SKILL^2: CONTINUALLY IMPROVING HOW AGENTS LEARN SKILLS FROM EXPERIENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Skills enable language model agents to turn execution experience into procedural knowledge that can support future tasks. However, existing skill generation methods largely focus on producing and refining individual skill artifacts, while the procedure used to generate these skills is typically kept fixed. As a result, improvements in generated skills do not systematically accumulate into a better process for generating future skills. We introduce SKILL^2 (Skill-Squared), which instead treats the skill generation procedure itself as an optimization target. SKILL^2 continually improves this procedure, termed the *MetaSkill*, and uses the evolved MetaSkill to induce new skills from trajectories. This allows improvements to accumulate in the skill generation procedure rather than only in individual skill artifacts. Across four benchmarks and two executor scales, SKILL^2 outperforms the strongest skill generation baseline by an absolute 5.00% on average. Controlled experiments further show that evolved MetaSkills produce better skills when given the same trajectories and retain their advantage when applied to previously unseen trajectories, while reducing the computational cost of adaptation. These results demonstrate the value of improving the procedure that generates skills, rather than optimizing each skill in isolation.

1 INTRODUCTION

Reusable skills allow language model agents to turn execution experience into procedural knowledge. Agents retain this knowledge as reflections, workflows, manuals, and skills that guide subsequent behavior (Shinn et al., 2023; Wang et al., 2024; Zhao et al., 2024; Chen et al., 2024; Wang et al., 2025; Zheng et al., 2025). Skill generation methods make this process explicit by distilling execution traces into reusable skills and, when applicable, refining them through feedback (Ni et al., 2026; Ma et al., 2026; Yang et al., 2026; Alzubi et al., 2026).

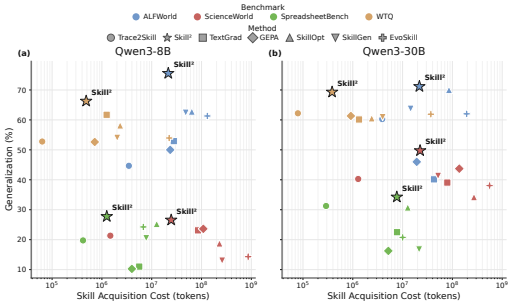


Figure 1: Skill acquisition cost and generalization. More acquisition computation does not consistently yield better generalization.

Despite their algorithmic differences, existing skill generation methods share a common design: they primarily improve the generated *skill artifact*, while the procedure that turns trajectory experience into a skill remains largely predefined. The decisions of what evidence to select, how to represent reusable behaviors, when the resulting rules should apply, and how these rules should be composed are typically specified by the method designer and reused across skill generation rounds. As a result, feedback from generating one skill may improve that particular artifact, but does not necessarily make the agent better at generating the next skill. Systematic generation errors, such as overlooking informative failures or over-generalizing conditional behaviors, can therefore recur across independently generated skills.

This limitation is also reflected in the cost generalization landscape of current skill acquisition methods. Figure 1 compares seven representative methods across four benchmarks and two executor

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.

scales. We define *skill acquisition cost* as the total number of tokens consumed by a method to obtain its final skill from trajectory experience, including induction, refinement, selection, or verification when applicable. Across methods, this cost spans several orders of magnitude, yet higher acquisition cost does not consistently yield stronger generalization.

This does not imply that additional computation is ineffective. Rather, more per-skill computation alone is insufficient to make skill acquisition progressively better. Experience is used to improve the current skill, but is not necessarily retained as knowledge about *how future skills should be generated*.

This motivates a more fundamental question: *can the skill generation procedure itself be learned and continually improved, such that experience from previous skill generation outcomes improves how future skills are generated?*

As illustrated in Figure 2, we represent the skill generation procedure as a persistent *MetaSkill*, which specifies how trajectory experience is transformed into reusable skills. SKILL^2 explicitly optimizes this MetaSkill, making the process of transforming trajectories into reusable skills itself a persistent object of learning.

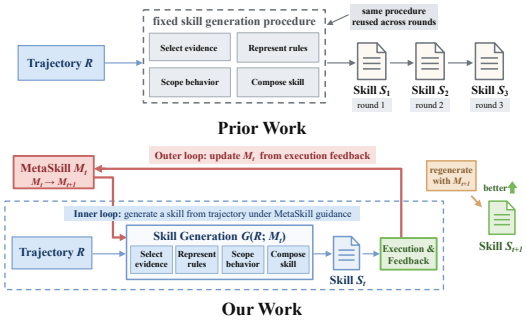


Figure 2: Fixed vs. learned skill generation. Prior work uses a predefined generation procedure, while SKILL^2 continually improves it.

Continual improvement of skill generation need not be monotonic. Successive MetaSkill updates can exhibit either positive transfer or interference with previously acquired generation behavior. For example, an update that encourages stronger abstraction may improve the current skill while erasing exceptions retained by an earlier update. Because the MetaSkill sits upstream of skill generation, such interference can propagate through regeneration and alter multiple downstream skill rules. We refer to this phenomenon as *MetaSkill interference*, an instance of the stability-plasticity tension studied in continual learning (Riemer et al., 2019). Continually improving skill generation ability therefore requires incorporating new feedback while retaining previously validated generation behavior.

We introduce SKILL^2 (Skill-Squared) to continually improve the skill generation procedure. Execution feedback diagnoses errors in how trajectories are selected, represented, scoped, or composed, and proposes corresponding MetaSkill updates. Before commit, updates are resolved against previously validated generation principles and evaluated by regenerating a fresh skill, allowing improvements to accumulate across rounds.

We evaluate SKILL^2 against eight baselines across four benchmarks and two executor scales. SKILL^2 outperforms the strongest baseline by **5.00 points on average** across all eight settings. More importantly, under fixed trajectory evidence, evolved MetaSkills outperform the initial MetaSkill by **13.09 points on average**. On previously unseen trajectories, the evolved MetaSkill further improves target utility by **23.12 points**. Finally, resolving interference between new and previously validated generation principles substantially reduces regressions across MetaSkill updates.

Our contributions are:

- **Continual optimization of the skill generation procedure.** We make an explicit skill generation procedure a persistent optimization target and assess its improvement through fresh generation from the same trajectories.

- **Continual skill generation learning under interference.** We formulate interactions between successive MetaSkill updates as a stability-plasticity problem and use previously validated induction behavior to preserve useful generation principles while incorporating new feedback.

- **Evidence that skill generation ability improves and transfers.** Across four benchmarks and two executor scales, evolved MetaSkills generate stronger skills from fixed trajectory evidence, retain this advantage on previously unseen evidence, and exhibit fewer harmful updates with interaction-aware resolution.

2 Related Work

Learning reusable skills from experience. Language model agents retain execution experience as reflections, memories, workflows, manuals, and skills (Shinn et al., 2023; Wang et al., 2024; Zhao et al., 2024; Wang et al., 2025; Chen et al., 2024; Zheng et al., 2025). Recent methods generate skills through trajectory distillation (Ni et al., 2026), verified synthesis (Ma et al., 2026), and feedback-driven refinement or discovery (Yang et al., 2026; Alzubi et al., 2026). These methods primarily improve skill artifacts under largely predefined generation procedures. Recent work also explores meta-level optimization for skill refinement (Wang et al., 2026). SKILL^2 uses execution feedback to refine a persistent MetaSkill for generating reusable skills from trajectory experience.

Learning and optimizing generation procedures. Meta learning studies how experience can improve the learning process (Finn et al., 2017), while LLM based optimization iteratively improves textual objects such as prompts, instructions, and programs through feedback (Zhou et al., 2023; Pryzant et al., 2023; Yang et al., 2023; Yuksekgonul et al., 2024; Agrawal et al., 2025; Khattab et al., 2024; Madaan et al., 2023). PromptBreeder evolves mutation prompts used to generate task prompts (Fernando et al., 2024); MCE co-evolves context engineering skills and context artifacts (Ye et al., 2026); and SkillOS learns a curator for an existing skill collection (Ouyang et al., 2026). These methods optimize prompt generation, context construction, or skill curation rather than the procedure that transforms trajectory experience into new reusable skills across successive generation rounds.

Continual learning, transfer, and interference. Making the skill generation procedure persistent introduces a stability and plasticity problem: successive updates may conflict with previously learned behavior. Continual learning addresses related interference through regularization (Kirkpatrick et al., 2017), experience replay (Lopez-Paz & Ranzato, 2017), architectural isolation (Rusu et al., 2016), and recent parameter efficient methods such as SD-LoRA (Wu et al., 2025b). These approaches primarily manage model parameter updates (Riemer et al., 2019). In our setting, model parameters remain frozen while feedback modifies textual generation principles in the MetaSkill, so a new principle can interfere with previously validated behavior in ways that parameter level mechanisms do not directly address. Continual refinement therefore requires resolving interactions among generation principles while incorporating new feedback.

3 Method

We formulate continual optimization of the skill generation procedure and present SKILL^2 through its MetaSkill representation, feedback based updating, interaction resolution, and verified commit (Figure 3).

3.1 Problem Formulation

Let R denote a pool of execution trajectories. A MetaSkill M specifies how a frozen skill generator $\mathrm{Generate}$ uses the trajectory evidence in R to produce a reusable skill,

$$S = \mathrm{Generate}(R; M). \tag{1}$$

The generated skill S guides a frozen executor in solving downstream tasks.

We assess a MetaSkill through the downstream utility of its generated skill. For an execution set $\mathcal{D}$, define

$$J_{\mathcal{D}}(M; R) = \mathrm{Util}(\mathrm{Execute}(\mathrm{Generate}(R; M); \mathcal{D}))\,, \tag{2}$$

where $\mathrm{Execute}$ denotes task execution and Util aggregates execution outcomes into a scalar score. Our objective is to learn a MetaSkill whose generated skills achieve higher utility on held out tasks.

We optimize a sequence of persistent MetaSkills $M_0, \ldots, M_T$. At round t, $S_t = \mathrm{Generate}(R; M_t)$ is executed to obtain update feedback. The trajectory pool R remains fixed, and every candidate MetaSkill generates a fresh skill from the same R, isolating changes in the skill generation procedure from changes in trajectory evidence.

We use $\mathcal{D}_{\mathrm{train}}$ for update feedback and a disjoint $\mathcal{D}_{\mathrm{val}}$ for candidate assessment. Test data are accessed only after evolution is frozen and never affect optimization or state selection.

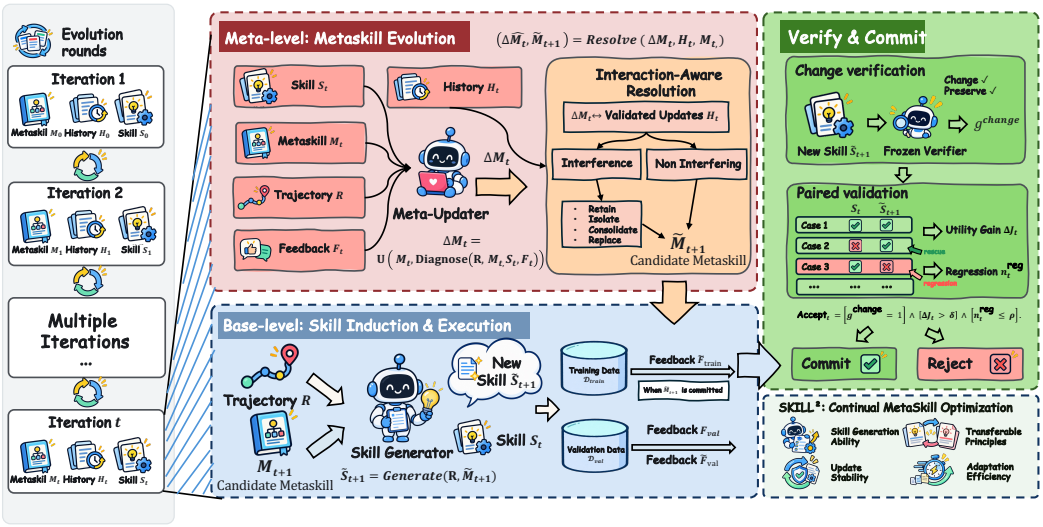


Figure 3: **Overview of SKILL^2.** One evolution round updates the persistent MetaSkill from execution feedback, regenerates a fresh skill from the fixed trajectory pool, and verifies the candidate before commit.

3.2 METASKILL: REPRESENTING THE SKILL GENERATION PROCEDURE

To make the skill generation procedure learnable and continually improvable from execution feedback, we represent it as a persistent textual MetaSkill M while keeping the skill generator itself frozen.

We organize its generation decisions into four components: SELECT chooses trajectory evidence, REPRESENT transforms it into reusable rules, SCOPE defines their applicability, and COMPOSE organizes and coordinates them in the generated skill.

This explicit representation provides localized update targets for improving the skill generation procedure from execution feedback. Further representation details and design constraints are provided in Appendix A.1.

3.3 FEEDBACK ATTRIBUTION AND METASKILL UPDATE

Because execution feedback is observed on the generated skill while the MetaSkill is the optimization target, each update first attributes the observed behavior to an upstream generation decision.

All runs start from the same hand authored M_0, with initial skill $S_0 = \text{Generate}(R; M_0)$ and validated history $H_0 = \emptyset$. At round t, let F_t denote the cached outcomes and execution traces obtained by running S_t on $\mathcal{D}_{\text{train}}$.

We form an evidence grounded attribution hypothesis by jointly examining the trajectory evidence, the current MetaSkill, the generated skill, and its execution feedback:

$$P_t = \text{Diagnose}(R, M_t, S_t, F_t). \tag{3}$$

The diagnosis compares the execution discrepancy in F_t with the behavior prescribed by S_t, the trajectory evidence in R, and the generation decisions in M_t to identify an evidence grounded candidate decision.

The Meta Updater converts this diagnosis into a bounded edit:

$$\Delta M_t = U(M_t, P_t). \tag{4}$$

Each proposal specifies the target decision, its current and proposed values, the supporting diagnosis, the intended skill change, and the behavior to preserve.

A concrete COMPOSE update. In a recorded ALFWorld cooling episode, the agent reached the refrigerator only at the 50 step limit, leaving cooling and final placement unfinished. The incumbent skill already contained search, cooling, and placement rules. The diagnosis attributed the failure to how these rules were coordinated and proposed enabling mutually exclusive execution stages in COMPOSE. The corresponding regenerated skill introduced a router that selects the first applicable observable stage and uses only compatible rules while retaining the existing rules within each stage. The complete recorded update is provided in Appendix F.1.

3.4 INTERACTION AWARE RESOLUTION

A proposed MetaSkill update may correct the current failure while altering generation behavior validated in earlier rounds. Because final validation cannot separate a useful correction from a conflicting change, we resolve each proposal against previously validated generation principles before candidate construction. Section 3.5 then determines whether the resolved candidate is committed.

The current MetaSkill M_t stores the generation principles in use, while the validated history H_t records previously accepted principles together with their applicability conditions, protected behavior, and paired validation evidence. Only the resolver receives H_t; the proposal function U and the executor do not. Thus, H_t is used specifically to determine how a new update interacts with generation behavior validated in earlier rounds.

If resolution produces a candidate, it returns a resolved update and the corresponding MetaSkill:

$$\left(\Delta\widehat{M}_t, \widetilde{M}_{t+1}\right) = \mathrm{Resolve}(M_t, \Delta M_t, H_t). \tag{5}$$

For each relevant history entry h_i, the resolver examines overlap in applicability conditions, affected generation decisions, and protected behavior, and classifies the interaction as

$$r(\Delta M_t, h_i) \in \{\textsc{Transfer}, \textsc{Non Interfering}, \textsc{Interference}\}. \tag{6}$$

TRANSFER reinforces or extends validated behavior, NON INTERFERING affects sufficiently separate conditions or behavior, and INTERFERENCE may alter behavior protected by an earlier accepted principle.

Updates exhibiting TRANSFER or NON INTERFERING can be incorporated directly. For an interfering proposal, the resolver can RETAIN an earlier principle and abstain when evidence for changing it is insufficient; ISOLATE conflicting uses through applicability conditions or negative guards; CONSOLIDATE compatible information into a more precise conditional generation principle; or REPLACE an earlier principle when new execution evidence directly contradicts it within its stated conditions.

In a recorded SCOPE proposal, reducing a search miss threshold from six to five revisited the same generation decision that had previously been validated at six. The resolver applied RETAIN, kept the validated value of six, and abstained before skill regeneration (Appendix F.2).

Abstention skips skill regeneration; otherwise, the candidate proceeds to fresh regeneration and verification in Section 3.5.

3.5 FRESH SKILL REGENERATION AND VERIFIED COMMIT

A resolved MetaSkill update is committed only if it is realized in the generated skill and improves downstream execution. Given the candidate MetaSkill $\widetilde{M}_{t+1}$, we regenerate a fresh skill from the same trajectory evidence:

$$\widetilde{S}_{t+1} = \mathrm{Generate}(R; \widetilde{M}_{t+1}). \tag{7}$$

The incumbent skill S_t is not provided to the generator. The resulting update is evaluated through two successive effects:

$$\Delta\widehat{M}_t \;\longrightarrow\; \Delta S_t \;\longrightarrow\; \Delta J_t.$$

First, a frozen LLM verifier checks whether the regenerated skill realizes the resolved MetaSkill update while retaining the behavior declared for preservation:

$$g_t^{\mathrm{change}} = \mathrm{Verify}\left(\Delta\widehat{M}_t; M_t, \widetilde{M}_{t+1}, S_t, \widetilde{S}_{t+1}\right) \in \{0, 1\}. \tag{8}$$

The verifier checks realization rather than downstream utility; failed candidates are discarded before validation execution, and the verifier remains fixed throughout evolution (Appendix A.4).

Second, candidates that pass change verification are executed on the fixed validation set $\mathcal{D}_{\text{val}}$ under the same conditions as the incumbent. Using the cached incumbent feedback F_{val} and candidate feedback $\widetilde{F}_{\text{val}}$, we compute

$$\Delta J_t = \text{Util}(\widetilde{F}_{\text{val}}) - \text{Util}(F_{\text{val}}). \tag{9}$$

Aggregate utility can conceal losses on instances that the incumbent already solves. Let n_t^{reg} count validation instances that succeed under S_t but fail under $\widetilde{S}_{t+1}$. We therefore accept a candidate only if

$$\text{Accept}_t = \left[g_t^{\text{change}} = 1\right] \wedge \left[\Delta J_t > \delta\right] \wedge \left[n_t^{\text{reg}} \leq \rho\right]. \tag{10}$$

Accepted candidates replace the incumbent MetaSkill and skill, and the validated generation principle is added to H_t. Rejected candidates leave the incumbent state unchanged.

For the COMPOSE example in Section 3.3, the change check passed and paired validation improved from 6 to 8 successes, with two rescues and no regressions. The update was therefore committed (Appendix F.1).

The complete state update and stopping procedure are given in Algorithm 1 and Appendix B.

4 EXPERIMENTS

We first describe the experimental setup and then evaluate SKILL^2 across multiple benchmarks and executor scales, followed by analyses of its continual evolution behavior, transfer, stability, and adaptation efficiency.

4.1 EXPERIMENTAL SETUP

Benchmarks and models. We evaluate **SpreadsheetBench** (Ma et al., 2024), **WikiTableQuestions (WTQ)** (Pasupat & Liang, 2015), **ALFWorld** (Shridhar et al., 2021), and **ScienceWorld** (Wang et al., 2022) with **Qwen3-8B** and **Qwen3-30B** executors (Yang et al., 2025). SKILL^2 starts from the same task-neutral M_0 across all settings.

Baselines. We compare with No-Skill; one-shot LLM Skill (Yang et al., 2026); Trace2Skill (Ni et al., 2026); SkillGen (Ma et al., 2026); SkillOpt (Yang et al., 2026); EvoSkill (Alzubi et al., 2026); TextGrad (Yuksekgonul et al., 2024); and GEPA (Agrawal et al., 2025). TextGrad and GEPA provide direct skill text optimization baselines. Implementation details are in Appendix D.

Evaluation protocol. Within each benchmark and model setting, all methods share the executor, data partitions, action interface, scorer, and native evaluation metric, reported on a 0–100 scale; gains are absolute differences. Results are averaged over three independent seeds unless otherwise noted; Table 1 reports means and Appendix C.6 reports mean and standard deviation. During evolution, R remains fixed, disjoint training and validation sets support updating and candidate acceptance, and test data are accessed only after evolution is frozen. Full splits and controls are in Appendix C.

4.2 FINAL SKILL QUALITY

Consistent gains across settings. SKILL^2 achieves the highest mean performance in all eight benchmark and model settings (Table 1), exceeding the strongest competing baseline by **1.10% to 12.78%** and **5.00% on average**. The largest margin occurs on ALFWorld/Qwen3-8B (75.55 vs. 62.77). The strongest competitor varies across settings, while SKILL^2 leads in all eight comparisons.

Contribution of MetaSkill evolution. Static M_0 underperforms No-Skill in all four Qwen3-8B settings and provides only modest gains at the 30B scale. Evolving the same initialization improves performance by **8.55% to 34.19%**, averaging **16.77%** across the eight settings. This

Method	SpreadsheetBench		WTQ		ALFWorld		ScienceWorld	
	Qwen3-8B	Qwen3-30B	Qwen3-8B	Qwen3-30B	Qwen3-8B	Qwen3-30B	Qwen3-8B	Qwen3-30B
No-Skill	15.83	17.67	53.91	60.64	41.73	50.73	13.85	34.65
LLM Skill	17.33$_{+1.50}$	20.50$_{+2.83}$	51.28$_{-2.63}$	52.95$_{-7.69}$	42.58$_{+0.85}$	55.96$_{+5.23}$	12.09$_{-1.76}$	33.20$_{-1.45}$
Trace2Skill	19.75$_{+3.92}$	31.25$_{+13.58}$	52.78$_{-1.13}$	62.21$_{+1.57}$	44.65$_{+2.92}$	60.34$_{+9.61}$	21.33$_{+7.48}$	40.27$_{+5.62}$
SkillOpt	25.25$_{+9.42}$	30.75$_{+13.08}$	58.15$_{+4.24}$	60.53$_{-0.11}$	62.77$_{+21.04}$	70.07$_{+19.34}$	18.74$_{+4.89}$	34.21$_{-0.44}$
SkillGen	20.50$_{+4.67}$	16.75$_{-0.92}$	54.02$_{+0.11}$	60.93$_{+0.29}$	62.41$_{+20.68}$	63.75$_{+13.02}$	13.01$_{-0.84}$	41.31$_{+6.66}$
EvoSkill	24.25$_{+8.42}$	20.75$_{+3.08}$	53.95$_{+0.04}$	61.89$_{+1.25}$	61.31$_{+19.58}$	62.04$_{+11.31}$	14.33$_{+0.48}$	38.07$_{+3.42}$
Text optimization methods								
TextGrad	11.00$_{-4.83}$	22.50$_{+4.83}$	61.71$_{+7.80}$	60.14$_{-0.50}$	52.92$_{+11.19}$	40.15$_{-10.58}$	23.11$_{+9.26}$	39.07$_{+4.42}$
GEPA	10.25$_{-5.58}$	16.25$_{-1.42}$	52.63$_{-1.28}$	61.32$_{+0.68}$	50.00$_{+8.27}$	45.99$_{-4.74}$	23.62$_{+9.77}$	43.72$_{+9.07}$
MetaSkill control								
Static MetaSkill (M_0)	9.75$_{-6.08}$	19.50$_{+1.83}$	51.89$_{-2.02}$	60.75$_{+0.11}$	41.36$_{-0.37}$	53.89$_{+3.16}$	12.89$_{-0.96}$	36.38$_{+1.73}$
SKILL^2 (Ours)	**27.75**$_{+11.92}$	**34.25**$_{+16.58}$	**66.28**$_{+12.37}$	**69.30**$_{+8.66}$	**75.55**$_{+33.82}$	**71.17**$_{+20.44}$	**26.56**$_{+12.71}$	**49.74**$_{+15.09}$

Table 1: **Final task performance across benchmarks and executor scales.** Subscripts denote change from No-Skill (green: gain, red: loss). Static MetaSkill (M_0) uses the same hand-authored initialization without evolution. **Bold** marks the best result.

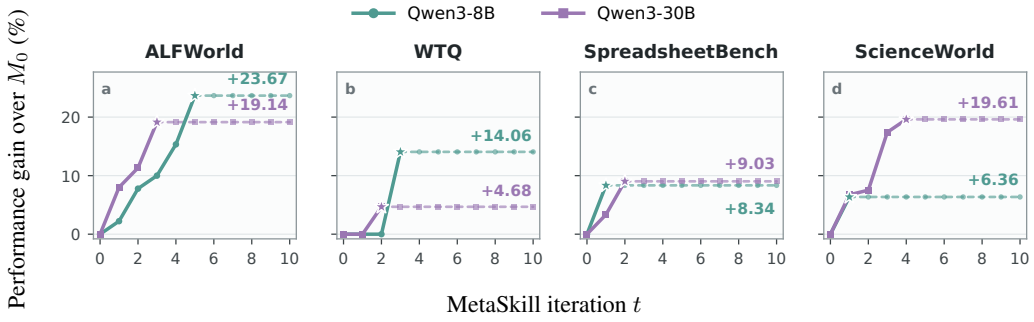


Figure 4: **Continual improvement under fixed trajectory evidence.** Mean performance gain over M_0 for fresh skills generated from each MetaSkill checkpoint. Stars mark the first checkpoint reaching the final mean; dashed segments show the subsequent plateau.

comparison demonstrates the contribution of refining generation decisions beyond the initial MetaSkill representation.

4.3 Does the Skill Generation Procedure Itself Improve?

Stronger final skills alone do not establish how skill generation improves over successive MetaSkill updates. We therefore revisit the frozen checkpoints after evolution and regenerate a fresh skill from each M_t using the same trajectory pool R. The generator, executor, and evaluation protocol remain fixed, and no previously optimized skill is provided as input. This comparison tests whether evolving MetaSkills generate better skills from unchanged trajectory evidence.

MetaSkill evolution improves generation from fixed evidence. Across all eight benchmark and model settings, later checkpoints generate skills that outperform those generated from M_0, with final gains of **4.68% to 23.60%** and **13.09% on average** (Figure 4). Because each skill is generated afresh from the same trajectory pool, these gains cannot be explained by additional trajectory evidence or direct inheritance of an optimized skill. The MetaSkill is the only systematically varied component. The results therefore provide controlled evidence that execution feedback improves the generation guidance retained in the MetaSkill, enabling it to transform the same experience into stronger skills.

Improvement extends beyond the first update. Six of the eight settings continue improving after M_1, indicating that the gains are not generally explained by a single correction to the initial MetaSkill. Repeated feedback can further refine generation decisions across successive rounds. Across settings, final performance levels are first reached between M_1 and M_5 and remain stable at later checkpoints.

This pattern supports both improvement over multiple updates and retention of the attained utility along the observed checkpoint sequence.

4.4 Does the Improved Skill Generation Procedure Transfer to New Evidence and Benchmarks?

Section 4.3 examines improved generation from fixed trajectory evidence. We now ask whether the learned guidance remains useful when the evidence changes, and how its utility varies across tasks with different reasoning requirements. Both studies use Qwen3-8B.

Transfer to unseen trajectory evidence. We evolve M_T on an ALFWorld source pool R_{src} and freeze it. Both M_0 and M_T then generate fresh skills from the same unseen R_{tgt}, with $R_{\mathrm{src}} \cap R_{\mathrm{tgt}} = \emptyset$ and no target MetaSkill update or checkpoint selection. Utility increases from **42.70** to **65.82**, a gain of **23.12%**, showing that learned generation guidance remains useful beyond the trajectories used during evolution. Full details are in Appendix C.4.

Transfer across benchmarks. We transfer a frozen WTQ learned principle for evidence grounded query planning to nine target benchmarks. Default uses M_0, while Transfer adds only this principle before fresh skill generation from the same target trajectories. Source trajectories, skills, and answers are never reused, so the difference isolates the utility of the transferred generation principle. We report this frozen comparison descriptively.

Target benchmark	No-Skill	Default	Transfer	Δ vs. Default
WikiSQL (Zhong et al., 2017)	71.0	71.0	82.0	+11.0
TabFact (Chen et al., 2020a)	80.0	80.0	88.5	+8.5
SQA-gold-history (Iyyer et al., 2017)	61.5	63.0	68.0	+5.0
TabMWP (Lu et al., 2023)	54.5	56.5	57.0	+0.5
TableBench-Exact (Wu et al., 2025a)	29.0	29.0	27.0	−2.0
FinQA-table-only (Chen et al., 2021)	3.5	6.0	5.0	−1.0
TAT-QA (Zhu et al., 2021)	49.5	50.0	50.0	0.0
HiTab (Cheng et al., 2022)	71.0	68.0	70.0	+2.0
HybridQA (Chen et al., 2020b)	58.0	56.0	60.0	+4.0
Macro average	53.11	53.28	**56.39**	+3.11

Table 2: **Transfer of a WTQ-learned generation principle to nine target benchmarks.** Scores are percentages. Δ is Transfer minus Default. Macro averages weight all target benchmarks equally.

Positive transfer across related reasoning tasks. Transfer improves **six of nine** targets, with an average gain of **3.11%** across all nine targets. The largest gains are on WikiSQL (**+11.0%**), TabFact (**+8.5%**), and SQA-gold-history (**+5.0%**). These tasks rely on selecting and comparing table evidence, consistent with the transferred query planning principle and suggesting stronger reuse when reasoning requirements align.

Transfer has an applicability scope. HiTab and HybridQA also improve, but TabMWP and TAT-QA change little, while FinQA-table-only and TableBench-Exact decrease by 1.0% and 2.0%. This pattern suggests that query planning transfers when target reasoning is compatible, but may provide limited or harmful guidance when additional requirements such as numerical derivation dominate.

4.5 Interference and Stability of Continual MetaSkill Updates

We examine whether interaction assessment identifies harmful updates and whether resolving them improves stability beyond simply retaining validated history.

Interaction assessment identifies elevated regression risk. Among **197 logged proposal events**, 19 are classified as Interference. Before resolution, **14/19 (73.7%)** produce a regression versus **55/178 (30.9%)** among the remaining proposals, a **2.38**× relative risk. Because the pooled events are not statistically independent, we treat this comparison as descriptive.

Resolution reduces damage while retaining useful corrections. For the same 19 proposals, resolution reduces regression producing cases from **14 to 5** and total regressions from **30 to 8** (**73.3%**), while rescues change only from **13 to 12**. The asymmetric change shows that resolution reduces harmful regressions without simply suppressing useful corrections.

Explicit resolution improves stability beyond history retention. We compare Greedy updates without persistent history, History without interaction resolution, and full SKILL^2 on ALFWorld/Qwen3-8B. All variants share the data protocol, initialization, models, generation and verification procedures, acceptance rule, and a fixed stream of 24 proposals (Appendix C.3).

Variant	Outcome		Search	Stability		
	Val. ↑	Test ↑	Regens. ↓	Rescues ↑	Regress. ↓	Reg./update ↓
Greedy	68/90	187/274	24	17	21	4.20
History (w/o resolution)	69/90	188/274	23	**18**	17	3.40
SKILL^2	**69/90**	**191/274**	**20**	17	**7**	**1.40**

Table 3: **Matched ablation of continual MetaSkill updates on ALFWorld/Qwen3-8B.** Rescues denote transitions from failure to success, while regressions denote transitions from success to failure.

All three variants accept five updates and reach similar validation performance. History reduces regressions from **21 to 17**; explicit resolution reduces them further to **7** while retaining **17 rescues**. Because the number of accepted updates is unchanged, the gain comes from resolving interactions rather than committing fewer updates. Full SKILL^2 also requires fewer regenerations (**20** vs. **24**), consistent with earlier filtering of incompatible proposals.

4.6 ADAPTATION EFFICIENCY

Because SKILL^2 adds diagnosis, resolution, and verification, we test whether its gains require more adaptation computation. On ALFWorld/Qwen3-8B, SKILL^2 and SkillOpt share the trajectory pool, data partitions, executor, and accounting boundary; cost excludes fixed trajectory collection and evaluation after evolution.

Stronger skills with lower adaptation cost. SKILL^2 improves test utility from **62.77 to 75.55** while reducing rollouts from **1,590 to 540** and tokens from **161.90M to 44.80M**, corresponding to **66.0%** and **72.3%** reductions. Thus, stronger final performance is achieved with lower total adaptation cost.

Avoiding repeated downstream execution. Savings come from resolution abstention before regeneration, verifier filtering before validation, cached incumbent feedback, and refreshing training feedback only after accepted updates. These mechanisms reduce repeated execution, consistent with the lower regeneration count in Section 4.5; detailed accounting is in Appendix E.

5 CONCLUSION

We introduced SKILL^2 to continually improve skill generation from execution experience. By learning a persistent MetaSkill and resolving interactions between successive updates, SKILL^2 accumulates generation principles while limiting interference with validated behavior. Experiments across four benchmarks and two executor scales demonstrate stronger final skills. Controlled analyses show that generation ability improves under fixed evidence and transfers to unseen trajectories. Cross benchmark results further suggest that transfer depends on the compatibility of reasoning requirements. Interaction resolution reduces regressions while retaining useful corrections, and matched evaluation shows lower adaptation cost.

These findings support treating skill generation itself as a capability that improves with experience. Our study fixes trajectory evidence and model parameters within each run. Future work will explore evolving evidence and the long term stability and applicability of learned generation principles.

AI USE STATEMENT

Generative AI tools were used primarily for language editing and improving the clarity of the manuscript, and as assistance for code development, including code drafting, debugging, and implementation support. The research questions, methodological design, experimental protocols, evaluation criteria, and scientific conclusions were developed and determined by the authors. All AI-assisted code and text were reviewed and verified by the authors, who take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

We provide the complete MetaSkill schema, initialization, update and verification procedures, benchmark splits, baseline protocols, statistical controls, and adaptation-cost accounting in the appendix. Main results are aggregated over three independent seeds, with variability reported where applicable.

REFERENCES

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. Gepa: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. Evoskill: Automated skill discovery for multi-agent systems. *arXiv preprint arXiv:2603.02766*, 2026.

Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. AutoManual: Constructing instruction manuals by LLM agents via interactive environmental learning. In *Advances in Neural Information Processing Systems*, 2024.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. TabFact: A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations*, 2020a. URL https://arxiv.org/abs/1909.02164.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020b. URL https://aclanthology.org/2020.findings-emnlp.91/.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. FinQA: A dataset of numerical reasoning over financial data. In *Conference on Empirical Methods in Natural Language Processing*, 2021. URL https://aclanthology.org/2021.emnlp-main.300/.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. HiTab: A hierarchical table dataset for question answering and natural language generation. In *Annual Meeting of the Association for Computational Linguistics*, 2022. URL https://aclanthology.org/2022.acl-long.78/.

Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In *International Conference on Machine Learning*, 2024.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. Search-based neural structured learning for sequential question answering. In *Annual Meeting of the Association for Computational Linguistics*, 2017. URL https://aclanthology.org/P17-1167/.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into self-improving pipelines. In *International Conference on Learning Representations*, 2024.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, 2017.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2209.14610.

Yuchen Ma, Yue Huang, Han Bao, Haomin Zhuang, Swadheen Shukla, Michel Galley, Xiangliang Zhang, and Stefan Feuerriegel. Skillgen: Verified inference-time agent skill synthesis. *arXiv preprint arXiv:2605.10999*, 2026.

Zeyao Ma, Bohan Zhang, Jing Zhang, Jifan Yu, Xiaokang Zhang, Xiaohan Zhang, Sijia Luo, Xi Wang, and Jie Tang. SpreadsheetBench: Towards challenging real world spreadsheet manipulation. In *Advances in Neural Information Processing Systems*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ac840df270ac537dd74530a15c332684-Paper-Datasets_and_Benchmarks_Track.pdf.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, 2023.

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. *arXiv preprint arXiv:2603.25158*, 2026.

Siru Ouyang, Jun Yan, Yanfei Chen, Rujun Han, Zifeng Wang, Bhavana Dalvi Mishra, Rui Meng, Chun-Liang Li, Yizhu Jiao, Kaiwen Zha, Maohao Shen, Vishy Tirumalashetty, George Lee, Jiawei Han, Tomas Pfister, and Chen-Yu Lee. Skillos: Learning skill curation for self-evolving agents. *arXiv preprint arXiv:2605.06614*, 2026.

Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. In *Annual Meeting of the Association for Computational Linguistics*, 2015.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Conference on Empirical Methods in Natural Language Processing*, 2023.

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*, 2019.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? In *Conference on Empirical Methods in Natural Language Processing*, 2022.

Zefeng Wang, Minxi Yan, Jinhe Bi, Sikuan Yan, Volker Tresp, and Yunpu Ma. Metaskill-evolve: Recursive self-improvement of LLM agents via two-timescale meta-skill evolution. *arXiv preprint arXiv:2607.05297*, 2026.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *International Conference on Machine Learning*, 2025.

Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, Xeron Du, Di Liang, Daixin Shu, Xianfu Cheng, Tianzhen Sun, Tongliang Li, Zhoujun Li, and Guanglin Niu. TableBench: A comprehensive and complex benchmark for table question answering. In *AAAI Conference on Artificial Intelligence*, 2025a. URL https://ojs.aaai.org/index.php/AAAI/article/view/34739.

Yichen Wu, Hongming Piao, Long-Kai Huang, Renzhen Wang, Wanhua Li, Hanspeter Pfister, Deyu Meng, Kede Ma, and Ying Wei. SD-LoRA: Scalable decoupled low-rank adaptation for class incremental learning. In *International Conference on Learning Representations*, 2025b.

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

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. Skillopt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026.

Haoran Ye, Xuning He, Vincent Arak, Haonan Dong, and Guojie Song. Meta context engineering via agentic skill evolution. *arXiv preprint arXiv:2601.21557*, 2026.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic "differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *AAAI Conference on Artificial Intelligence*, 2024.

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. Skillweaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017. URL https://arxiv.org/abs/1709.00103.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *International Conference on Learning Representations*, 2023.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing*, 2021. URL https://aclanthology.org/2021.acl-long.254/.

A IMPLEMENTATION DETAILS

Model configuration. The Meta-Updater, interaction-aware resolver, skill generator, frozen skill-change verifier, and task executor all use the same backbone model within each benchmark and model setting (Qwen3-8B or Qwen3-30B). All model parameters remain frozen; adaptation updates the MetaSkill and the skill it generates.

A.1 METASKILL REPRESENTATION

We represent a MetaSkill as a persistent textual representation of the skill generation procedure. To make this procedure explicitly editable, the implementation organizes its generation decisions into four components: SELECT, REPRESENT, SCOPE, and COMPOSE. These components correspond to key stages of skill generation. SELECT determines which trajectory evidence is used for generation; REPRESENT determines how selected evidence is transformed into reusable rules; SCOPE determines the applicability conditions of generated rules; and COMPOSE determines how individual rules are organized and coordinated in the generated skill. The implementation represents these decisions with typed fields and parameters; `assemble` in the saved records corresponds to COMPOSE. Appendix F provides concrete recorded updates.

Representation scope. The four components provide an operational organization of the generation decisions that are explicitly updated in our implementation. SKILL^2 does not assume that this organization is a unique or exhaustive decomposition of skill generation. The method only requires that a proposed update can be associated with an explicit generation decision whose effect on the generated skill can subsequently be evaluated.

To keep the MetaSkill focused on reusable generation principles rather than task specific memory, it does not store raw trajectory text, instance specific answers, object locations, or one off action patches. Candidate updates are bounded interventions rather than unrestricted rewrites. Internally, a proposal record stores its target decision, before and after value, supporting diagnosis, expected changed fields, invariants, and intended behavioral effect. After interaction aware resolution, the corresponding resolved update record is provided as structured input to the frozen LLM skill change verifier; neither proposal nor resolved update records are exposed to the executor.

A.2 INITIAL METASKILL M_0

All experiments start from the same hand-authored, task-neutral MetaSkill M_0, used unchanged across benchmarks and executor scales. M_0 is hand-authored and never adapted to any task data; it encodes deliberately conservative defaults for the four MetaSkill decisions:

- SELECT: retain only verified transitions occurring before critical failures, require at least two independent supporting occurrences, deduplicate repeated evidence, and down-weight failure-prefix (0.6) and exploration-noise (0.2) segments;
- REPRESENT: admit workflow, state-transition, prior, constraint, and exception forms, preferring state transitions; positional priors are allowed and benchmark-specific representations are disabled by default;
- SCOPE: scope every rule by goal signature and execution stage, require an observable stop condition and a fallback, and cap consecutive search misses at twelve;
- COMPOSE: assemble at most twelve units, preserve workflow, search, operation, and stopping coverage, merge redundant units, and order them by execution dependency.

At initialization the conditional-principle set and the validated history are empty ($H_0 = \emptyset$). The shared principle store is capped at 12 installed principles and 36 activation/negative-guard predicates, and the validated-history buffer stores at most 10 entries. The numeric defaults are hand-set and shared across all benchmarks. Each main benchmark and model setting evolves its own persistent MetaSkill from this common initialization.

A.3 SKILL GENERATOR REALIZATION

The main paper abstracts skill generation as

$$S = \text{Generate}(R; M). \tag{11}$$

Our implementation separates the MetaSkill-side decision record from the rendering of the final skill text. This separation gives the frozen LLM verifier explicit before-and-after inputs for checking whether a proposed MetaSkill update is reflected in the regenerated skill while preserving unrelated behavior. It is an implementation choice rather than a conceptual requirement of SKILL^2.

Every candidate MetaSkill generates a skill from scratch from the same fixed trajectory pool R. The candidate is never allowed to patch the incumbent skill directly. Consequently, later MetaSkill rounds do not receive privileged trajectories simply because they occur later in the optimization sequence.

A.4 FROZEN LLM SKILL-CHANGE VERIFIER

Before a candidate consumes validation rollouts, a frozen LLM verifier checks whether the regenerated skill realizes the meta-level change declared by the MetaSkill update. Its inputs are the resolved update $\Delta\widehat{M}_t$, the incumbent and candidate MetaSkills $(M_t, \widetilde{M}_{t+1})$, the incumbent and regenerated skills $(S_t, \widetilde{S}_{t+1})$, and the resolved update's declared invariants. The verifier returns the binary signal g_t^{change} used in Eq. 8. The verifier is held fixed throughout an optimization run and does not generate MetaSkill updates, edit skills, or use validation utility to select a candidate; utility acceptance is handled separately by execution on $\mathcal{D}_{\text{val}}$. Candidates that fail this check are rolled back before validation execution.

A.5 VALIDATED UPDATE HISTORY

Each accepted update creates a compact validated-history entry with the following information:

```
{
  "decision_type": "SELECT | REPRESENT | SCOPE | COMPOSE",
  "condition": "when the principle applies",
  "principle": "execution-validated generation principle",
  "negative_guard": "when not to apply",
  "protected_effect":
    "generation behavior that should be retained",
  "paired_validation": {...},
  "confidence": "...",
  "status": "active | superseded"
}
```

Only updates that satisfy Eq. 10 can enter the validated history H_t. Upon acceptance, the resolved generation principle is applied to the MetaSkill as a conditional principle and reaches the skill generator through M. The history H_t stores only the bookkeeping record above (paired validation record, conditions, status), which is consumed exclusively by the resolver. The shared principle store is bounded to 12 installed principles and 36 activation/negative-guard predicates, while the validated-history buffer is capped at 10 entries. Raw trajectories, generated skills, and rejected proposals are excluded. Superseded principles are retained for audit rather than silently deleted.

A.6 INTERACTION-AWARE RESOLUTION

For each proposed update, the interaction-aware resolver receives H_t, retrieves potentially relevant generation principles, and evaluates three signals: (i) whether their applicability conditions overlap, (ii) whether they affect overlapping skill generation decisions, and (iii) whether the new update may alter behavior retained by an earlier principle. The proposal function U does not receive H_t. Using the interaction categories in Section 3.4, the resolver assigns

$$r(\Delta M_t, h_i) \in \{\text{TRANSFER}, \text{NON INTERFERING}, \text{INTERFERENCE}\}. \tag{12}$$

TRANSFER reinforces or extends validated behavior, NON INTERFERING affects sufficiently separate conditions or behavior, and INTERFERENCE may alter behavior protected by an earlier accepted principle. These labels describe assessed interactions between generation principles; INTERFERENCE denotes an interaction risk rather than an observed execution regression. Lexical disagreement alone is not sufficient for interference.

If no relevant entry indicates interference, the proposed edit can be incorporated into a candidate while retaining earlier principles. Interfering or weakly supported proposals are handled through four resolution operations. RETAIN keeps an existing principle and abstains when evidence for changing it is insufficient. ISOLATE separates conflicting principles by adding an applicability condition or negative guard. CONSOLIDATE combines compatible information into a more precise conditional generation principle. REPLACE substitutes an earlier principle only when new execution feedback directly contradicts it within its stated conditions. Here, ISOLATE acts over applicability scopes rather than model parameters. These operations allow apparent contradictions to be represented as conditional generation knowledge. When resolution produces a candidate, it emits both a resolved update record $\Delta\widehat{M}_t$ (containing the final before-and-after values and invariants) and the candidate MetaSkill obtained by applying that resolved update to M_t. Appendix F.2 documents a recorded RETAIN decision, including the distinction between the raw model suggestion and the canonical guarded decision.

A.7 OPTIMIZATION CONFIGURATION

Table 4 reports the fixed optimization hyperparameters used by SKILL^2. We use a short horizon of at most twelve evolution rounds and a conservative patience of eight consecutive rounds without an accepted update. The utility margin is set to $\delta = 0$, so every accepted update must strictly improve validation utility. The paired forgetting budget is $\rho = 1$, meaning that a candidate may introduce at most one success-to-failure forgetting event on the fixed validation set.

Hyperparameter	**Value**
Maximum evolution rounds $T_{\max}$	12
Patience P	8 rounds
Utility margin δ	0
Forgetting budget ρ	1 validation instance
Meta-Updater temperature	0.0
Meta-Updater max output tokens	240
Resolver temperature	0.0
Resolver classification max output tokens	400
Resolver resolution max output tokens	220
Shared principle capacity	12
Predicate capacity	36
Validated-history capacity	10

Table 4: **Default SKILL^2 optimization configuration.** The acceptance thresholds and capacities are fixed before final-test evaluation.

B COMPLETE OPTIMIZATION PROCEDURE

Algorithm 1 summarizes the complete optimization loop. The trajectory pool R is fixed across all rounds; training feedback and candidate validation use disjoint execution sets. Training feedback is cached: $\mathcal{D}_{\text{train}}$ is executed at initialization and then only after a candidate satisfies the full acceptance rule, whereas rejected candidates never trigger a training-feedback refresh.

Algorithm 1 SKILL^2: Continual MetaSkill Optimization

Require: Fixed trajectory pool R, training data $\mathcal{D}_{\text{train}}$, validation data $\mathcal{D}_{\text{val}}$, initial MetaSkill M_0, horizon $T_{\max}$, patience P, margin δ, forgetting budget ρ

$H_0 \leftarrow \emptyset$; $c \leftarrow 0$ ▷ c: consecutive no-update counter
$S_0 \leftarrow \text{Generate}(R; M_0)$
$F_0 \leftarrow \text{Execute}(S_0; \mathcal{D}_{\text{train}})$; $F_{\text{val}} \leftarrow \text{Execute}(S_0; \mathcal{D}_{\text{val}})$
for $t = 0, \ldots, T_{\max} - 1$ **do**
  $P_t \leftarrow \text{Diagnose}(R, M_t, S_t, F_t)$; $\Delta M_t \leftarrow U(M_t, P_t)$
  $(\Delta\widehat{M}_t, \widetilde{M}_{t+1}) \leftarrow \text{Resolve}(M_t, \Delta M_t, H_t)$; $a_t \leftarrow 0$ ▷ a_t: acceptance flag
  if resolution returns a candidate **then**
    $\widetilde{S}_{t+1} \leftarrow \text{Generate}(R; \widetilde{M}_{t+1})$
    $g_t^{\text{change}} \leftarrow \text{Verify}(\Delta\widehat{M}_t; M_t, \widetilde{M}_{t+1}, S_t, \widetilde{S}_{t+1})$
    if $g_t^{\text{change}} = 1$ **then**
      $\widetilde{F}_{\text{val}} \leftarrow \text{Execute}(\widetilde{S}_{t+1}; \mathcal{D}_{\text{val}})$
      $\Delta J_t \leftarrow \text{Util}(\widetilde{F}_{\text{val}}) - \text{Util}(F_{\text{val}})$; $n_t^{\text{reg}} \leftarrow \text{RegressionEvents}(F_{\text{val}}, \widetilde{F}_{\text{val}})$
      if $\Delta J_t > \delta$ **and** $n_t^{\text{reg}} \leq \rho$ **then** $a_t \leftarrow 1$ ▷ candidate is accepted
  if $a_t = 1$ **then**
    $\widetilde{F}_{\text{train}} \leftarrow \text{Execute}(\widetilde{S}_{t+1}; \mathcal{D}_{\text{train}})$ ▷ refresh once
    $(M_{t+1}, S_{t+1}, F_{t+1}) \leftarrow (\widetilde{M}_{t+1}, \widetilde{S}_{t+1}, \widetilde{F}_{\text{train}})$
    $H_{t+1} \leftarrow \text{Commit}(H_t, \Delta\widehat{M}_t, \widetilde{M}_{t+1})$; $F_{\text{val}} \leftarrow \widetilde{F}_{\text{val}}$; $c \leftarrow 0$
  else
    $(M_{t+1}, S_{t+1}, F_{t+1}, H_{t+1}) \leftarrow (M_t, S_t, F_t, H_t)$; $c \leftarrow c + 1$
  if $c \geq P$ **then break** ▷ early stop after P consecutive no-update rounds
return final MetaSkill, generated skill, and validated history

Here, RegressionEvents counts validation instances that succeed under the incumbent but fail under the candidate, matching n_t^{reg} in Eq. 10. For ScienceWorld, utility is the mean normalized task score, whereas a regression is counted only when an instance fully completed by the incumbent is not fully completed by the candidate. Score decreases on instances that the incumbent did not fully complete affect utility but do not count as regression events.

Validation utility follows the native criterion of each benchmark and is reported on a 0 to 100 scale in the experiments. Candidate acceptance always uses the corresponding validation utility difference ΔJ_t together with the prespecified forgetting guard. For benchmarks with binary task outcomes, paired learning and forgetting events provide an equivalent task-level view of the change. All acceptance thresholds are fixed before final-test outcomes are observed.

C Evaluation Protocol

C.1 Data Roles

We separate the trajectory pool, training feedback, validation, and final evaluation into four roles:

Trajectory pool R
fixed execution trajectories consumed by every skill generation and candidate regeneration;

Training data $\mathcal{D}_{\text{train}}$
execution instances used to obtain the cached feedback F_t for diagnosis. They are executed once for S_0 and thereafter only after a candidate satisfies the full acceptance rule to refresh F_{t+1}; rejected candidates do not re-execute $\mathcal{D}_{\text{train}}$;

Validation data $\mathcal{D}_{\text{val}}$
disjoint instances used for paired validation of incumbent and candidate skills and for acceptance;

Test data $\mathcal{D}_{\text{test}}$ a read only evaluation set accessed only after the evolution path, final skill, thresholds, stopping rule, and final state are frozen. It is used for the final main result and for the frozen checkpoint analysis in Sec. 4.3.

The training and validation data together form the optimization data available during MetaSkill evolution, whereas $\mathcal{D}_{\text{test}}$ cannot affect updating, rollback, stopping, threshold selection, or final state selection. The fixed trajectory pool, diagnosis data, validation data, and test data retain their stated roles throughout a run.

Benchmark	Trajectory evidence	Evolution		Read-only
	Pool R	$\mathcal{D}_{\text{train}}$	$\mathcal{D}_{\text{val}}$	$\mathcal{D}_{\text{test}}$
ALFWorld	$180 \times 4 = 720$	90	90	274 (140 + 134)
SpreadsheetBench	$60 \times 4 = 240$	90	90	400 (200 + 200)
WTQ	$128 \times 1 = 128$	64	64	2,810
ScienceWorld	$118 \times 1 = 118$	58	114	271

Table 5: **Dataset protocol.** R is the immutable trajectory pool used for skill generation; $\mathcal{D}_{\text{train}}$ provides update feedback; $\mathcal{D}_{\text{val}}$ governs candidate acceptance; and $\mathcal{D}_{\text{test}}$ is opened only after evolution is frozen. Counts use each benchmark's native unit (game, task, or question); parentheses split seen and unseen instances. The same SpreadsheetBench split is used for both executor scales.

C.2 Matched MetaSkill Arms

Greedy, History (w/o resolution), and full SKILL^2 are initialized from the same M_0 and share the same R, $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$, candidate budget, update horizon, skill generation procedure, frozen LLM skill-change verifier, executor configuration, random seeds, paired acceptance rule, and rollback rule. They differ only in how previously validated principles are retained and resolved:

- **Greedy**: applies each proposed update sequentially without persistent validated history or interaction-aware resolution;
- **History (w/o resolution)**: makes accepted generation principles available to the resolution stage but performs no explicit transfer and interference assessment or resolution;
- **SKILL^2**: passes H_t only to the interaction-aware resolver, which explicitly assesses transfer and interference before resolving and applying the edit.

This matched design separates the benefit of retaining earlier improvements from the additional benefit of reasoning about how new MetaSkill updates interact with previously acquired generation behavior.

C.3 Matched Studies and Reported Metrics

The interference ablation follows the same ALFWorld/Qwen3-8B data protocol as the corresponding main experiment, using the identical fixed trajectory pool, training set, validation set, and 274-instance read-only test set. It is conducted as a separate matched ablation run. Greedy, History (w/o resolution), and full SKILL^2 share the same ordered stream of 24 MetaSkill proposals and accept exactly five updates. A regeneration is one full call to the skill generator after resolution has produced a candidate MetaSkill. Table 3 reports final validation and test outcomes separately from the cumulative candidate-transition counts defined below.

Candidate-transition accounting. The rescues and regressions in Table 3 are accumulated over all candidates evaluated on the fixed validation set, including candidates subsequently rejected by the acceptance rule. For each evaluated candidate, outcomes are paired with those of the incumbent used for that candidate's validation comparison. A rescue is a failure-to-success transition, and a regression is a success-to-failure transition in that paired comparison. The counting unit is a candidate–validation-instance pair; the same instance can therefore contribute an event in more than one candidate comparison. Proposals stopped before validation do not contribute measured transition events.

Let $\mathcal{C}_{\text{val}}$ contain the candidates that undergo paired validation, and let n_c^{reg} be the number of regressions for candidate c. The last column of Table 3 is

$$\text{Reg./accepted update} = \frac{\sum_{c \in \mathcal{C}_{\text{val}}} n_c^{\text{reg}}}{N_{\text{accepted}}}, \qquad N_{\text{accepted}} = 5.$$

Thus, the reported values are $21/5 = 4.20$, $17/5 = 3.40$, and $7/5 = 1.40$. This normalization measures cumulative candidate regressions per accepted update, not the regression count caused by an average accepted update. The guard $n_t^{\text{reg}} \leq \rho$ with $\rho = 1$ constrains each accepted candidate separately; candidates that violate the guard are rejected but their measured regressions remain in the candidate-search totals. Consequently, those totals can exceed five even though only five updates are accepted.

Proposal-level mechanism corpus. The proposal-level analysis in Sec. 4.5 pools **197 logged proposal events**: 173 events from the eight fixed-evidence benchmark and executor cells and 24 events from the full SKILL^2 arm of the matched ALFWorld resolver ablation. The Greedy and History (w/o resolution) arms are not added again because they reuse the same underlying raw proposal sequence. The corpus is used for descriptive mechanism analysis; we do not interpret the 197 logged events as unique or statistically independent proposals.

The adaptation-efficiency comparison follows the full ALFWorld/Qwen3-8B main-experiment protocol: both SkillOpt and SKILL^2 use the identical fixed trajectory pool ($R = 180 \times 4$), $|\mathcal{D}_{\text{train}}| = 90$, $|\mathcal{D}_{\text{val}}| = 90$, and the 274-instance read-only test set, together with the same executor and execution configuration. Results are aggregated over three independent seeds. Adaptation cost includes the model and executor calls required during optimization while excluding fixed trajectory collection and final test evaluation; detailed results are reported in Table 10 and Appendix E.

C.4 Fresh-Evidence Transfer Protocol

The fresh-evidence transfer study is conducted on ALFWorld/Qwen3-8B by partitioning the trajectory evidence into two disjoint, equally sized trajectory pools. The source pool R_{src} contains trajectories from 90 tasks ($90 \times 4 = 360$ trajectories), while the target pool R_{tgt} contains trajectories from another 90 tasks ($90 \times 4 = 360$ trajectories), with $R_{\text{src}} \cap R_{\text{tgt}} = \emptyset$. The evolved MetaSkill M_T is optimized using only R_{src}; R_{tgt} is unavailable to diagnosis, MetaSkill updating, interaction resolution, verification, candidate acceptance, and checkpoint selection during evolution.

After evolution is complete, M_T is frozen. We then generate two fresh target skills, $\text{Generate}(R_{\text{tgt}}; M_0)$ and $\text{Generate}(R_{\text{tgt}}; M_T)$, using the same target trajectories, frozen generator, executor, and execution configuration. No target-side MetaSkill adaptation is permitted. Both skills are evaluated on the same read-only target evaluation set D_{target} containing 274 ALFWorld games. Reported target utilities are averaged over the same three seeds (41, 42, and 43) used in the main experiments. Neither R_{tgt} nor D_{target} is used for MetaSkill evolution or checkpoint selection.

Data	Size	Role
R_{src}	$90 \times 4 = 360$ trajectories	MetaSkill evolution only
R_{tgt}	$90 \times 4 = 360$ trajectories	Fresh skill generation after freezing M_T
D_{target}	274 games	Read-only target evaluation

Table 6: **Fresh-evidence transfer data split.** The source and target trajectory pools contain disjoint tasks. Target trajectory evidence and target evaluation outcomes are unavailable during MetaSkill evolution and checkpoint selection.

MetaSkill	Target evidence	Target utility ↑
Static M_0	unseen R_{tgt}	42.70
Evolved M_T	unseen R_{tgt}	**65.82**
Gain	*N/A*	**+23.12 pp**

Table 7: **Fresh-evidence transfer on ALFWorld/Qwen3-8B.** Both MetaSkills induce from the same unseen target trajectory pool after M_T is frozen; results average three seeds.

C.5 Execution and Statistical Controls

Incumbent and candidate skills are evaluated on identical validation instances with matched environment initialization, decoding configuration, action interface, and step limit. The incumbent validation result is cached; within a round, only the candidate validation execution is new. Acceptance additionally enforces a paired forgetting guard $n_t^{\text{reg}} \leq \rho$. The frozen LLM verifier is also held fixed across matched arms. The same controls apply to all arms of the matched ablation. The task executor uses temperature 0.0 in all benchmark and model settings; benchmark-specific output and interaction limits are listed in Table 8.

Benchmark	Max output tokens	Max steps
ALFWorld	128	50
SpreadsheetBench	3,072	10
WTQ	120	1
ScienceWorld	128	100

Table 8: **Benchmark-specific executor limits.** The same limits are used for incumbent and candidate execution within each benchmark.

C.6 Main Results with Seed Variability

Table 9 reports the seed-level variability for the main results in Table 1. Each entry is the mean ± standard deviation over three independent seeds. The main paper reports the corresponding means for readability.

Method	SpreadsheetBench		WTQ		ALFWorld		ScienceWorld	
	Qwen3-8B	Qwen3-30B	Qwen3-8B	Qwen3-30B	Qwen3-8B	Qwen3-30B	Qwen3-8B	Qwen3-30B
No-Skill	15.83 ± 1.00	17.67 ± 1.20	53.91 ± 0.30	60.64 ± 0.40	41.73 ± 1.10	50.73 ± 0.70	13.85 ± 0.20	34.65 ± 1.20
LLM Skill	17.33 ± 0.90	20.50 ± 2.20	51.28 ± 0.20	52.95 ± 0.10	42.58 ± 0.20	55.96 ± 0.40	12.09 ± 1.50	33.20 ± 1.20
Trace2Skill	19.75 ± 0.00	31.25 ± 0.40	52.78 ± 0.10	62.21 ± 0.30	44.65 ± 2.50	60.34 ± 2.90	21.33 ± 2.00	40.27 ± 0.60
SkillOpt	25.25 ± 0.40	30.75 ± 1.20	58.15 ± 0.60	60.53 ± 0.40	62.77 ± 3.70	70.07 ± 0.60	18.74 ± 0.90	34.21 ± 1.40
SkillGen	20.50 ± 0.80	16.75 ± 1.30	54.02 ± 2.60	60.93 ± 0.10	62.41 ± 4.50	63.75 ± 1.10	13.01 ± 0.90	41.31 ± 1.40
EvoSkill	24.25 ± 1.90	20.75 ± 0.40	53.95 ± 0.10	61.89 ± 2.00	61.31 ± 2.10	62.04 ± 1.10	14.33 ± 2.50	38.07 ± 5.20
Text optimization methods								
TextGrad	11.00 ± 1.00	22.50 ± 1.40	61.71 ± 4.10	60.14 ± 1.10	52.92 ± 2.40	40.15 ± 0.80	23.11 ± 1.40	39.07 ± 0.90
GEPA	10.25 ± 1.10	16.25 ± 1.00	52.63 ± 0.10	61.32 ± 0.30	50.00 ± 6.00	45.99 ± 2.60	23.62 ± 2.70	43.72 ± 4.60
MetaSkill control								
Static MetaSkill (M_0)	9.75 ± 0.00	19.50 ± 1.10	51.89 ± 0.20	60.75 ± 1.10	41.36 ± 0.20	53.89 ± 1.10	12.89 ± 0.20	36.38 ± 1.00
Skill^2 (Ours)	**27.75 ± 0.40**	**34.25 ± 1.30**	**66.28 ± 0.20**	**69.30 ± 0.30**	**75.55 ± 1.30**	**71.17 ± 0.90**	**26.56 ± 1.90**	**49.74 ± 2.40**

Table 9: **Main results with seed variability.** Each entry reports mean ± standard deviation over three independent seeds. The evaluation protocol, data partitions, executor, and scoring metric are shared across methods within each benchmark and model setting.

C.7 Frozen Checkpoint Analysis

Table 1 reports the performance of the final skill obtained at the predeclared final MetaSkill M_T, together with the Static MetaSkill (M_0) control. For the learning dynamics analysis in Sec. 4.3,

we additionally revisit the already frozen sequence $M_0, \ldots, M_T$ only after optimization has ended. For each checkpoint, a fresh skill is generated from the same fixed trajectory pool R and evaluated under the same benchmark, executor, and test protocol as the corresponding main experiment. No checkpoint outcome is used for MetaSkill updating, rollback, stopping, threshold selection, or final state selection. This analysis therefore measures how the skill generation procedure changes across the frozen evolution path while keeping trajectory evidence and evaluation conditions fixed.

The gains in Figure 4 compare fresh skills regenerated from M_t and M_0 within this checkpoint analysis. They are distinct from subtracting the Static MetaSkill (M_0) result from the final SKILL^2 result in Table 1, which compares the skills evaluated in the main experiment. The two comparisons therefore need not have identical numerical gains.

D BENCHMARK AND BASELINE PROTOCOL DETAILS

Benchmark metrics. We preserve the native evaluation criterion of each benchmark and place the reported value on a common 0 to 100 scale. ALFWorld and SpreadsheetBench use task success, WTQ uses denotation exact match, and ScienceWorld uses normalized score. The common scale is used only for reporting; it does not change the benchmark-specific evaluator.

Baseline configurations. Trace2Skill, SkillOpt, SkillGen, and EvoSkill are run using the optimization procedures and recommended settings from their corresponding papers and official implementations. Within each benchmark and model comparison, the executor, test instances, executor interface, and benchmark-native scorer are shared across methods. This separates differences in skill generation procedure from differences in the final evaluation environment.

E ADAPTATION COST ACCOUNTING

Accounting boundary. Adaptation cost covers initial skill generation, initial training and validation executions, MetaSkill proposals and resolution, candidate generation and verification, candidate validation, and training-feedback refresh after acceptance. Collection of the fixed trajectory pool and all evaluation performed after evolution is frozen, including the final and frozen checkpoint evaluations, are outside this boundary. Let b_t indicate that resolution produces a candidate, v_t that the candidate passes skill-change verification, and a_t that it passes the full acceptance rule. The work performed by the optimization loop is

$$\begin{aligned} C_{\text{adapt}} = C_{\text{init}} + \sum_t \Big[& C_{\text{update},t} + C_{\text{resolve},t} \\ & + b_t\big(C_{\text{generate},t} + C_{\text{verify},t}\big) + v_t C_{\text{val},t} + a_t C_{\text{train},t} \Big], \\ C_{\text{init}} = C_{\text{generate},0} & + C_{\text{train},0} + C_{\text{val},0}. \end{aligned} \tag{13}$$

For executor-rollout accounting, only the execution terms contribute. Incumbent validation feedback is cached, so $C_{\text{val},t}$ contains only the new candidate execution. Reported token costs follow the recorded model-call coverage within the adaptation boundary.

Matched ALFWorld comparison. The study in Table 10 follows the full ALFWorld/Qwen3-8B main-experiment protocol. Both methods use the identical fixed trajectory pool ($R = 180 \times 4$), training set ($|\mathcal{D}_{\text{train}}| = 90$), validation set ($|\mathcal{D}_{\text{val}}| = 90$), and read-only 274-instance test set, together with the same executor and execution configuration. As reported in Section 4.6, the three-seed mean test utility is 62.77 for SkillOpt and 75.55 for SKILL^2, a gain of 12.78 points. The main-result standard deviations are reported in Table 9. Mean adaptation cost decreases from 1,590 to 540 executor rollouts (-66.0%) and from 161.90M to 44.80M recorded tokens (-72.3%).

Metric	ALFWorld / Qwen3-8B		
	SkillOpt	**SKILL^2 (Ours)**	**Improvement**
Held out test utility ↑	62.77	75.55	+12.78 pp
Adaptation rollouts ↓	1,590	540	-66.0%
Adaptation tokens ↓	161.90M	44.80M	-72.3%

Table 10: **Adaptation efficiency on ALFWorld/Qwen3-8B.** Both methods use the same data protocol and accounting boundary. Improvement is computed relative to SkillOpt.

Implementation-level savings. SKILL^2 can avoid repeated execution at four stages. First, interaction resolution can abstain before skill regeneration when a proposed update conflicts with validated behavior and lacks sufficient evidence for replacement. Second, the frozen verifier can reject skills that fail to realize the declared change before validation execution. Third, incumbent validation feedback is cached, so only the candidate validation execution is new within a round. Fourth, training feedback is refreshed only after a committed update; rejected candidates do not re-execute the training set. These mechanisms allocate downstream execution selectively while preserving the full paired acceptance check.

F RECORDED METASKILL UPDATES

Recorded setting. We present two recorded cases from ALFWorld 0.4.2 using Qwen3-30B-A3B-Instruct-2507 with AWQ-W4A16 weights, seed 41, thinking disabled, and a maximum of 50 execution steps. The cases concern cooling and heating tasks, respectively. The paired-validation results reported below each use 15 instances and are specific to these case records. In the implementation, `assemble` corresponds to the paper's COMPOSE decision. The two cases come from different task families and do not form a single evolution sequence.

F.1 CASE A: AN ACCEPTED UPDATE TO SKILL COMPOSITION

Execution evidence and diagnosis. A representative failed episode required the agent to cool a mug and place it in a coffee machine. The agent picked up the mug at step 49 and reached the refrigerator at step 50, exhausting the budget before cooling and final placement. Selected terminal actions were:

> **Step 47:** `go to coffeemachine 1`
> **Step 49:** `take mug 1 from coffeemachine 1`
> **Step 50:** `go to fridge 1`

The diagnosis summarized seven failed support episodes in this cooling task family and associated the selected proposal with four episode IDs. It recorded supporting episode IDs without exact per-episode step ranges. The incumbent skill already contained separate search, cooling, and placement instructions; the diagnostic hypothesis was that explicit stage routing could improve how these rules were used.

Localized MetaSkill edit. The selected update enabled one COMPOSE parameter:

> `render_mutually_exclusive_execution_stages`: `false` $\longrightarrow$ `true`.

The intended change concerned stage routing in the selected units. The evidence, represented units, and scoped units were declared invariant. Validated history was empty, and the canonical decision added the update without a history-conflict rewrite. No interaction label was recorded for this case.

Regenerated skill and change check. Fresh regeneration added an execution-stage router before the task rules. The router instructed the executor to select the first matching observable stage, use only compatible rules, and recompute the stage after each observable transition. Its stage-selection priority was COMPLETE, VERIFY, COMMIT, TRANSFORM, ACQUIRE_OR_INSPECT, and otherwise DISCOVER; these are conditional routing checks, not a mandatory temporal sequence. Within the selected stage, the router was instructed to preserve the existing search order, object preference, numeric-ID policy, fallback, and stopping condition, and to choose among already induced rules.

The recorded change check reported changes to the selected units and final skill, with the declared upstream fields unchanged. Both the induction-fidelity and resolution-contract checks passed. A separate natural-language verifier explanation was not recorded.

Paired validation and commit. On the 15 validation instances used in this case, the incumbent succeeded on 6 and the candidate on 8. The paired records contained two failure-to-success transitions and no success-to-failure transitions, giving a gain of $2/15$, or 13.33 percentage points. The update was accepted, and its generation principle and paired-validation evidence were added to history. The two rescued instances involved a tomato and a wine bottle; they were distinct from the mug episode used to illustrate the diagnostic evidence. This case documents the realization and acceptance of a COMPOSE update, with no unresolved-versus-resolved conflict comparison.

F.2 CASE B: RETAINING A PREVIOUSLY VALIDATED SEARCH SETTING

Validated history and a new proposal. A previously accepted SCOPE update reduced `maximum_consecutive_search_misses` from 12 to 6. In that paired validation, successes increased from $10/15$ to $11/15$, with one rescue and no regressions. The incumbent skill instructed the executor to visit a different receptacle family after six distinct unsuccessful locations in the current family.

In a subsequent round, continuing search failures motivated a proposal to reduce the threshold from 6 to 5. The diagnosis reported five failed support episodes. A representative episode required placing a hot cup in a cabinet and exhausted the 50-step budget while searching. The proposed update targeted the same SCOPE parameter as the previously accepted change.

Raw proposal and canonical decision. The raw model completion labelled the proposal compatible and suggested a merge, reasoning that earlier fallback continued the direction of the previously accepted update. The canonical guarded assessment instead labelled it `conflicting`: the proposal revisited the same decision under overlapping diagnostic conditions and extrapolated beyond the previously validated value. The final decision was RETAIN, with the recorded abstention reason `verified_coordinate_reentry`. The label `conflicting` is retained here as recorded in the artifact; it denotes an assessed interaction risk, not a measured performance loss.

> Previously accepted: search-miss threshold $12 \rightarrow 6$.
> New proposal: search-miss threshold $6 \rightarrow 5$.
> Canonical decision: retain 6 and abstain.

Outcome and interpretation. No candidate skill was regenerated for the $6 \rightarrow 5$ proposal. Change verification and paired validation were therefore not performed, the incumbent remained active, and no new history entry was committed. This case illustrates abstention governed by a previously validated setting. It does not establish that a threshold of five would reduce utility or that six is an optimal threshold.

Record	MetaSkill change	Paired validation	Decision
A: cooling	COMPOSE stage routing: off $\rightarrow$ on	$6/15 \rightarrow 8/15$; 2 rescues, 0 regressions	Commit
B: prior update	SCOPE threshold: $12 \rightarrow 6$	$10/15 \rightarrow 11/15$; 1 rescue, 0 regressions	Commit
B: later proposal	SCOPE threshold: $6 \rightarrow 5$	Not evaluated	Retain 6; abstain

Table 11: **Recorded decisions and outcomes for the two cases.** The later proposal in Case B was stopped before regeneration and evaluation.

Scope of the examples. Case A illustrates an accepted update and its realized change in a generated skill; Case B illustrates a guarded decision to retain a previously validated setting before regeneration. Neither case demonstrates an ISOLATE or CONSOLIDATE decision followed by fresh skill regeneration and acceptance. General effectiveness is assessed through the aggregate experiments and matched comparisons, while these cases make the recorded update operations and their outcomes concrete.