LEARNING FROM REVISION CONSEQUENCES: HINDSIGHT META-EXPERIENCE DISTILLATION FOR SELF-IMPROVING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

As agents continuously improve by generating and revising Skills, the process that discovers and refines those Skills becomes a learnable object in its own right. Task-Skills directly act on task execution, whereas Meta-Skills govern how agents discover and improve future Skills; their value therefore emerges through the subsequent search processes they induce. Existing approaches improve Meta-Skills from observed raw Skill-search trajectories and branch outcomes. However, branch performance entangles the effects of the initial discovery state and the Meta-Skill revision that generated the search process, making it difficult to characterize what a particular revision actually changed, and potentially favoring revisions evaluated from advantageous starting states. We introduce **HMED** (**H**indsight **M**eta-**E**xperience **D**istillation), a mechanism for constructing **Meta-Experience** for self-improving agents. HMED restores the entry state of the completed search event cited by the revision and re-executes the incumbent and revised Meta-Skills from that state, so that the changes associated with the revision can be observed under a shared condition. Each comparison is distilled into a Meta-Experience, a structured record that can be reused by future updates, so that even revisions that are not ultimately retained still contribute a learning signal. Across three interactive agent benchmarks and two representative large language models with agentic capabilities, HMED consistently improves Skill discovery performance over strong baselines, shifting Meta-Skill learning beyond branch outcomes toward the consequences of changing the improvement process.

1 INTRODUCTION

Agents increasingly improve by turning interaction into reusable capabilities. A household agent that repeatedly has to locate objects can accumulate a Skill for finding them, so that a lesson learned once is available later without being rediscovered (Shinn et al., 2023; Zhao et al., 2023; Wang et al., 2023). As agents operate in increasingly open environments, however, improvement is no longer limited to acquiring individual capabilities. An agent must also improve the process through which it discovers, evaluates, and refines those capabilities. We refer to the policy governing Skill discovery and evolution as a *Meta-Skill*, and to the capabilities it produces as *Task-Skills*. A Task-Skill determines how an agent performs a task and can usually be evaluated through task outcomes. A Meta-Skill instead shapes how future capabilities are generated. Its value is realized only through the search processes it subsequently guides.

Existing approaches revise prompts and improvement policies using execution trajectories, descendant outcomes, and accumulated evolution histories (Fernando et al., 2023; Agrawal et al., 2025; Wang et al., 2026). In ongoing Skill evolution, the library and search history change across branches. Outcomes observed from different discovery states therefore do not directly show how the incumbent and revised Meta-Skills would guide search from the same starting point. A branch's final outcome does not directly reveal the contribution of a particular Meta-Skill revision: a favorable discovery state may make a weak revision appear effective, while a useful revision may appear ineffective when evaluated from a difficult starting point. In other words, a good branch is not necessarily a good improver.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

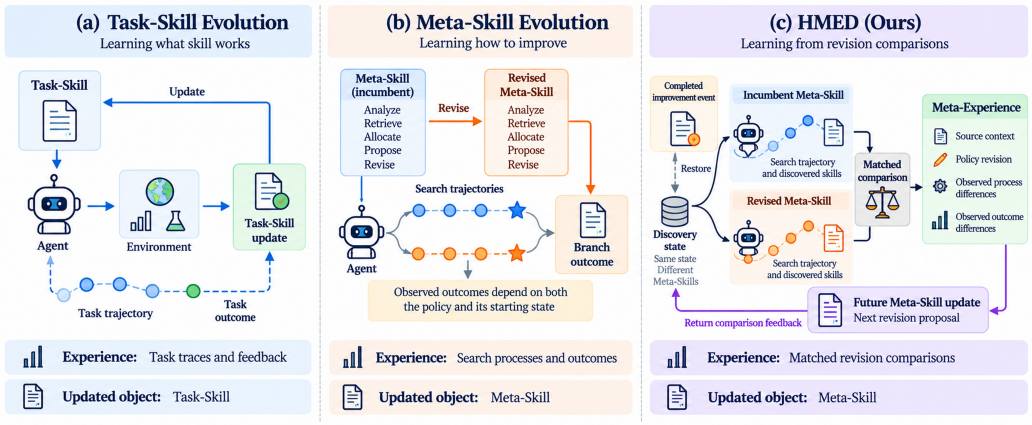

Figure 1: Learning signals in Skill evolution. Task-Skill methods update from task outcomes; existing Meta-Skill evolution updates from branch-level search signals, where the starting discovery state and the Meta-Skill are mixed; HMED updates from matched revision consequences, where the same entry state is restored and the Meta-Skill revision is the intended experimental difference.

This limitation comes from a mismatch between the experience collected by existing Skill evolution methods and the experience required for learning a reusable improvement policy. A single search trajectory records what happened under one Meta-Skill, but provides no direct comparison with its alternative. We address this gap with an experience unit centered on the revision itself: what changed in the improvement process, and what followed from that change. We call this unit *Meta-Experience*. Conceptually, it links four aspects of a revision: the completed event it cites, the modification to the Meta-Skill, the changes observed in the subsequent search process, and the resulting changes in discovered capabilities. For example, a revision that adjusts the retrieval component of the Meta-Skill may show which prior experience was consulted differently and whether that change led to different discoveries. Unlike Task-level Experience, which describes whether an existing capability can solve a task, Meta-Experience describes what a change to the improvement process produced.

We introduce **Hindsight Meta-Experience Distillation (HMED)** to construct and reuse this experience. HMED restores the entry state of the completed search event cited by the revision and re-runs the search under both the incumbent and revised Meta-Skills from that shared starting point. The resulting paired traces are distilled into a compact Meta-Experience record. This record enters the next round of Meta-Skill updates, where it can inform how future revisions are proposed. A comparison therefore serves two roles: it decides whether to retain the current revision, and it contributes a record that shapes later revisions. Records of valid comparisons remain available regardless of adoption, so even a non-adopted revision can inform a later update.

We evaluate HMED on three interactive agent benchmarks, using DeepSeek-v4-Flash and Qwen3-32B. HMED achieves the highest mean downstream performance in all six model–benchmark configurations, and an ALFWorld ablation shows that returning revision consequences to later updates provides additional gains over using comparison only for selection. When the learned Meta-Skill is frozen and the Task-Skill library is reset, it continues to guide effective discovery under the reuse protocol at new ALFWorld discovery origins. Together, these results suggest that learning an improvement policy benefits from an experience representation aligned with the object being learned. Our contributions are threefold:

- **Meta-Experience for Self-Improving Agents.** We identify the mismatch between branch-level Skill-evolution outcomes and the revision-level experience required for learning reusable improvement policies, and introduce Meta-Experience as a distinct experience unit that links the completed event, the revision, the observed process changes, and the resulting discovery outcomes of a Meta-Skill update.

- **Hindsight Meta-Experience Distillation.** We introduce HMED, a mechanism that constructs Meta-Experience by grounding Meta-Skill revisions in completed improvement

events, re-running the incumbent and revised Meta-Skills from matched discovery states, and distilling the resulting comparison into records that future updates can read.

- **Empirical Evidence for Learning from Revision Consequences.** We evaluate HMED across interactive agent environments and distinguish the value of consequence feedback from comparison-based selection, showing that learned Meta-Skills remain effective under the reuse protocol at new ALFWorld discovery origins.

2 Related Work

2.1 Self-Improving Agents and Evolving Improvement Procedures

Agents increasingly transform interaction experience into reusable capabilities. Early work such as Reflexion converts failures into verbal feedback for future trials (Shinn et al., 2023), ExpeL distills reusable lessons from repeated interactions (Zhao et al., 2023), and Voyager maintains a growing Skill library that can be accumulated, retrieved, and composed (Wang et al., 2023). As the learned object expands from task behavior to the mechanisms that produce such behavior, self-improvement becomes increasingly recursive. PromptBreeder evolves the prompts that guide generation itself (Fernando et al., 2023), GEPA updates textual instructions through reflection on execution trajectories and feedback (Agrawal et al., 2025), and SkillEvolver iteratively refines reusable Skill artifacts through execution-driven improvement (Zhang et al., 2026a). Related directions in recursive self-improvement further explore agents that modify their own reasoning, search, or capability construction procedures (Zhang et al., 2025a; Kim et al., 2026; Zhang et al., 2026b). These approaches progressively shift the focus from improving what an agent does to improving how an agent improves.

Recent work further advances this direction by making the improvement procedure itself an editable object. MetaSkill-Evolve introduces a two-timescale evolution framework in which Task-Skills and Meta-Skills co-evolve, with the Meta-Skill governing components such as failure analysis, experience retrieval, budget allocation, candidate proposal, and Skill revision (Wang et al., 2026). Its updates incorporate accumulated evolution signals such as descendant improvements and diagnoses, while maintaining branch-local Skill and Meta-Skill states. This line of work establishes the improver itself as a persistent object that can be represented, revised, and inherited. As the scope of learning shifts from task capabilities to the improvement process, a natural question emerges: what form of experience should guide the evolution of such an improver?

2.2 Learning Improvement Policies from Experience

Learning how to improve a search or optimization procedure from previous experience has a long history in machine learning. Meta-learning studies how experience across tasks can produce better initializations or adaptation strategies (Finn et al., 2017; Ravi & Larochelle, 2017); learned optimizers parameterize update rules and learn optimization behaviors from prior training trajectories (Andrychowicz et al., 2016; Oh et al., 2020); and evolutionary search has been used to discover reinforcement learning update rules (Co-Reyes et al., 2021). These approaches share a common principle: a search process produces not only a solution, but also experience that can shape future search. Large language models extend this idea by representing improvement procedures as natural-language policies and open-ended agent workflows. EvoPrompt evolves natural-language prompts through evolutionary operators (Guo et al., 2024), while recent agent systems learn from execution feedback, memory, and accumulated experience to improve future behavior (Zhang et al., 2025b; Ouyang et al., 2025; Wu et al., 2025). As the optimized object moves from parameterized optimization rules toward language-based improvement policies that govern diagnosis, retrieval, proposal, and revision, the experience available to such policies becomes increasingly heterogeneous. A final search outcome indicates whether a branch succeeded, but it reflects both the starting discovery state and the improvement procedure that generated it. While GEPA evaluates prompt variants on the same minibatch (Agrawal et al., 2025), HMED constructs revision-centered experience for future Meta-Skill updates by restoring the entry states of cited Skill-evolution events.

This distinction connects to several neighboring methodological areas. Hindsight Experience Replay makes failed attempts useful by relabeling the objective associated with an observed trajectory (Andrychowicz et al., 2017); in contrast, hindsight in HMED identifies which Meta-Skill revision

should be revisited and which completed improvement event provides the basis for examining its consequences. Paired comparisons under shared conditions are widely used to reduce variance and improve interpretability in simulation and experimental design (Owen, 2013), while off-policy evaluation estimates the performance of one policy from observations generated by another (Thomas et al., 2015). HMED follows a direct comparison perspective by running matched continuations rather than estimating the value of an unseen policy from logged trajectories. These directions motivate our study of revision-centered experience for learning language-based improvement procedures.

3 METHOD

3.1 META-EXPERIENCE FOR META-SKILLS

An agent that improves itself operates on two levels. At the task level, it executes a policy π on an environment and receives a task outcome $r \in \mathbb{R}$. At the meta level, it maintains a Meta-Skill m that governs how Task-Skills are discovered, evaluated, and revised across windows. It consists of textual guidance for diagnosis, experience retrieval, search allocation, candidate proposal, and Skill revision. A Task-Skill determines how the agent performs a task; a Meta-Skill instead shapes the capability-production process itself. Let x denote the discovery state at the entry to a search step: the Skill library, the visible history, and the current task context. Running Skill discovery from x under m produces an observed branch outcome $y = f(x, m, \xi)$, where ξ denotes execution randomness. The outcome depends on both the starting discovery state and the Meta-Skill, as well as stochastic execution.

This distinction creates a core difficulty for Meta-Skill learning: branch outcomes reflect both the starting state and the improvement policy. A revision may appear effective simply because it was evaluated from a favorable discovery state, and a useful revision may appear weak because it was evaluated from a difficult one. Branch outcomes that do not control or model the starting discovery state cannot directly serve as paired evidence about a particular Meta-Skill revision, because the branches may start from different x.

We therefore construct a *revision-centered* form of experience, one that records the consequences of changing the improvement process. We call this experience **Meta-Experience**. Its structure is determined by what is needed to characterize a revision and its consequences: the completed event it cites, the modification it makes, the changes it produces in the subsequent search process, and the resulting changes in discovered capabilities. We write the record as

$$\mathcal{E}_t = (e_t, \Delta m_t, \Delta_{\text{process}}, \Delta_{\text{discovery}}),$$

where e_t provides the revision's provenance, Δm_t describes the Meta-Skill change, Δ_{process} captures how the intermediate discovery trajectory changed—for example, how prior experience was utilized, how candidates were generated, and whether a recurring failure was resolved—and $\Delta_{\text{discovery}}$ records the resulting change in discovered capabilities. The learning object of Meta-Experience is thus not task execution itself, but the consequences of changing the improvement process.

3.2 HINDSIGHT META-EXPERIENCE DISTILLATION

We propose Hindsight Meta-Experience Distillation (HMED), a mechanism that constructs Meta-Experience for Meta-Skill learning. HMED restores the entry state of the completed search event cited by the revision and re-runs the search under the Meta-Skills before and after the modification from the same starting point. Here, *hindsight* refers to using a completed search event to ground a revision and its subsequent comparison, and *distillation* refers to converting the resulting paired traces into a compact update-level record that later revisions can read. One Meta-Skill update therefore proceeds in three steps: a revision is proposed and grounded in a completed event; the incumbent and revised Meta-Skills are re-run from matched discovery states; and the resulting comparison is distilled into a Meta-Experience record.

A candidate revision is first produced from the existing improvement process and bound to its source event, giving the modification a clear problem context and verification location. The incumbent and revised Meta-Skills are then re-run from matched discovery states, and the search differences they

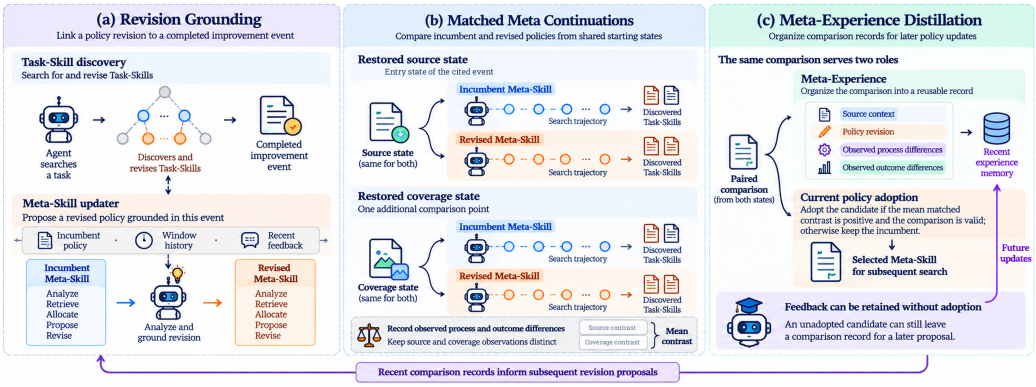

Figure 2: Overview of Hindsight Meta-Experience Distillation (HMED). A Meta-Skill revision cites a completed search event and is evaluated through matched continuations from that event's entry state and an additional discovery state. The comparison provides an adoption signal and a Meta-Experience record for future revisions.

induce are compared. This comparison is finally distilled into Meta-Experience and returned to future Meta-Skill updates. One revision comparison therefore produces two outputs: a selection signal that determines whether the current revision continues to be used, and a Meta-Experience record that carries what the comparison revealed about the proposed change.

3.3 Revision Grounding in Completed Events

Each Meta-Skill revision cites a completed Task-Skill evolution step with a recorded search result. The cited event provides the revision's provenance and an entry state from which to compare the incumbent and candidate Meta-Skills. Its entry state is the discovery state immediately before that step executes.

For example, a completed search step may propose a Task-Skill that repeats an unsuccessful strategy already present in the library. The updater could cite this event when revising the Meta-Skill's candidate-generation guidance to ask for a different strategy. The event identifies the search context in which to examine the proposed change: the available Skills, visible history, and task context before that proposal was generated.

Let the current Meta-Skill be m_t, the search window be W_t, and the feedback from prior Meta-Experience be F_{t-1} (Section 3.5). The candidate revision and its source event are represented as

$$(m'_t, e_t) = G(m_t, W_t, F_{t-1}),$$

where G denotes the Meta-Skill update procedure and e_t denotes the completed Task-Skill evolution step cited by the revision. A revision modifies the Meta-Skill representation while preserving the underlying discovery framework; it is a textual change to the policy that governs search, not a replacement of the search procedure itself. The event e_t identifies the search step cited by the revision. HMED restores the entry state of this completed search event:

$$x_t^{src} = \text{Restore}(e_t).$$

The comparison starts before the cited Task-Skill evolution step executes, not from the state reached at its completion or when the revision was proposed. HMED also evaluates the revision from an additional discovery state:

$$\mathcal{X}_t = \{x_t^{src}, x_t^{cov}\}.$$

The source state anchors the comparison to the cited search context, while the coverage state supplies an observation from an additional discovery state.

3.4 Matched Meta Continuations

Matched Meta Continuations are the core mechanism by which HMED constructs Meta-Experience. They hold the discovery state fixed while varying the Meta-Skill, so that the change associated with the revision can be observed under a shared reference condition.

In the example above, both continuations begin with the library and history available before the repetitive proposal. One uses the incumbent guidance; the other uses the revised guidance. Their trajectories show what each search produced and how its utility developed over the continuation. This comparison connects a proposed change in search guidance to observed discovery outcomes from the same starting point.

For any discovery state $x \in \mathcal{X}_t$, HMED runs the current Meta-Skill m_t and the candidate revision m'_t separately:

$$\tau^-(x) = \mathcal{S}(x, m_t), \qquad \tau^+(x) = \mathcal{S}(x, m'_t),$$

where $\mathcal{S}$ denotes a Skill discovery continuation. The two continuations start from the same discovery state, so they share the same initial conditions, and the Meta-Skill revision is the intended experimental difference. Since $\mathcal{S}$ is a sampled execution with model or environment randomness, the paired result is a local observed contrast under matched starting conditions. We compute it as

$$\Delta(x) = g(\tau^+(x)) - g(\tau^-(x)),$$

where

$$g(\tau) = \frac{1}{H_\tau} \sum_{k=1}^{H_\tau} (u_{\mathrm{best},k} - u_{\mathrm{start}})$$

is the anytime-mean utility gain over H_τ realized steps, where u_{start} is the utility at the restored entry state and $u_{\mathrm{best},k}$ is the running-best utility after step k. We set $g(\tau) = 0$ when $H_\tau = 0$. This score rewards improvements achieved earlier in the continuation: searches that reach the same final best utility can receive different gains depending on when that utility is attained. For multiple evaluation states, HMED aggregates:

$$\Delta_t = \frac{1}{|\mathcal{X}_t|} \sum_{x \in \mathcal{X}_t} \Delta(x).$$

Holding the discovery state fixed removes differences in the starting state from this comparison. Both arms are re-executed, so Δ_t summarizes observed continuation differences under the matched conditions.

Based on the matched comparison, the current Meta-Skill is updated according to the revision consequence:

$$m_{t+1} = \begin{cases} m'_t, & \Delta_t > 0 \text{ and the comparison passes validity checks,} \\ m_t, & \text{otherwise.} \end{cases}$$

A comparison is eligible for adoption when the restored states are valid, the candidate changes at least one Meta-Skill component, and the modified components are supplied at every realized step of the candidate continuation. Early convergence is allowed under the continuation protocol.

3.5 Distilling Revision Consequences into Meta-Experience

HMED stores a structured comparison record using the representation in Section 3.1 and supplies subsequent Meta-Skill updates with a bounded textual serialization of recent records. This distillation is deterministic and requires no additional language-model call.

The record links the cited event and proposed Meta-Skill change to the paired search trajectories and their outcomes. In the representation of Section 3.1, these observations describe the process and discovery consequences of the revision. The feedback summarizes these observed consequences through trajectory summaries, utility gains, realized search effort, and the adoption outcome. Keeping the observations from both starting states preserves differences that the aggregate adoption score can obscure, such as a gain at the source state accompanied by a loss at the coverage state.

Let F_t denote the feedback constructed from recent records in $\mathcal{E}_{\leq t}$. The next revision is proposed as

$$(m'_{t+1}, e_{t+1}) = G(m_{t+1}, W_{t+1}, F_t).$$

The updater receives this feedback alongside the current Meta-Skill and search window, so a new proposal can take account of how earlier revisions performed. For example, a source gain paired with a coverage loss gives the updater a reason to reconsider when a proposed strategy should apply. Valid non-adopted comparisons remain available for this purpose; invalid comparisons are excluded from adoption and are not treated as evidence of the revision's effect.

Table 1: Downstream task performance after Skill evolution. ALFWorld and ScienceWorld report success rate (%). BiomniBench-DA reports a judge score out of 100.

Model	Method	ALFWorld	ScienceWorld	BiomniBench-DA
Qwen3-32B	No Skill	45.15	16.68	14.33
	Static Skill	47.76	21.55	16.36
	SkillEvolver	49.25	20.59	18.33
	MetaSkill-Evolve	51.49	20.84	17.80
	HMED (ours)	**54.11**	**22.56**	**20.78**
DeepSeek-v4-Flash	No Skill	71.64	23.58	24.70
	Static Skill	73.51	26.33	22.84
	SkillEvolver	76.87	24.30	25.67
	MetaSkill-Evolve	76.49	28.70	36.80
	HMED (ours)	**80.60**	**32.47**	**43.19**

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate on three interactive agent benchmarks covering task execution, open-ended exploration, and scientific discovery. **ALFWorld** (Shridhar et al., 2021) provides a text-based embodied environment in which agents complete everyday tasks through long-horizon planning, tool use, and environment feedback; task success depends on accumulating and reusing effective action strategies. **ScienceWorld** (Wang et al., 2022) provides a multi-step exploration setting in which agents iteratively test hypotheses and refine strategies, so a useful Skill encodes a procedure rather than a single answer. **BiomniBench-DA** (Qu et al., 2026) targets complex biomedical analysis tasks that require coordinating multi-step reasoning, tool use, and knowledge retrieval. We use the official rubric-based LLM-judge procedure to score BiomniBench-DA agent trajectories; the other benchmarks are evaluated using the task success rates provided by the environment.

Models and settings. We evaluate on Qwen3-32B (Qwen Team, 2025) and DeepSeek-v4-Flash (DeepSeek-AI, 2026), two representative large language models with agentic capabilities. All methods run under identical inference configurations, with reasoning/thinking modes disabled. During initial evolution, each method interacts only with the training tasks specified by the benchmark and updates its Task-Skills or Meta-Skills through its own mechanism; the resulting Skill library and Meta-Skill are then frozen and evaluated on test tasks not used during evolution. Unless otherwise stated, we report mean results from two runs.

4.2 OVERALL SKILL DISCOVERY PERFORMANCE

We evaluate whether HMED improves downstream task performance after Skill evolution. We compare with **SkillEvolver** (Zhang et al., 2026a), which continually updates Task-Skills, and **MetaSkill-Evolve** (Wang et al., 2026), which additionally allows the Meta-Skill itself to evolve. We additionally include **No Skill** and **Static Skill** as reference points measuring the contribution of accumulated Skill knowledge without an evolving mechanism.

Table 1 reports the results. HMED achieves the highest mean in all six model–benchmark configurations. Relative to MetaSkill-Evolve, HMED improves ALFWorld from 51.49 to 54.11 ($+2.62$), ScienceWorld from 20.84 to 22.56 ($+1.72$), and BiomniBench-DA from 17.80 to 20.78 ($+2.98$) on Qwen3-32B. On DeepSeek-v4-Flash, HMED reaches 80.60 on ALFWorld, 32.47 on ScienceWorld, and 43.19 on BiomniBench-DA. Against the strongest baseline in each setting, the gains range from 1.01 to 2.62 points on Qwen3-32B and from 3.73 to 6.39 points on DeepSeek-v4-Flash. The following ablations examine the contribution of Meta-Experience feedback to these improvements.

Two observations follow from the baseline ordering. First, Static Skill improves over No Skill on most benchmarks, indicating that accumulated historical Skills can improve downstream performance. The benefit is not uniform: on DeepSeek-v4-Flash BiomniBench-DA, Static Skill scores 22.84, below No Skill at 24.70. Second, SkillEvolver and MetaSkill-Evolve are not uniformly better than Static Skill, and MetaSkill-Evolve does not consistently outperform SkillEvolver. These com-

Table 2: Ablation of Meta-Experience construction and feedback. All variants retain matched comparison. Comparison-only removes the explicit feedback returned to later updates; Outcome-only restricts its content. Values are success rate (%) on ALFWorld.

Variant	Event grounding	Matched comparison	Consequence feedback	ALFWorld
HMED (full)	✓	✓	✓	**54.11**
- Comparison-only	✓	✓	×	48.88
- Outcome-only Meta-Experience	✓	✓	Partial	51.49
- No hindsight grounding	×	✓	✓	47.76

Table 3: Agreement between branch outcome and matched-continuation Δ on Qwen3-32B. Agreement rate is computed over comparable cases (Agree + Disagree). Tie indicates that the matched Δ is zero or the sign cannot be determined.

Benchmark	Agree	Disagree	Tie	Agreement rate
ALFWorld	18	15	15	54.5%
ScienceWorld	6	18	12	25.0%

parisons suggest that how improvement signals are organized and reused can influence downstream self-improvement performance.

4.3 Ablation Study on Meta-Experience Construction and Use

The main results show that HMED improves downstream performance. This section examines which aspects of Meta-Experience construction and use drive that improvement. Table 2 reports four variants, all retaining matched comparison.

Selection versus learning. Matched comparison has two possible uses: deciding whether to adopt a revision, and informing how future revisions are proposed. The Comparison-only variant retains matched selection but removes revision-consequence feedback from subsequent updates, while preserving event grounding, matched continuations, and the adoption rule. According to Table 2, the full method exceeds this variant by 5.23 points. With matched selection retained in both variants, this difference supports the contribution of consequence feedback to subsequent updates in this setting.

Content and source. Two further variants target the construction of the record. Removing event grounding reduces the score to 47.76, indicating that anchoring the comparison to the event cited by the revision matters for downstream performance. Restricting the record to outcome-level information reduces the score to 51.49. Outcome-only remains competitive, suggesting that outcome information contributes substantially, while the additional process-level observations provide further benefit in this setting.

4.4 Analysis of Branch Outcomes and Revision Consequences

The ablation shows which components of Meta-Experience matter. A separate question is whether the matched comparison merely reproduces the branch outcome that existing methods rely on. If branch outcome already captured the consequence of a revision, additional Meta-Experience feedback would provide limited additional information. For each evaluated revision, we compare the sign of its original branch outcome with the sign of its matched-continuation Δ. A revision whose branch outcome is positive but whose matched Δ is negative, or vice versa, is a case where branch outcome alone and matched comparison give opposite directional signals.

Table 3 reports the results. In ALFWorld, 45.5% of comparable cases show opposite signs; in ScienceWorld, the disagreement rate is 75.0%. These results show that branch outcomes and matched comparisons are not interchangeable as directional signals in the evaluated revisions. The analysis characterizes their observed agreement; the feedback ablation above evaluates the contribution of revision-consequence feedback to later updates.

Table 4: Performance under the ALFWorld reuse protocol. Results report success rate (%) on the final evaluation split. For HMED and MetaSkill-Evolve, the learned Meta-Skill is frozen and the Task-Skill library is reset before new discovery.

Model	No Skill	Static Skill	SkillEvolver	MetaSkill-Evolve	HMED (ours)
Qwen3-32B	41.05	43.28	42.54	49.25	**50.75**
DeepSeek-v4-Flash	68.65	70.15	73.13	76.12	**76.87**

Appendix B provides a mechanism trace linking a Meta-Experience record to a later update request and subsequent proposal. The feedback ablation evaluates effectiveness; the trace documents the feedback pathway.

4.5 Reuse of Learned Meta-Skills under New Discovery Origins

The preceding sections evaluate the full evolution pipeline. A separate question is whether the learned Meta-Skill remains effective under a reuse protocol at new ALFWorld discovery origins. We test this by freezing the Meta-Skill and resetting the Task-Skill library before new discovery. In this protocol, we reuse the final Meta-Skills learned by HMED and MetaSkill-Evolve on the training origins; the agent then starts from an initial Skill library on the held-out set and runs one complete new round of Skill discovery guided by the frozen Meta-Skill; the resulting Skill library is frozen and evaluated on the new test set. The held-out and new test sets are obtained by randomly partitioning the test set into two disjoint halves, neither of which overlaps with the training tasks; this prevents the reuse phase from adapting directly to the final evaluation tasks.

Table 4 reports the results. HMED achieves the highest mean on both base models, with margins of 1.50 and 0.75 points over MetaSkill-Evolve. Since the Task-Skill library is reset before reuse, the remaining performance difference provides evidence that the learned Meta-Skill contributes beyond the stored Skill library. The learned Meta-Skill remains effective under the reuse protocol at new ALFWorld discovery origins.

Execution cost. Appendix C discusses evolution cost, task-execution cost, and amortization through reuse. In a supplementary ScienceWorld evaluation, HMED's Skills use 6.9% fewer model calls and 6.5% fewer environment steps than MetaSkill-Evolve's. These measurements complement the reuse experiment: the ALFWorld protocol evaluates the learned Meta-Skill at new discovery origins, while the ScienceWorld records characterize the subsequent task-execution cost of the produced Skills. Together, the two analyses examine the use of learned improvement policies and their resulting artifacts after the initial evolution stage.

5 Conclusion

Self-improving agents increasingly accumulate experience from interaction to discover and revise Skills. As the improvement process itself becomes an editable object, an important question arises: what form of experience should guide Meta-Skill learning? We introduce Hindsight Meta-Experience Distillation (HMED), which shifts the experience unit for Meta-Skill learning from search-branch outcomes to the consequences of Meta-Skill revisions on subsequent Skill discovery. HMED anchors each revision to the completed event it cites, compares the incumbent and revised Meta-Skills from matched discovery states, and distills the resulting revision consequences into a revision-centered Meta-Experience record. The comparison therefore provides both an adoption signal for the current revision and a Meta-Experience record for future updates. Across three interactive agent benchmarks and two base model families, HMED consistently improves downstream Skill discovery performance over strong baselines. Further experiments show that its benefit comes not only from comparison-based selection, but also from the learning signal provided by Meta-Experience, and that learned Meta-Skills remain effective under the reuse protocol at new ALFWorld discovery origins. These results support organizing experience around revision consequences when learning an editable improvement policy.

STATEMENT ON THE USE OF GENERATIVE AI

In this work, we used generative AI tools (primarily Codex) for implementing methods and creating/editing software code. We have not used generative AI tools for generating synthetic datasets, developing theoretical models, formulating mathematical claims, writing proofs, proposing hypotheses, designing the core research methodology, translation, data cleaning, qualitative analysis, or interpreting results. Additionally, we used generative AI tools for drafting parts of the paper and editing for readability. All LLM-generated or AI-assisted code was thoroughly reviewed, tested, and validated by at least two authors. All AI-assisted text was rewritten or heavily edited by the authors. We take full responsibility for the final content of this work.

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, 2025. URL https://arxiv.org/abs/2507.19457.

Marcin Andrychowicz, Misha Denil, Sergio Gómez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, 2016.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, 2017.

John D. Co-Reyes, Yingjie Miao, Daiyi Peng, Esteban Real, Sergey Levine, Quoc V. Le, Honglak Lee, and Aleksandra Faust. Evolving reinforcement learning algorithms. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2101.03958.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution, 2023. URL https://arxiv.org/abs/2309.16797.

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

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *International Conference on Learning Representations*, 2024.

Zae Myung Kim, Young-Jun Lee, Seungyeon Jwa, and Dongyeop Kang. Meta^n: Recursive self-improvement through emergent depth, 2026. URL https://arxiv.org/abs/2608.24735.

Junhyuk Oh, Matteo Hessel, Wojciech M. Czarnecki, Zhongwen Xu, Hado van Hasselt, Satinder Singh, and David Silver. Discovering reinforcement learning algorithms. In *Advances in Neural Information Processing Systems*, 2020.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory, 2025. URL https://arxiv.org/abs/2509.25140.

Art B. Owen. *Monte Carlo Theory, Methods and Examples*. Stanford University, 2013. URL https://artowen.su.domains/mc/.

Yuanhao Qu, Yingzhou Lu, Xinming Tu, Serena Zhang, Tianwei She, Alexander Glenn Shaw, Jou-Ho Shih, Bingqing Zhao, Minjie Shen, Haochen Yang, Jielin Yan, Rongchuan Zhang, Xinze Wu, Tingting Li, Bin Zhou, Ning Wang, Adam Ma, Le Cong, Xiaobo Hu, Yuan Jiang, Jiayun Dong, Tao Peng, Jure Leskovec, and Kexin Huang. BiomniBench: Process-level evaluation of LLM agents for real-world biomedical research. *bioRxiv*, 2026. doi: 10.64898/2026.05.12.724604. URL https://www.biorxiv.org/content/10.64898/2026.05.12.724604v2.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rJY0-Kcll.

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. URL https://papers.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

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. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Philip Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High confidence policy improvement. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2380–2388, 2015. URL https://proceedings.mlr.press/v37/thomas15.html.

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, 2023. URL https://arxiv.org/abs/2305.16291.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022. URL https://arxiv.org/abs/2203.07540.

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, 2026. URL https://arxiv.org/abs/2607.05297.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, and Botian Shi. EvolveR: Self-evolving LLM agents through an experience-driven lifecycle, 2025. URL https://arxiv.org/abs/2510.16079.

Genrui Zhang, Erle Zhu, Jinfeng Zhou, Caiyan Jia, and Hongning Wang. SkillEvolver: Skill learning as a meta-skill, 2026a. URL https://arxiv.org/abs/2605.10500.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin gödel machine: Open-ended evolution of self-improving agents, 2025a. URL https://arxiv.org/abs/2505.22954.

Jenny Zhang, Bingchen Zhao, Wannan Yang, Jakob Foerster, Jeff Clune, Minqi Jiang, Sam Devlin, and Tatiana Shavrina. Hyperagents, 2026b. URL https://arxiv.org/abs/2603.19461.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models, 2025b. URL https://arxiv.org/abs/2510.04618.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners, 2023. URL https://arxiv.org/abs/2308.10144.

A HMED UPDATE PROTOCOL

Our implementation follows the two-level organization used in MetaSkill-Evolve (Wang et al., 2026), where Task-Skill evolution is governed by an editable Meta-Skill. HMED operates in repeated Meta-Skill update windows, each comprising revision proposal, matched comparison, an adoption decision, and feedback construction. In each window, the updater proposes a revision and cites a completed Task-Skill evolution event with a recorded search result. The comparison restores the event's entry state as the source state and uses an additional coverage state for a supplementary observation. At each state, the incumbent and candidate are evaluated under the same continuation protocol, using anytime-mean utility gains over realized steps. A candidate is adopted only when the comparison is valid and the aggregate contrast is strictly positive. The resulting structured Meta-Experience record is retained, and recent records are deterministically serialized into bounded feedback for subsequent updates without an additional language-model distillation call.

For ALFWorld, the utility $u(s)$ is the success rate of Task-Skill s on the run's frozen 25-game learning frontier. The starting Skill and all continuation candidates share this frontier, which is separate from the final test set. The coverage state is restored from the entry snapshot of the most recent eligible completed event preceding the current window in a lineage selected by a fixed schedule, skipping the source lineage. This choice is made before either continuation is executed and does not use their outcomes. Entry snapshots are checked for event provenance, snapshot integrity, task resolution, Meta-Skill consistency, and agreement with the frozen frontier. Each continuation executes at most $H = 2$ Task-Skill discovery steps and stops early when no unsolved origin task remains. Gain is averaged over realized steps, with $g(\tau) = 0$ for a zero-step continuation; if all four continuations execute zero steps, the comparison is recorded as untested and the incumbent is retained. An unusable source reference likewise yields an untested comparison, whereas missing required coverage or a violation of the frozen frontier integrity check terminates the run.

B CASE STUDY: TRACING META-EXPERIENCE THROUGH THE UPDATE LOOP

This appendix provides a mechanism trace of the Meta-Experience feedback pathway: a comparison record enters a later update request, followed by a new proposal and comparison. The trace documents how comparison records enter later update requests and how their content corresponds to subsequent proposals. We analyze one training run of DeepSeek-v4-Flash on ALFWorld (seed 2), which contains four Meta-Skill update windows under the protocol in Appendix A. We take the earliest complete chain in the run and include the last comparison in the same run as a counterpoint. All numbers and texts are quoted directly from the run records.

Throughout this appendix, *Gain* denotes the anytime-mean utility improvement relative to the starting utility:

$$g(\tau) = \frac{1}{H_\tau} \sum_{k=1}^{H_\tau} \left(u_{\mathrm{best},k} - u_{\mathrm{start}} \right),$$

where H_τ is the number of realized continuation steps and $u_{\mathrm{best},k}$ is the running-best utility after step k. *Steps* reports realized steps over allowed steps, and the u_{best} column lists the running-best sequence. Each row in a comparison table corresponds to a single continuation.

B.1 AN ADOPTED CHAIN: WINDOW 0 TO WINDOW 1

Step 1: The original comparison record. Window 0 produced the first Meta-Skill comparison in this run. Table 5 lists its key fields.

The outcome is that the candidate is adopted. The record contains both starting states, the per-arm best-so-far trajectories, the gains, the step counts, and the inheritance decision. The record is serialized into the feedback provided to the next window's updater.

Step 2: How the next revision uses the Meta-Experience. Window 1's candidate rewrites three of the five components: the allocator, the proposer, and the evolver. The new allocator rule is

Table 5: Comparison record at window 0. The candidate gains a positive Δ at the source state and zero at the coverage state.

Start	Continuation	u_{start}	Steps	u_{best}	Gain
1 (source)	incumbent m	0.80	2/2	[0.80, 0.80]	0.0000
1 (source)	candidate m'	0.80	2/2	[0.84, 0.84]	0.0400
2 (coverage)	incumbent m	0.84	2/2	[0.84, 0.84]	0.0000
2 (coverage)	candidate m'	0.84	2/2	[0.84, 0.84]	0.0000

conditioned on gain values of the same form as those reported in window 0's record, as shown in Table 6.

Table 6: Correspondence between the gains in window 0's record and the condition in window 1's allocator rule.

Gains in window 0's record	Condition in window 1's allocator rule
source: $+0.0400$; coverage: 0.0000	"If the previous window produced exactly one archived child with a small positive Δ (≤ 0.05) and at least one non-archived child, keep K at 2 but require the two children to differ in edit action."

The phrase "small positive Δ" is numerically compatible with the $+0.0400$ and 0.0000 in the record. The condition also refers to the archive status of Task-Skill children, which the paired gains alone do not determine. The same candidate additionally introduces an evolver instruction that prefers `modify` over `add`, with a rationale consistent with the pattern in the record, where the candidate gained at the source state and was flat at the coverage state.

Step 3: The next comparison. Window 1's own comparison provides a subsequent positive outcome for that revision, as shown in Table 7.

Table 7: Comparison result at window 1. The candidate outperforms the incumbent at both starting states; the aggregate matched contrast is $\Delta_t = +0.04$, and the candidate is adopted.

Start	Continuation	u_{start}	Steps	u_{best}	Gain
1 (source)	incumbent m	0.76	2/2	[0.76, 0.76]	0.0000
1 (source)	candidate m'	0.76	2/2	[0.80, 0.84]	0.0600
2 (coverage)	incumbent m	0.80	2/2	[0.80, 0.84]	0.0200
2 (coverage)	candidate m'	0.80	2/2	[0.84, 0.84]	0.0400

The candidate text is delivered at every step; retrieval and proposals differ from the incumbent arm at both states, while the failure tag, the allocator width, and the operation type remain the same. The candidate is adopted and becomes the incumbent for the following windows.

Step 4: Evidence that the record reached the updater. Window 0's comparison is written into a feedback block that appears verbatim in both of window 1's requests. This directly documents the record's inclusion in the updater's input.

B.2 A Non-adopted Revision

The feedback chain from window 2 to window 3 provides a counterpoint.

Window 2's comparison is shown in Table 8. The candidate gains at the source state but loses by a larger amount at the coverage state. The aggregate matched contrast is negative, so the candidate is not adopted and the incumbent is kept.

Table 8: Comparison record at window 2. The candidate leads at the source state but falls behind at the coverage state; the aggregate matched contrast is $\Delta_t = -0.01$, and the candidate is not adopted.

Start	Continuation	u_{start}	Steps	u_{best}	Gain
1 (source)	incumbent m	0.76	2/2	[0.84, 0.84]	0.0800
1 (source)	candidate m'	0.76	2/2	[0.88, 0.88]	0.1200
2 (coverage)	incumbent m	0.76	1/2	[0.88]	0.1200
2 (coverage)	candidate m'	0.76	2/2	[0.80, 0.84]	0.0600

Window 3's updater receives a feedback block containing the comparisons from windows 1 and 2, and proposes a candidate that sharpens the rules for `inventory_loop` failures. That candidate loses in its own comparison, as shown in Table 9.

Table 9: Comparison result at window 3. The candidate ties with the incumbent at the source state and loses at the coverage state; the aggregate matched contrast is $\Delta_t = -0.03$, and the candidate is not adopted.

Start	Continuation	u_{start}	Steps	u_{best}	Gain
1 (source)	incumbent m	0.76	2/2	[0.84, 0.84]	0.0800
1 (source)	candidate m'	0.76	2/2	[0.84, 0.84]	0.0800
2 (coverage)	incumbent m	0.76	1/2	[0.88]	0.1200
2 (coverage)	candidate m'	0.76	2/2	[0.80, 0.84]	0.0600

The updater receives the earlier comparison record and proposes a new revision, which is subsequently evaluated and not adopted. Its changes do not enter the final Meta-Skill.

B.3 Summary of the Case Study

Together, the two chains illustrate the Meta-Experience loop of HMED:

- a comparison is recorded and distilled into Meta-Experience;
- the record is included in a later update request, and part of its content corresponds to a subsequent proposal;
- the new proposal is again tested by a matched comparison, and is either adopted or rejected.

The case study documents the feedback pathway from a comparison record to the input of a later Meta-Skill update, followed by another matched comparison.

C Computational Cost and Reuse

HMED incurs additional computation during evolution by comparing Meta-Skill revisions through matched continuations. Across the measured runs, HMED's evolution cost is on average 27% higher than MetaSkill-Evolve's. We distinguish this initial evolution cost from the recurring cost of executing tasks with the resulting Skills. Table 10 reports task-execution costs from a supplementary ScienceWorld evaluation comprising three trials. We report model calls and environment steps, which measure the amount of model interaction and environment interaction during execution.

In this evaluation, HMED's resulting Skills require 6.9% fewer model calls and 6.5% fewer environment steps than MetaSkill-Evolve's. Model calls are also reduced by 23.7% relative to No Skill and by 4.4% relative to Static Skill. The additional comparison computation during evolution is thus accompanied by lower task-execution costs in the measured workload.

Repeated deployment spreads the initial evolution cost across subsequent tasks. We estimate the break-even point in model calls as $N_b = (C_{\text{evo}}^{\text{HMED}} - C_{\text{evo}}^{b})/(C_{\text{exec}}^{b} - C_{\text{exec}}^{\text{HMED}})$, where b denotes a baseline, C_{evo} is the initial evolution cost in model calls, and C_{exec} is the execution cost per round in model calls. One execution round repeats the full workload of the three ScienceWorld trials in

Table 10: Task-execution costs in the supplementary ScienceWorld evaluation, aggregated over three trials. Counts cover execution with the resulting Skills after evolution.

Method	Model calls	Environment steps
No Skill	254,561	139,974
Static Skill	203,229	107,995
MetaSkill-Evolve	208,732	109,624
HMED	194,311	102,511
Reduction vs. MetaSkill-Evolve	6.9%	6.5%

Table 10. Table 11 uses the ScienceWorld-specific evolution counts and assumes that the measured execution cost per round remains constant.

Table 11: Estimated break-even points for repeated ScienceWorld execution, measured in model calls. HMED uses 194,311 execution calls per round in these records. Fractional rounds indicate equivalent execution volume.

Baseline	Execution calls/round	Break-even rounds
No Skill	254,561	2.1
MetaSkill-Evolve	208,732	6.2
SkillEvolver	208,688	6.7
Static Skill	203,229	14.6

Under this cost model, HMED recovers its additional evolution calls relative to MetaSkill-Evolve after approximately 6.2 equivalent execution rounds; further reuse accumulates call savings. The reuse experiment in Section 4.5 also shows that a learned Meta-Skill can guide Skill discovery at new ALFWorld origins. Together, these results illustrate the role of learning a reusable improvement policy: the initial learning effort produces a Meta-Skill that can guide later discovery, with its resulting Skills available for repeated task execution.