task set" into *a practice policy over a weakness graph*, whose single decision variable is *what to practice next*: a selection score with a few calibrated coefficients aggregates the graph's scattered diagnostic signals—failure prevalence, remaining facets, and trials—into one sortable scalar that picks the target, and the target's repair state (its *tier*) sets whether practice proceeds by *breadth-oriented consolidation* or *depth-oriented breakthrough*. Practice is executable and automatically checkable, and only experience validated as effective and non-regressing enters the skill library, with its measured effect written back to condition the next selection (Figure 1).

4 The Design of EvoGap

EvoGap places an updatable weakness graph at the control center of practice: it selects the next practice target, routes how the target is practiced, and, after validation, writes repair evidence back so as to condition the next selection. As shown in Figure 1, a round of practice proceeds through a closed loop—SELECT, GENERATE, RUN, DIAGNOSE, PROPOSE, ACCEPT, SETTLE—in which the same weakness ID, facet ID, task family, evidence, and skill version remain traceable end to end.

The next three subsections develop the core components in turn: Section 4.1 (weakness graph and selection), Section 4.2 (weakness-conditioned executable practice), and Section 4.3 (validated skill evolution).

4.1 Weakness Graph and Practice Selection

Weakness-graph state. Each weakness node records (c_w, σ_w), its facets, a *tier* (its repair state: tier 1 for a weakness whose repair already works on some facets, tier 2 for one that no intervention has yet resolved), a failure-prevalence prior, the remaining facets, the trials, and the accepted experience; derived quantities such as *mastery*, residual error, and loop state are reconstructed from the two-arm run logs rather than stored as a separate source of truth. A failed task alone is not a node:

Table 1: Prior selection heuristics as special cases of the selection score $S(w) = \alpha P(w) + \gamma F(w) - \lambda N(w)$, obtained by fixing the coefficient vector $(\alpha, \gamma, \lambda)$.

Selection heuristic	$(\alpha, \gamma, \lambda)$	Behavior
Random	$(0, 0, 0)$	uniform over eligible weaknesses
Failure-frequency / weakest-node	$(1, 0, 0)$	rank by failure prevalence only
Cost-aware value	$(\alpha, \gamma, 0)$	benefit over remaining cost, no stop-loss
EvoGap (full)	calibrated	benefit, cost, and stop-loss

without a machine-checkable trigger and error signature, a failure cannot be attributed, aggregated, or selected.

Selection score $S(w)$. The core of this module is a selection score that fuses the graph's diagnostic signals into a single sortable scalar. Each weakness contributes three signals rescaled to $[0, 1]$: a failure prevalence $P(w)$—the train-side fraction of failures, with the tier deciding which failures to count (tier 1 counts base-flaky failures, tier 2 counts base-always-fail cases); an affordability $F(w) = 1/(\mathrm{RemainingFacets}(w) \times \mathrm{TrialsPerFacet})$, higher when a weakness is cheaper to finish; and a sunk effort $N(w)$, how many rounds have already been spent. The score is their calibrated linear fusion

$$S(w) = \alpha\, P(w) + \gamma\, F(w) - \lambda\, N(w), \tag{3}$$

where P and F favor weaknesses that are at once worth repairing and cheap to finish, while the $-\lambda N(w)$ term is an exploration/stop-loss signal that discourages over-investing in a weakness that resists repair. The coefficients $(\alpha, \gamma, \lambda)$ are fixed by frozen calibration on the initial split rather than tuned per benchmark (default $\alpha{=}\gamma{=}\lambda{=}\frac{1}{3}$), and the discrete states (SOLVED / ACTIVE / BLOCKED) are handled by a state machine outside the score. Crucially, this linear form makes prior selection heuristics *literal* special cases obtained by fixing the coefficient vector: $(0, 0, 0)$ recovers random selection, $(1, 0, 0)$ recovers failure-frequency—equivalently weakest-node—priority, and $(\alpha, \gamma, 0)$ recovers cost-aware value without stop-loss (Table 1). Selection thus reduces to ranking the weakness graph by $S(w)$; once a weakness is chosen, its tier fixes the practice mode: tier 1 goes breadth, tier 2 goes depth. We take the benefit–cost part (α, γ) as the main claim and ablate the exploration term λ separately in Section 5.

Prerequisite edges. The ranking above operates within an eligible pool, and which weaknesses enter that pool—which to repair first—is decided by prerequisite edges. The graph keeps a single kind of edge $A \to B$ ("master A before practicing B is meaningful"), which keeps B out of the pool until A is done and unlocks B afterward. Such an edge is added only when probe evidence shows that B cannot be repaired because A is not yet mastered; otherwise nodes are selected in parallel as isolated nodes.

Write-back contract. Each SETTLE atomically writes back only three authoritative fields: `remaining_facets` and `attempts` (incremented), which re-enter the next score through F and N, and `accepted_rules` (with provenance). The failure-prevalence prior P stays static, so accumulated evidence updates the affordability and sunk-effort signals without rewriting what a weakness is worth. This closes the select–update–reselect loop.

4.2 WEAKNESS-CONDITIONED EXECUTABLE PRACTICE

Solution-before-task generation. Given the target, the generator first materializes a legal initial state, the target mechanism, the correct actions, and an automatic verifier, and only then generates the persona, user script, and surface variations—all conditioned on the target weakness and its mode. In this way, superficially different tasks share the same target error mechanism and cannot be passed while the key behavior is omitted, while the initial state, tool chain, and verifier stay comparable to the source case. An admission gate checks that each task is executable, that its scoring is consistent, that a reachable gold exists, and that it is isolated from the official test set. The repair mode follows the tier: tier-1 breadth consolidation expands along triggering variations (trigger strength, number of candidates, distractors, user behavior) to make an already-working repair stable, whereas tier-2 depth breakthrough locates the decision point in the failure trajectory and customizes a few probes to test whether a different intervention can turn a persistently hard case correct.

Two-arm execution and diagnosis. A task family is executed in the tool environment along two arms—a base control and the candidate skill. The verifier returns pass or fail; a monitor supplies step-level evidence and missing items from the tool calls, returns, and explicit user statements visible to the agent, without accessing the official test's hidden initial state or answers. Diagnosis checks whether the target error signature disappears and whether the remaining facets decrease, turning "solved in one shot / still flaky / unrepairable" into checkable facts. Here environment feedback serves both as execution support and as the source of evidence for later distillation; the two roles are ablated separately in our experiments.

4.3 Validated Skill Evolution

Dual-channel distillation and admission. Practice yields two channels of product: environment knowledge answers what holds in the current environment and which return confirms success, while behavioral skill answers how the agent should act, continue, or close under the given trigger. Both are organized hierarchically across protocol, environment, weakness, and case, and retrieval selects a granularity by task and weakness while capping the injection budget. A candidate first passes checks on factual source, schema, leakage red lines, and tool consistency, and is then validated against the current library through paired, multi-trial runs on self-generated target practice surfaces. The main gate is a net improvement on the target surfaces without regression on tasks that were already correct; held-out variants and unrelated canaries are audited separately as early warnings rather than as hard gates. We note that the environment/strategy dual channel is not itself introduced by this work; the emphasis here is not the naming but how weakness-directed practice decides what evidence to gather, what to produce, and what to practice next, and how the admission gate is tied to the training-side surfaces.

Write-back and closure. Only after passing the main gate is the official skill version advanced; otherwise the proposal and its evidence are kept but the merge is rejected. The measured repair evidence and remaining facets are then written back to the same weakness node (the three-field contract of Section 4.1), and the selector is called to begin the next round. In this scheme the official test set is used only for final evaluation and never feeds proposals or selection.

5 Experiments

Our experiments answered two questions: whether EvoGap's weakness-directed skill evolution improves a frozen agent over strong memory- and skill-based baselines (Section 5.2), and whether each component contributes to that improvement—the weakness-graph selection score (Section 5.3), the two distillation channels and the two practice modes (Section 5.4 and Section 5.5), and the closed loop (Section 5.6).

5.1 Experimental setup

Benchmarks. We used five long-horizon, user-interactive benchmarks: BFCL-v3, AppWorld, and the Retail, Airline, and Telecom domains of τ^2-Bench. They span multi-turn tool use (BFCL-v3), interactive application control (AppWorld), and dual-control customer-service dialogue with a simulated user (τ^2-Bench), so a method must handle both tool grounding and multi-step interaction. The ablations focused on τ^2-Telecom, the hardest domain for the frozen base (the lowest no-memory success), which leaves the most room for weakness-directed repair.

Backbones and transfer. Every method ran under two frozen backbones, a stronger Qwen3-Coder-Plus and a weaker Qwen-Flash (Cao et al., 2026), with no gradient update to either. The skill library was built once with the stronger backbone and injected into both, testing the strong-to-weak transfer setting.

Baselines. We compared against a no-memory agent (the frozen backbone alone) and four external memory- and skill-based methods that share the same frozen-backbone, prompt-injection interface: A-Mem (Xu et al., 2025), ExpeL (Zhao et al., 2024), AWM (Wang et al., 2025), and SkillX (Wang et al., 2026). All methods used identical backbones, tools, task splits, and injection budget, so differences reflect what is stored and selected rather than a change to the agent or the environment.

Table 2: Main results: task success rate (%, Avg@4 over four trials) across five benchmarks. The shaded row is our full method; dashes mark results not yet available. Best per model block is in **bold**.

Model	Method	BFCL-v3	AppWorld	τ^2-Bench		
				Retail	Airline	Telecom
Qwen3-Coder-Plus	No Memory	35.3	55.4	65.5	48.8	50.0
	A-Mem	48.7	50.6	65.0	**56.3**	50.0
	ExpeL	48.7	61.3	72.5	50.0	50.0
	AWM	46.0	56.0	75.0	50.0	45.0
	SkillX	48.0	59.5	75.0	**56.3**	60.0
	EvoGap (Ours)	**59.3**	**63.1**	**79.3**	**56.3**	**72.5**
Qwen-Flash	No Memory	50.7	25.0	52.5	21.3	12.5
	A-Mem	49.3	22.0	45.0	33.8	25.0
	ExpeL	50.0	26.8	52.5	25.0	26.7
	AWM	49.3	28.0	42.5	36.3	15.0
	SkillX	54.0	23.2	52.5	26.3	17.5
	EvoGap (Ours)	**61.3**	**29.8**	52.5	**37.5**	**35.0**

Metric and protocol. The primary metric was task success rate (SR), the fraction of tasks whose verifier returned full reward, reported as Avg@4 over four trials on the main comparison. Retrieval injected skills up to a fixed budget for every method, bounding the inference-time prompt overhead. The official test set was used only for final evaluation and never fed practice generation, diagnosis, or selection; the closed-loop analysis additionally tracked the two diagnostic signals introduced earlier: disappearance of the target error signature and reduction of remaining facets.

5.2 Overall comparison

Table 2 reports task success rate across the five benchmarks under both backbones.

EvoGap was the best or tied-best method in every column of Table 2. Under Qwen3-Coder-Plus it reached 72.5% on τ^2-Telecom, 12.5 points above the strongest baseline (SkillX, 60.0%) and 22.5 above the no-memory agent, with a further $+10.6$ on BFCL-v3 over the best baseline and smaller margins on AppWorld and τ^2-Retail. The same library transferred to the weaker Qwen-Flash: EvoGap led on four of the five benchmarks and matched the field on τ^2-Retail, where every method sat near the base. Two patterns stood out. First, the largest gains appeared where the frozen base was weakest—Telecom and Airline—consistent with a method that locates and repairs specific weaknesses rather than adding generic hints. Second, several baselines helped on some benchmarks but regressed on others (for example, AWM fell below no-memory on Coder-Plus Telecom, 45.0 vs 50.0, and A-Mem on AppWorld, 50.6 vs 55.4), whereas EvoGap stayed at or above the base in every cell; we attribute this to the admission gate, which merges a skill only when it shows net improvement without regression. Weakness-directed evolution thus improved the frozen agent across benchmarks and backbones without the per-benchmark regressions seen in the baselines.

5.3 Selection policy comparison

We tested the selection step in isolation. Prior selection heuristics are literal special cases of $S(w)$ (Table 1), so we compared them under a single protocol: every policy shared the same closed loop—generation, execution, diagnosis, distillation, and the admission gate—and differed only in which weakness Select picked next. Each policy ran three rounds on the same τ^2-Telecom set, and we report the per-round candidate score together with the deployed result (the best score the gate admitted), averaged over three trials (Table 3).

The comparison separated three effects. The benefit and cost terms mattered: the three value-driven policies all secured the same first-round win ($29/40$), whereas random selection reached only $20/40$, typically spending its round on lower-value or costlier weaknesses. Prevalence alone then plateaued: failure-frequency selection ranks by the static prevalence prior, returned to the same already-repaired

Table 3: Selection-policy comparison on τ^2-Telecom (40 tasks, Qwen3-Coder-Plus, 3-trial average). All policies share the same closed loop and differ only in the SELECT step; each is a special case of $S(w) = \alpha P(w) + \gamma F(w) - \lambda N(w)$ obtained by fixing $(\alpha, \gamma, \lambda)$. Columns R1–R3 are per-round candidate scores; *Deployed* is the best score the admission gate admits. Best in **bold**.

Policy	$(\alpha, \gamma, \lambda)$	R1	R2	R3	Deployed
Base (no memory)	—	—	—	—	16 (40.0%)
Random	$(0, 0, 0)$	20	20	17	20 (50.0%)
Failure-frequency / weakest-node	$(1, 0, 0)$	29	28	29	29 (72.5%)
Cost-aware value	$(\alpha, \gamma, 0)$	29	29	23	29 (72.5%)
EvoGap (full)	calibrated	29	24	31	**31 (77.5%)**

Table 4: Channel ablation on τ^2-Telecom (40 tasks, Qwen3-Coder-Plus, single trial). Environment knowledge and behavioral skill each improve over the base, and combining them is best.

Behavioral skill	w/o env knowledge	w/ env knowledge
No skill (base)	16 (40.0%)	26 (65.0%)
With skill	25 (62.5%)	**29 (72.5%)**

weakness, and stayed near 29 without further progress. The stop-loss term earned its place: cost-aware selection lacks $-\lambda N(w)$, kept investing in a resistant weakness, and its third-round candidate regressed to $23/40$; the gate kept this out of the library, so it deployed at 29. EvoGap instead redirected effort after the same non-improving round-2 candidate, and its third round was admitted at $31/40$, the best deployed result. The margin over the two informed baselines is small ($+2$), and additional trials would be needed to sharpen it, but the ordering—random $<$ prevalence-only $\approx$ cost-aware $<$ full—matches the role each term is designed to play.

5.4 Ablation: environment knowledge vs. behavioral skill

To isolate the contribution of the two distillation channels (Section 4.3), we crossed environment knowledge (on/off) with behavioral skill (on/off) on τ^2-Telecom (40 tasks, Qwen3-Coder-Plus), holding the skill source fixed across the skill-on arms. Table 4 reports the four arms. Both channels improved over the no-memory base: environment knowledge added 10 and 4 tasks (without and with skill), and behavioral skill added 9 and 3 tasks (without and with environment knowledge). Together they reached 29 of 40, the best of the four arms, indicating that the gains are complementary rather than redundant. Their complementarity reflects a division of labor. Behavioral skill prescribes the decision—what to do, and when it is safe to close a case—but does not on its own repair the intermediate process steps, such as checking that the tool return of a key write operation confirms the change took effect (the premature-closing failure from the introduction). Environment knowledge supplies this process-step grounding, so it continues to help even after a skill is in place.

5.5 Practice generation: breadth and depth

The loop generates its own practice in two modes, matched to the tier of the targeted weakness (Section 4.3): a tier-1 *breadth* mode that spreads across many facets, and a tier-2 *depth* mode that concentrates on one weakness. We evaluated each after a single closed-loop iteration on its weakness-specific held-out tasks, over three trials (Table 5). Both modes improved over the base: breadth lifted the tier-1 set from $18/39$ to $26/39$, and depth lifted the tier-2 set from $3/27$ to $14/27$. The tier-2 set started much lower, consistent with these being harder weaknesses that broad coverage alone did not reach, and it showed the larger gain.

5.6 Closed-loop iteration

We traced three rounds of the closed loop (Section 4.3) on the same τ^2-Telecom set (Table 6). Each round is driven by the weakness graph: it selects a target, practice yields a candidate skill,

Table 5: Practice-generation ablation on τ^2-Telecom weakness sets (Qwen3-Coder-Plus, 3 trials). Each mode is one closed-loop iteration: breadth targets tier-1 weaknesses (broad coverage across facets), depth targets a single tier-2 weakness (focused practice).

Generation mode	Target weaknesses	Base (pass)	After generation
Breadth (tier-1)	13 tasks $\times$ 3	18/39 (46.2%)	**26/39 (66.7%)**
Depth (tier-2)	9 tasks $\times$ 3	3/27 (11.1%)	**14/27 (51.9%)**

Table 6: Closed-loop iteration on τ^2-Telecom (40 tasks, Qwen3-Coder-Plus, 3-trial average). Each round is driven by the weakness graph. The round-2 candidate is not admitted because it does not improve on round 1; the write-back evidence guides round 3 to recover and surpass it.

Round	Candidate skill	pass/40	Admitted
—	Base (no memory)	16 (40.0%)	—
1	First admitted skill	29 (72.5%)	yes
2	Auto-derived candidate (drops an action)	24 (60.0%)	no
3	Repaired candidate (action restored)	**31 (77.5%)**	yes

the admission gate accepts or rejects it, and the measured evidence is written back to condition the next round. From the no-memory base ($16/40$), the first round produced a candidate that the gate admitted at $29/40$. In the second round the graph advanced to the next target, and the proposed candidate—its structure derived automatically from a single environment source—scored $24/40$; because this was not a net improvement over the round-1 library, the gate rejected it and the node stayed active. The diagnosis written back to that node traced the shortfall to a user-side action (a device reboot) that the automatic derivation had dropped and that the traces confirmed was needed; guided by this evidence, the third round proposed a candidate that restored the action and was admitted at $31/40$, above the round-1 29. Across the three rounds the gate keeps a non-improving candidate out of the library, and the write-back-and-reselect loop turns the diagnosed cause into a lasting fix rather than a one-off patch.

6 Related Work

Experiential memory for frozen agents. A-Mem, ExpeL, and AWM (Xu et al., 2025; Zhao et al., 2024; Wang et al., 2025) let a fixed-weight agent improve by storing past trajectories, reflections, or induced workflows and injecting them at inference. This memory grows passively as tasks arrive, without diagnosing which behavioral weakness drives the failure or checking that a new entry helps without regressing other tasks. EvoGap instead makes the target of repair an explicit decision, and admits a skill only when it improves the targeted surface without regression.

7 Conclusion

We have presented EvoGap, a framework that lets a frozen agent keep closing the capability gaps its own practice exposes. By recording recurring failures as weakness nodes and reading their repair state, EvoGap has recast skill self-evolution as an explicit decision about what to practice next, and has shown that random and failure-frequency heuristics are special cases of a single weakness-conditioned selection score. Around this score we have built a closed loop that generates practice reproducing the target failure, distills environment-grounded and behavioral experience from it, and admits a skill only when it improves the targeted surface without regressing others—writing every validated result back to the same graph to steer the next round. Across AppWorld, BFCL-v3, and τ^2-Bench, this loop has improved task success over memory- and skill-based baselines. EvoGap thus points toward frozen agents maintained not by retraining but by deciding, one weakness at a time, what to practice next.

AI USE STATEMENT

We used generative AI tools to assist with writing, including language polishing and grammar checking.

REPRODUCIBILITY STATEMENT

All models, datasets, and baseline methods used in this work are publicly available. Detailed descriptions of experimental settings, hyperparameter configurations, and training procedures are provided in the Experiments section and further elaborated in the Appendix. To ensure full reproducibility, we will also release the complete implementation and scripts for all experiments.

REFERENCES

Salaheddin Alzubi, Noah Provenzano, Christian Alexander Calvo, Jaydon Bingham, Weiyuan Chen, Himanshu Tyagi, Dastin Huang, and Tu Vu. Evoskill: Automated skill discovery for agents. In *First Workshop on Agent Skills*, 2026. URL https://openreview.net/forum?id=i4b4As7RFX.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik R Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. In *Forty-third International Conference on Machine Learning*, 2026.

Ruisheng Cao, Mouxiang Chen, Jiawei Chen, Zeyu Cui, Yunlong Feng, Binyuan Hui, Yuheng Jing, Kaixin Li, Mingze Li, Junyang Lin, Zeyao Ma, Kashun Shum, Xuwu Wang, Jinxi Wei, Jiaxi Yang, Jiajun Zhang, Lei Zhang, Zongmeng Zhang, Wenting Zhao, and Fan Zhou. Qwen3-coder-next technical report. *CoRR*, abs/2603.00729, 2026. doi: 10.48550/ARXIV.2603.00729.

Siyuan Huang, Pengyu Cheng, Haotian Liu, Tao Chen, Yihao Liu, Jingwei Ni, Shijie Zhou, Ziyi Yang, Gangwei Jiang, Mengyu Zhou, Yu Cheng, Xiaoxi Jiang, and Guanjun Jiang. Skill self-play: Pushing the frontier of llm capability with co-evolving skills, 2026.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. Trace2skill: Distill trajectory-local lessons into transferable agent skills, 2026.

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.850.

Chenxi Wang, Zhuoyun Yu, Xin Xie, Wuguannan YAO, Runnan Fang, Shuofei Qiao, Kexin Cao, Guozhou Zheng, Xiang Qi, Peng Zhang, and Shumin Deng. Skillx: Automatically constructing skill knowledge bases for agents. In *3rd AI for Math Workshop: Toward Self-Evolving Scientific Agents*, 2026.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Forty-second International Conference on Machine Learning (ICML)*, 2025.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, et al. Skillopt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: LLM agents are experiential learners. In *Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI)*, pp. 19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936.

A Appendix

You may include other additional sections here.