RETHINKING SELF-EVOLVING AGENTS: DO WE STILL NEED PRESCRIBED OPTIMIZATION PIPELINES?

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-evolving agents are usually built around prescribed optimization pipelines: the framework decides how to gather evidence, revise a persistent artifact, select candidates, and stop. We ask whether this task-specific procedure remains necessary when a frontier model acts as the optimizer. We introduce Open-Ended Optimization (OEO), which fixes the objective, permitted interactions, resource budget, data boundary, and evaluation while allowing the optimizer to compose the improvement process online. We compare OEO with two complementary prescribed approaches: SKILLOPT, a staged pipeline with bounded edits, and GEPA, a reflective evolutionary search. Across a complete grid of 16 head-to-head comparisons over 8 benchmark–target-model settings, GPT-5.5-driven OEO produces 12 higher final-score point estimates, 1 tie, and 3 lower estimates, with a median absolute difference of 3.1 percentage points. It improves the shared initial skill in all 8 settings while using a median 34.3% of SKILLOPT's configured target-interaction token allocation. Across 18 repeated optimization runs on 2 anchor settings, OEO retains the highest mean on LiveMath, whereas the 3 methods' SearchQA means lie within 0.84 percentage points. A capability probe further exposes the boundary of delegation: a weak optimizer cannot execute the open-ended loop under the fixed interface, prescription is task-dependent at medium capability, and frontier OEO is competitive on both tested tasks. A one-shot control shows that the gains are not explained by a single prior-driven rewrite. Finally, the fully instrumented OEO–SKILLOPT pair shows that prescription changes committed optimization paths more consistently than selected item-level behavior. Together, these results suggest that task-specific optimization procedures need not always be prescribed in advance. With a sufficiently capable optimizer, the procedure itself can instead be composed online within externally fixed objectives, permitted interactions, resource budgets, data boundaries, and evaluation, while explicit prescription remains a task- and capability-dependent scaffold rather than a universal requirement.

1 INTRODUCTION

Self-evolving agents seek to turn experience into persistent improvement: they revise prompts, memories, or skills so that future tasks can be solved better (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2023). In many systems, the framework also prescribes the procedure that turns experience into an update: which evidence to collect, how to revise the artifact, which candidate to keep, and when to stop. This arrangement makes sense when the model primarily supplies proposals within a framework-controlled optimization loop. More capable models, however, can use task feedback to reason about the improvement process itself, rather than only generate candidate revisions (Yang et al., 2024; Yuksekgonul et al., 2024; Agrawal et al., 2026). If improvement can be measured, must the framework still prescribe the route to it?

We study this question through persistent natural-language skill optimization. A *target model* executes tasks using a reusable skill, while an *optimizer model* improves that skill for subsequent tasks. In Open-Ended Optimization (OEO), a frontier optimizer receives the current skill, accumulated observations, permitted interactions, and remaining budget. It decides what to inspect, which target rollouts to request, how to revise the skill, and when to stop. The framework still fixes the objective, permitted interactions, data boundary, budget, and final evaluation. We compare this optimizer-composed procedure with two prescribed approaches: SKILLOPT's staged, bounded edit-

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.

and-validation pipeline (Yang et al., 2026) and GEPA, where a framework runs reflective evolutionary search over free-form textual mutations (Agrawal et al., 2026). The comparison centers on where control over the task-specific improvement procedure resides under a common external contract.

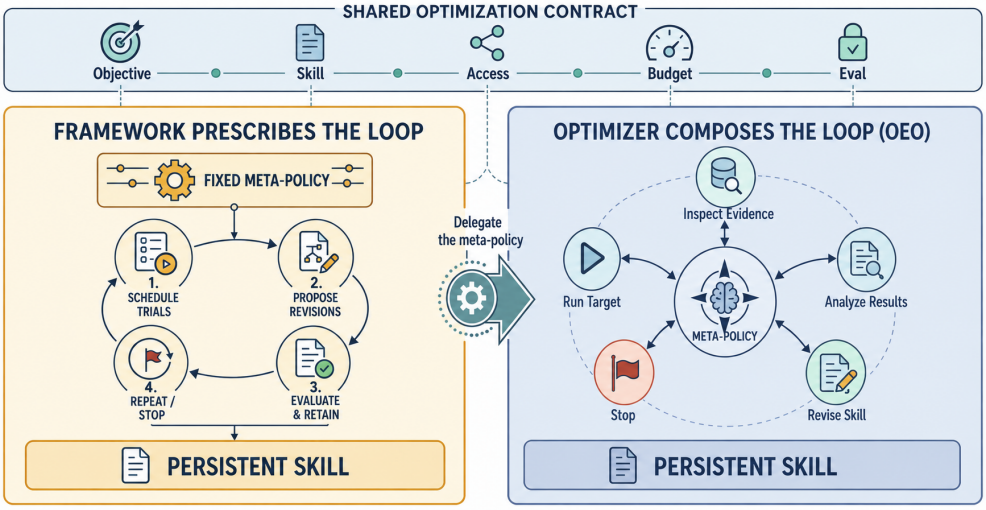

Figure 1: **Where should optimization responsibility live?** Under the same external optimization contract, a prescribed procedure fixes the task-specific improvement process in advance; OEO lets the optimizer compose it online. In the header, Access covers permitted interactions and the data boundary, while Eval covers the frozen evaluator and sealed final split.

With GPT-5.5 as the optimizer, OEO improves the shared initial skill in all 8 benchmark–target-model settings. Across 16 head-to-head comparisons with the prescribed approaches, OEO attains a higher final-score point estimate in 12, ties in 1, and is lower in 3, with a median absolute difference of 3.1 percentage points. It uses a median 34.3% of SKILLOPT's configured target-interaction token allocation.

Prescription remains valuable in some settings: GEPA leads on both SpreadsheetBench targets, and SKILLOPT outperforms OEO with a medium optimizer on SearchQA. Taken together, these results suggest that task-specific optimization procedures need not always be prescribed in advance. With a sufficiently capable optimizer, the procedure itself can be composed online within an externally fixed optimization contract, while explicit prescription remains a task- and capability-dependent scaffold rather than a universal requirement.

2 FROM PRESCRIBED PROCEDURES TO OPEN-ENDED OPTIMIZATION

The central distinction is between governing an optimization problem and prescribing how to solve it. We hold the problem fixed and compare how much of the improvement procedure the framework specifies in advance.

2.1 PROBLEM SETUP: CONTRACT AND META-POLICY

The *target model* M_{tgt} executes tasks with a skill, while the *optimizer model* M_{opt} revises that skill from interaction evidence; one model identity may serve both roles. An external *optimization contract* $\mathcal{C}$ fixes the objective, initial skill s_0, target model, permitted interactions and data boundary, resource budget B, frozen evaluator, and sealed final split D_{final}.

The *optimization meta-policy* is the procedure that acts within this contract: it decides what evidence to gather, how to interpret it, when and how to revise the skill, which candidate to retain, and when to stop. It governs how the skill is improved, not how the target model solves an individual task. Across our comparisons, the contract stays fixed; what varies is which of these decisions the framework prescribes and which the optimizer makes during the run.

In a *prescribed procedure*, an external controller fixes substantial parts of the meta-policy before optimization begins, and the model supplies diagnoses or revisions within it. In *Open-Ended Optimization* (OEO), the runner still enforces $\mathcal{C}$, but M_{opt} chooses and sequences the improvement actions online.

2.2 Two representative forms of prescribed optimization

We select two different forms of prescription. SkillOpt represents a staged skill-training pipeline with bounded edits; GEPA allows free-form textual mutation within a reflective evolutionary search. Their edit languages and search topologies differ, but both frameworks organize major parts of the improvement process before the run. Table 1 shows which decisions each method fixes and which OEO leaves to the optimizer.

Table 1: **Control allocation in the frontier comparison.** The contract is shared; the task-specific improvement process is controlled differently.

Control dimension	SkillOpt	GEPA	OEO
External contract	Shared objective, initial skill, target model, permitted interactions, data boundary, resource budget, frozen evaluator, and sealed final split		
Evidence scheduling	Fixed rollout and reflection batches	Parent and minibatch selection	Optimizer-directed
Diagnosis	Success/failure decomposition	Trace-conditioned reflection	Optimizer-directed
Revision space	Bounded patch operators	Free-form textual mutation	Optimizer-selected edit or rewrite
Candidate retention	Strict validation-improvement gate	Pareto-aware pool update	Optimizer decides when to commit or replace the current skill
Schedule and stopping	Fixed stages and epochs	Budgeted evolutionary loop	Adaptive within the budget
Final return rule	Validation-selected best	Validation-selected candidate	Current committed skill at finish

SkillOpt: staged and bounded skill training. SkillOpt schedules evidence collection and reflection, restricts revisions to bounded patch operators, and accepts only validation-improving candidates (Yang et al., 2026). Its prescription reaches both the search schedule and the form of an update.

GEPA: reflective evolutionary search. GEPA lets the model write free-form, trace-conditioned mutations, while the framework selects parents and retains candidates through a Pareto evolutionary loop (Agrawal et al., 2026). Its prescription governs the search and selection of candidates rather than their edit language.

Together, the comparators test OEO against two distinct forms of externally prescribed optimization. All three methods consolidate experience into a persistent skill; the question is where control over the improvement procedure resides. Shared starting conditions and alignment details appear in Section 3 and Appendix E.

2.3 OEO: an optimizer-composed procedure over a general interface

To make this delegation operational, OEO starts from s_0 and gives M_{opt} a general runner with five capabilities: InspectData, ReadArtifact, RunTarget, WriteSkill, and Finish. At each turn, the optimizer sees the current skill, accumulated observations, and remaining budget. It may combine these capabilities within one request; they are tool affordances, not prescribed stages or task-facing actions of the target model. For example, it may inspect a failed case, request a target rollout, revise the skill, and then decide what evidence to seek next. The runner executes permitted requests, records the returned evidence and cost, presents that history on the next turn, and keeps the final split sealed. When the optimizer finishes or the budget expires, the current committed skill is returned for sealed evaluation. Algorithm 1 in the appendix gives the complete loop.

OEO therefore does not remove the improvement procedure; it lets the optimizer compose that procedure as evidence arrives. The following experiments ask whether an optimizer-composed procedure yields competitive skills, and when external prescription still helps.

3 Can the Optimizer Compose the Improvement Procedure?

We test the central proposition in three steps. First, we ask whether a frontier optimizer can turn measurable task feedback into competitive skill improvement without inheriting a task-specific procedure. Second, we vary optimizer capability to see how the role of prescription changes. Third, we remove interaction entirely to test whether a strong model's static prior can explain the result. Section 4 then examines what prescription changes inside the optimization process.

Experimental setting. Experiments span SearchQA (search-snippet QA), OfficeQA Full (document QA), SpreadsheetBench, and LiveMath, with Qwen3.5-4B and GPT-5.5 as target models (Dunn et al., 2017; Opsahl-Ong et al., 2026; Ma et al., 2024; He et al., 2026; Qwen Team, 2026b; OpenAI, 2026). Within each benchmark–target setting, methods share the initial skill, permitted data and target interface, resource envelope, sealed examples, and evaluator. GPT-5.5 serves as the optimizer in the frontier comparison.[1] We compare selected artifacts on paired sealed items and use repeated optimization seeds to assess run-level variation. Target-interaction use is expressed relative to SkillOpt's configured reference allocation, not its realized cost; Appendices A and E give the statistical and accounting details.

3.1 A frontier optimizer can compose the improvement procedure online

Headline finding. A frontier optimizer can achieve effective skill improvement without inheriting a task-specific optimization procedure prescribed in advance. Across four task domains and two target models, OEO produces higher final-score point estimates than SkillOpt in 7 of 8 settings and ties in the remaining setting; against GEPA, it leads in 5 of 8 settings. Meanwhile, OEO uses a median 34.3% of SkillOpt's configured target-interaction token allocation. These results show that, under a shared external contract, the improvement procedure itself can be composed online by a capable optimizer, although explicit prescription remains advantageous on some tasks.

Table 2: **Canonical sealed-split scores with GPT-5.5 as optimizer.** All methods start from the same skill; bold marks the best optimized score per row. Every GEPA cell uses its native full-envelope, validation-selected artifact. Multi-seed anchors appear in the appendix.

Benchmark	Target	n	Initial	SkillOpt	GEPA	OEO
SearchQA	Qwen3.5-4B	1,400	0.6693	0.7221	0.7507	**0.7514**
SearchQA	GPT-5.5	1,400	0.7857	0.8657	**0.8736**	0.8714
SpreadsheetBench	Qwen3.5-4B	280	0.2107	0.1964	**0.3036**	0.2464
SpreadsheetBench	GPT-5.5	280	0.3964	0.7607	**0.8036**	0.7607
OfficeQA	Qwen3.5-4B	172	0.2616	0.2849	0.2500	**0.3081**
OfficeQA	GPT-5.5	172	0.5872	0.7035	0.6977	**0.7267**
LiveMath	Qwen3.5-4B	124	0.3065	0.6129	0.6210	**0.6532**
LiveMath	GPT-5.5	124	0.3871	0.5000	0.4032	**0.6855**

Competitive gains do not require a prescribed procedure. Table 2 shows that all 3 procedures can improve a persistent skill, but only OEO improves every shared initial artifact. Its competitiveness spans two different forms of prescription: SkillOpt supplies staged editing and validation, whereas GEPA supplies evolutionary search and selection. The exception is instructive: GEPA leads on both SpreadsheetBench targets even though its textual mutations are free-form. What distinguishes OEO is therefore not freedom to edit text, but freedom to decide how to gather evidence, revise,

[1] All GPT-5.5 calls use the GitHub Copilot Responses API.

and select. Paired analysis resolves the larger margins while leaving several narrow differences unresolved (Appendix A). The best procedure still depends on the task.

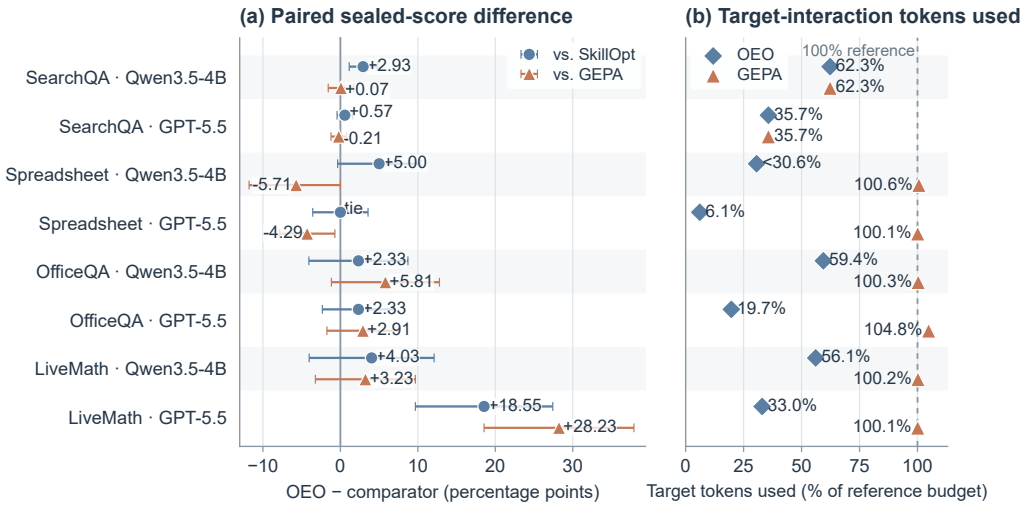

Figure 2: **Relative performance and target-interaction use.** Left: OEO score differences with paired 95% bootstrap intervals. Right: target-interaction tokens consumed by OEO and GEPA, expressed as a percentage of the common configured reference budget; the dashed line marks 100%. This reference budget is inherited from SKILLOPT's configured allocation and is not its realized token usage. All 8 GEPA cells use native full-envelope selection.

Open-ended optimization allocates interaction selectively. Figure 2 shows that OEO does not obtain its result by spending more target interaction: it remains below the configured reference allocation in every setting. The allocation is a ceiling for OEO, rather than a required rollout schedule. SpreadsheetBench makes the trade-off visible. There, GEPA uses approximately the full reference allocation and achieves the highest score for both targets, while OEO uses substantially less target interaction. The procedures can therefore differ in how much evidence they gather. These ratios describe optimization-time target interaction, not total monetary or deployment cost; full accounting appears in Appendix D.

Repeated runs reveal task-dependent variation. Figure 3 tests whether the frontier result depends on one trajectory. On LiveMath, OEO retains the highest mean and upside, but also has the largest run-to-run variation. On SearchQA, all 3 methods repeatedly reach similar scores and none wins every seed. Thus, the competitive result persists across repeated runs, while the balance between upside and consistency differs by task. Appendix B reports all 18 runs and exact counts.

3.2 OPTIMIZER CAPABILITY CHANGES THE ROLE OF PRESCRIPTION

We next hold the target model fixed at Qwen3.5-4B and vary the optimizer from Qwen3.5-4B to Qwen3.5-27B and GPT-5.5 (Qwen Team, 2026b;a; OpenAI, 2026). For OEO, the same action interface is used at all three levels; the matched medium OEO and SKILLOPT runs additionally share identical configured optimizer- and target-optimization-token caps. This system-level ladder asks how much of the improvement procedure the optimizer can carry for itself.

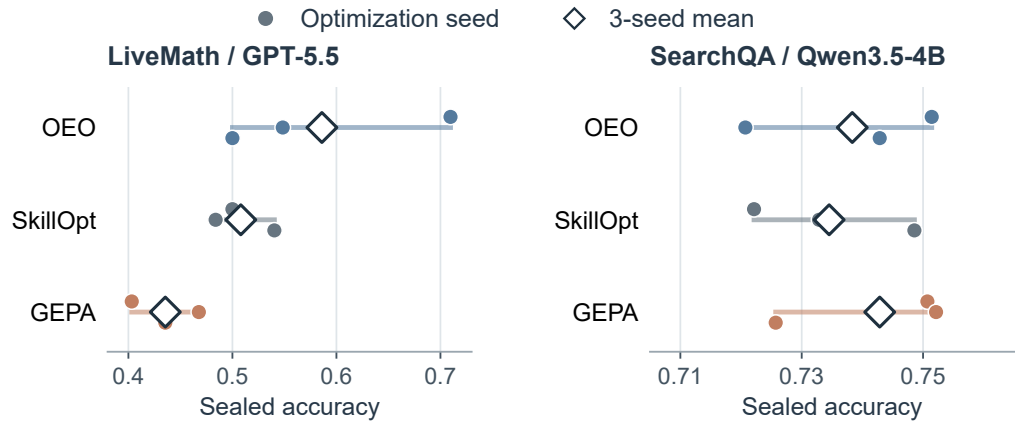

Figure 3: **Run-to-run variation in two frontier anchors.** Each circle is one optimization seed (42–44); diamonds mark 3-seed means, and horizontal segments span observed runs rather than confidence intervals. All selected skills use final-evaluation seed 42. The LiveMath OEO seed-42 point is the harmonized reevaluation (88/124); Table 2 retains the historical canonical score (85/124). Exact counts appear in Appendix B.

Table 3: **Optimizer-capability ladder with Qwen3.5-4B fixed as the target.** Completed cells report sealed-split scores. "Blocked" means that the optimizer did not emit an executable OEO action under the fixed interface; it is not a performance score. Bold marks the higher score where both procedures complete. The medium comparison uses matched configured caps; Appendix E.4 reports paired inference and accounting.

Benchmark	Method	Weak Qwen3.5-4B	Medium Qwen3.5-27B	Frontier GPT-5.5
LiveMath	OEO	*blocked*	**0.5968**	**0.6532**
	SKILLOPT	0.5565	0.5887	0.6129
SearchQA	OEO	*blocked*	0.6743	**0.7514**
	SKILLOPT	0.6579	**0.7050**	0.7221

Table 3 shows how the role of prescription changes with the optimizer. The weak optimizer cannot operate the delegated loop under the unchanged interface, while SKILLOPT remains executable because the framework supplies the control sequence. At medium capability, OEO can run, but the outcome depends on the task: it reaches parity on LiveMath, whereas SKILLOPT has a resolved advantage on SearchQA. The SearchQA contrast also exposes a procedural difference: SKILLOPT gathers evidence through 2,880 target calls, compared with 155 for OEO; on LiveMath, the corresponding counts are much closer (256 versus 205). Prescription can thus shape how thoroughly the optimizer explores, not only how it edits. At the frontier, OEO is higher on both tasks.

Together, these probes suggest a shift in how much procedural responsibility the optimizer can carry: the weak model needs the framework to execute, the medium model can act independently but does not always produce the better-performing skill, and the frontier model is competitive without the prescribed sequence. The question is therefore not whether structure is intrinsically good or bad, but whether the optimizer can organize the evidence–revision–selection loop that the framework would otherwise encode.

3.3 A SINGLE ZERO-INTERACTION REWRITE DOES NOT EXPLAIN THE GAINS

The frontier result raises a simpler alternative: perhaps GPT-5.5 already knows a better skill and interaction is unnecessary. Table 4 tests this by giving the model the same static starting information but no examples, trajectories, scores, or opportunity to revise again.

Table 4: **Static-input-matched one-shot control.** Each cell uses one GPT-5.5 rewrite and zero target-model optimization rollouts. The final column reports OEO minus the rewrite in percentage points; bold marks the highest score in each row.

Benchmark	Target	Initial	One-shot rewrite	OEO	Δ pp
SearchQA	Qwen3.5-4B	0.6693	0.6900	**0.7514**	+6.14
SearchQA	GPT-5.5	0.7857	0.8357	**0.8714**	+3.57
LiveMath	Qwen3.5-4B	0.3065	0.2742	**0.6532**	+37.90
LiveMath	GPT-5.5	0.3871	0.3468	**0.6855**	+33.87

The static rewrite helps on SearchQA but does not reach OEO; on LiveMath it moves in the wrong direction. Across all 4 cells, OEO remains ahead under paired analysis (Appendix C). The gain is therefore not explained by asking a frontier model to write a better skill once. OEO instead combines evidence gathering, repeated revision, and selection in an interactive optimization loop.

4 What Does Procedural Prescription Change?

The frontier comparison shows that OEO can be competitive without either tested prescribed procedure. We next use the fully instrumented OEO–SkillOpt pair to ask what prescription changes inside the optimization process. We compare committed skill paths and selected-skill behavior; Appendix E.5 gives the reconstruction and metric definitions.

4.1 Prescription reshapes committed paths

Figure 4 shows that the clearest difference is the size of the largest revisions, not how many revisions occur. The methods have no consistent ordering in committed-update count. Yet across all 8 instrumented settings, OEO makes a larger maximum edit, changes more sections at once, and has higher revision churn. In this pair, SkillOpt's prescribed procedure acts as a search-path constraint: it favors narrower revisions, whereas OEO can reorganize the skill within the same external contract.

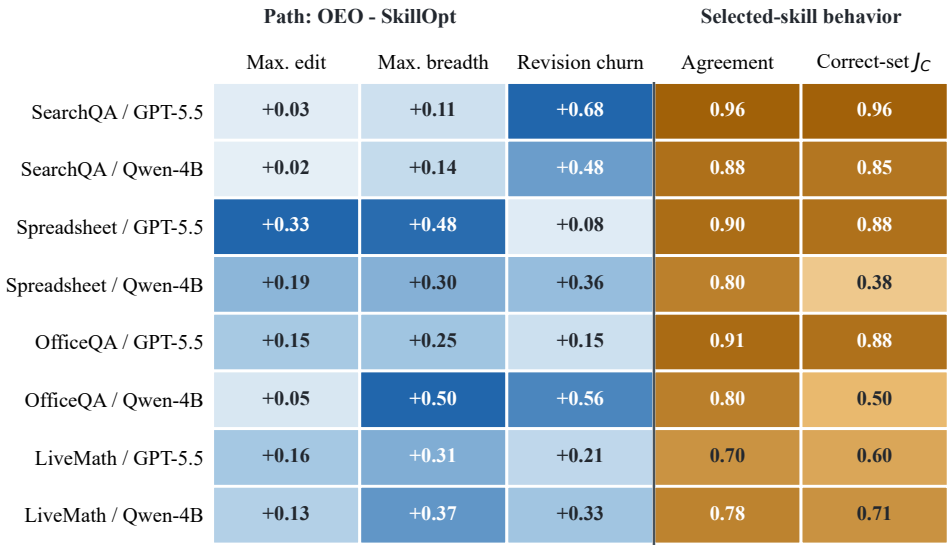

Figure 4: **Path divergence does not imply divergent evaluated behavior.** The first 3 columns report OEO minus SkillOpt committed-path metrics; all differences are positive. The final columns show pass/fail agreement and correct-set Jaccard (J_C) between selected skills. Color intensity is normalized within each column; annotations give the measured values.

4.2 PATH DIVERGENCE NEED NOT IMPLY BEHAVIORAL DIVERGENCE

The behavioral columns of Figure 4 do not mirror the path differences. On SearchQA and GPT-5.5-target SpreadsheetBench, the selected skills solve largely the same examples despite different edit histories. But similar scores can also conceal different correct-item sets. With the Qwen3.5-4B target, the OEO–SKILLOPT score gaps on SearchQA and OfficeQA are close (2.93 versus 2.33 percentage points), while correct-set Jaccard falls from 0.85 to 0.50. LiveMath/GPT-5.5 shows a larger score gap and less overlap. Across the 8 settings, final skill-text distance and item-level behavior distance are only weakly associated (Spearman $\rho = 0.31$). The document, its optimization path, and its behavior on evaluated items are therefore different objects of analysis; Appendix Table 11 gives the full measurements.

This distinction clarifies the role of prescription in the instrumented pair. It reliably shapes the path through skill space, but its effect on selected item-level behavior varies by task. Different paths can reach strongly overlapping correct-item sets; conversely, nearby aggregate scores can hide different sets. The value of a prescribed procedure lies in what that path delivers—performance, repeatability, evidence coverage, or operational control—not in producing a visibly different document.

5 DISCUSSION

Prescription supplies a conditional search prior. The results place prescription on a capability spectrum. At weak optimizer capability, the framework can make the loop executable under the tested interface. At medium capability, the optimizer can act independently, but on SearchQA the prescribed schedule gathers substantially more evidence and performs better. At the frontier, OEO is competitive without either tested procedure; nevertheless, GEPA leads on SpreadsheetBench, where its full-budget evolutionary search explores more extensively. Repeated runs also show that the ranking can change with the optimization trajectory. Prescription is therefore a useful scaffold when it improves execution, evidence collection, or reliability, rather than a universal prerequisite for skill improvement.

Delegate the procedure, not the contract. Letting the optimizer compose a procedure does not surrender control of the problem: the framework still fixes objectives, permissions, budgets, data boundaries, and evaluation. A natural design sequence is to specify this contract and measurable feedback first, then add stages or search rules where they demonstrably improve the outcome. The one-shot control distinguishes this design from simply asking a frontier model for a better skill: OEO organizes interaction, revision, and selection over time. Evaluating such systems requires more than a final score. Paired items show what the selected skills solve, repeated runs show how reliably they are reached, and path analysis shows how the procedure steers the search. A different-looking skill or trajectory alone does not establish a different capability.

6 RELATED WORK

Persistent artifacts and adaptive agents. Many self-evolving agents preserve experience in artifacts rather than changing model weights. Reflexion stores verbal feedback, ExpeL extracts trajectory insights, and Voyager grows an executable skill library (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2023). Recent work studies how skills are curated, revised, and audited (Ouyang et al., 2026; Shang & Yang, 2026; Gao et al., 2026); metacognitive and agent-evolution work also asks whether an agent can adapt its own learning process (Liu & van der Schaar, 2025; Zhang et al., 2026). Artifact persistence, however, does not determine who controls the procedure that turns experience into the next artifact. That allocation of control is our focus.

Model-generated proposals versus optimizer-composed procedures. OPRO, Promptbreeder, and TextGrad use language models to generate or revise candidates within optimization loops; ADAS and AFlow similarly use models in searches over agent designs (Yang et al., 2024; Fernando et al., 2023; Yuksekgonul et al., 2024; Hu et al., 2024; Zhang et al., 2024). The model contributes proposals, while an outer procedure still organizes evaluation and candidate selection. Bayesian-Agent, SKILLOPT, and GEPA encode different such procedures: posterior-guided operations, staged bounded editing, and Pareto evolutionary search (Wu et al., 2026; Yang et al., 2026; Agrawal et al., 2026). Other

systems vary curation, verification, and rollback (Shen et al., 2026; Ouyang et al., 2026; Shang & Yang, 2026; Gao et al., 2026). Our distinction is not whether a model writes revisions—it often does—but whether the framework prescribes the task-specific procedure around those revisions. OEO delegates that procedure under a fixed contract, and SKILLOPT and GEPA provide two complementary points of comparison.

7 CONCLUSION

Self-evolution needs a governed optimization problem, but a capable model need not inherit every step of the procedure for solving it. Across our tested settings, OEO composes the improvement procedure online and achieves competitive outcomes, while prescribed search remains valuable on particular tasks and at particular capability levels. The resulting design principle is contract first: govern the objective and evidence externally, let capable optimizers organize improvement, and prescribe the procedure when doing so adds value.

AI USE

Generative AI tools assisted with refining the research framing and hypotheses, designing experiments, implementing methods, translating manuscript text, qualitatively analyzing optimization trajectories, and interpreting results. The authors reviewed the AI-assisted work, verified the reported results, and take responsibility for the final paper.

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. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2507.19457.

Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Guney, Volkan Cirik, and Kyunghyun Cho. SearchQA: A new Q&A dataset augmented with context from a search engine. arXiv:1704.05179, 2017.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. arXiv:2309.16797, 2023.

Haowen Gao, Haoran Chen, Can Wang, Shasha Guo, Liang Pang, Zhaoyang Liu, Huawei Shen, and Xueqi Cheng. SkillAudit: Ground-truth-free skill evolution via paired trajectory auditing. arXiv:2606.14239, 2026.

Linyang He, Qiyao Yu, Hanze Dong, Baohao Liao, Xinxing Xu, Micah Goldblum, Jiang Bian, and Nima Mesgarani. LiveMathematicianBench: A live benchmark for mathematician-level reasoning with proof sketches. arXiv:2604.01754, 2026.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. arXiv:2408.08435, 2024.

Tennison Liu and Mihaela van der Schaar. Position: Truly self-improving agents require intrinsic metacognitive learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 81714–81727, 2025. URL https://proceedings.mlr.press/v267/liu25cw.html.

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. arXiv:2406.14991, 2024.

OpenAI. GPT-5.5 system card. Technical report, 2026. URL https://openai.com/index/gpt-5-5-system-card/.

Krista Opsahl-Ong, Arnav Singhvi, Jasmine Collins, Ivan Zhou, Cindy Wang, Ashutosh Baheti, Owen Oertell, Jacob Portes, Sam Havens, Erich Elsen, Michael Bendersky, Matei Zaharia, and Xing Chen. OfficeQA Pro: An enterprise benchmark for end-to-end grounded reasoning. arXiv:2603.08655, 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:2605.06614, 2026.

Qwen Team. Qwen3.5-27B model card. Official model card, 2026a. URL https://huggingface.co/Qwen/Qwen3.5-27B.

Qwen Team. Qwen3.5-4B model card. Official model card, 2026b. URL https://huggingface.co/Qwen/Qwen3.5-4B.

Fangxin Shang and Yehui Yang. Hypothesis-driven skill optimization for LLM agents. arXiv:2606.22330, 2026.

Yifei Shen, Bo Li, and Xinjie Zhang. SkillOpt-Lite: Better and faster agent self-evolution via one line of vibe. arXiv:2607.03451, 2026.

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 36*, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.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. arXiv:2305.16291, 2023.

Xiaojun Wu, Cehao Yang, Honghao Liu, Xueyuan Lin, Wenjie Zhang, Zhichao Shi, Xuhui Jiang, Chengjin Xu, Jia Li, and Jian Guo. Bayesian-Agent: Posterior-guided skill evolution across LLM agent harnesses. arXiv:2606.08348, 2026.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, 2024.

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:2605.23904, 2026.

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

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. arXiv:2410.10762, 2024.

Jiayi Zhang, Yongfeng Gu, Jianhao Ruan, Maojia Song, Yiran Peng, Zhiguang Han, Jinyu Xiang, Zhitao Wang, Caiyin Yang, Yixi Ouyang, Bang Liu, Chenglin Wu, and Yuyu Luo. Harnessing agentic evolution. arXiv:2605.13821, 2026.

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

REPRODUCIBILITY STATEMENT

The supplementary material provides optimization contracts, prompts and action schemas, per-item outcomes, audited resource ledgers, deterministic analysis scripts, the fixed bootstrap seed, and artifact hashes. It also identifies which initial and selected skills and run manifests are available, and inventories any unavailable material. Appendix E specifies the evaluation, accounting, and trajectory protocols.

A FULL PAIRED ITEM-LEVEL RESULTS

Tables 5 and 6 report the complete paired analysis underlying the aggregate comparison. Effects are percentage-point differences in pass rate. Confidence intervals use 100,000 shared-index bootstrap resamples with random seed 20260831; exact p-values are two-sided McNemar tests; and Holm values are corrected within the stated comparison family. The intervals condition on one selected artifact per optimization run and therefore do not quantify optimization-run stochasticity.

Table 5: **Paired** OEO–SKILLOPT **results.** Δn is the exact net difference in correct items.

Setting	Δ pp	95% CI	Δn	Exact p	Holm p
LiveMath / GPT-5.5	+18.55	$[+9.68, +27.42]$	+23	0.000191	0.00153
LiveMath / Qwen3.5-4B	+4.03	$[-4.03, +12.10]$	+5	0.442	1.000
OfficeQA / GPT-5.5	+2.33	$[-2.33, +6.98]$	+4	0.454	1.000
OfficeQA / Qwen3.5-4B	+2.33	$[-4.07, +8.72]$	+4	0.608	1.000
SearchQA / GPT-5.5	+0.57	$[-0.43, +1.57]$	+8	0.332	1.000
SearchQA / Qwen3.5-4B	+2.93	$[+1.14, +4.79]$	+41	0.00200	0.0140
SpreadsheetBench / GPT-5.5	+0.00	$[-3.57, +3.57]$	0	1.000	1.000
SpreadsheetBench / Qwen3.5-4B	+5.00	$[-0.36, +10.36]$	+14	0.0814	0.489

Table 6: **Paired** OEO–GEPA **results across the complete 8-setting grid.** Δn is the exact net difference in correct items.

Setting	Δ pp	95% CI	Δn	Exact p	Holm p
LiveMath / GPT-5.5	+28.23	$[+18.55, +37.90]$	+35	1.77×10^{-7}	1.42×10^{-6}
LiveMath / Qwen3.5-4B	+3.23	$[-3.23, +9.68]$	+4	0.481	1.000
OfficeQA / GPT-5.5	+2.91	$[-1.74, +7.56]$	+5	0.332	1.000
OfficeQA / Qwen3.5-4B	+5.81	$[-1.16, +12.79]$	+10	0.154	0.769
SearchQA / GPT-5.5	-0.21	$[-1.21, +0.79]$	-3	0.784	1.000
SearchQA / Qwen3.5-4B	+0.07	$[-1.57, +1.71]$	+1	1.000	1.000
SpreadsheetBench / GPT-5.5	-4.29	$[-7.86, -0.71]$	-12	0.0290	0.203
SpreadsheetBench / Qwen3.5-4B	-5.71	$[-11.79, +0.00]$	-16	0.0764	0.458

The full-envelope SpreadsheetBench runs reverse the earlier checkpoint-level diagnostic ordering: GEPA leads OEO by 12 items for the GPT-5.5 target and 16 for the Qwen3.5-4B target. The GPT-5.5-target paired interval excludes zero before family correction, but neither SpreadsheetBench comparison remains significant after Holm correction over all 8 OEO–GEPA cells. Relative to the common initial skill, OEO has positive final-score point estimates in all 8 settings and GEPA in 7; 6 of OEO's paired intervals exclude zero.

B FRONTIER-ANCHOR MULTI-SEED REPLICATION

We selected 2 anchors before running the replications: LiveMath with a GPT-5.5 target, where the canonical paired analysis showed the largest corrected OEO advantage, and SearchQA with a Qwen3.5-4B target, where it showed the other corrected OEO–SKILLOPT advantage. For each anchor and each of OEO, SKILLOPT, and GEPA, the historical seed-42 artifact is combined with new optimization seeds 43 and 44. All 12 planned new runs completed. The optimization seed changes the permitted optimization-data ordering and stochastic model trajectory; final evaluation uses the same frozen split and sampling seed 42 for every artifact.

Table 7: **Complete frontier-anchor multi-seed results.** Parentheses give exact correct-item counts. The last column reports mean and sample standard deviation across optimization seeds.

Setting	Method	Seed 42	Seed 43	Seed 44	Mean ± SD
LiveMath / GPT-5.5 target	OEO	0.7097 (88)	0.5484 (68)	0.5000 (62)	0.5860 ± 0.1098
	SKILLOPT	0.5000 (62)	0.4839 (60)	0.5403 (67)	0.5081 ± 0.0291
	GEPA	0.4032 (50)	0.4677 (58)	0.4355 (54)	0.4355 ± 0.0323
SearchQA / Qwen3.5-4B target	OEO	0.7514 (1052)	0.7207 (1009)	0.7429 (1040)	0.7383 ± 0.0158
	SKILLOPT	0.7221 (1011)	0.7329 (1026)	0.7486 (1048)	0.7345 ± 0.0133
	GEPA	0.7507 (1051)	0.7521 (1053)	0.7257 (1016)	0.7429 ± 0.0149

Table 7 reports every score and exact correct-item count. The runs reuse each method's canonical frontier configuration and resource envelope. Infrastructure-affected attempts are excluded before aggregation: they are never substituted for low scores, and only clean runs with complete unique final item IDs enter the table. The LiveMath seed-42 OEO artifact is re-evaluated under the common final-evaluation protocol, producing 88 rather than the 85 correct items recorded by the historical canonical evaluation. We retain the historical value in Table 2 and use the harmonized value only in the multi-seed block.

The replication distinguishes evaluation-item uncertainty from optimization-run variation. The paired intervals in Appendix A condition on one selected artifact; the sample standard deviations in Table 7 measure variation across 3 optimization trajectories. With only 3 seeds per cell, we report means, sample standard deviations, exact counts, and matched-seed win patterns rather than fit a general scaling law or claim a definitive method ranking.

C PAIRED ANALYSIS OF THE ONE-SHOT CONTROL

Table 8 compares OEO with the static-input-matched one-shot artifact on the same sealed items. All 4 intervals exclude zero, and every exact McNemar test remains significant after Holm correction over the 4-comparison family. The result strengthens the aggregate control: one prior-driven rewrite does not reproduce OEO's selected behavior.

Table 8: **Paired** OEO**–one-shot results.** Δn is the exact net difference in correct items; discordant counts are OEO-only / one-shot-only correct.

Setting	Δ pp	95% CI	Δn	Discordant	Exact p	Holm p
SearchQA / Qwen3.5-4B	+6.14	$[+4.29, +8.07]$	+86	137 / 51	2.78×10^{-10}	8.34×10^{-10}
SearchQA / GPT-5.5	+3.57	$[+2.36, +4.79]$	+50	64 / 14	8.58×10^{-9}	1.51×10^{-8}
LiveMath / Qwen3.5-4B	+37.90	$[+26.61, +49.19]$	+47	58 / 11	7.56×10^{-9}	1.51×10^{-8}
LiveMath / GPT-5.5	+33.87	$[+25.00, +42.74]$	+42	45 / 3	1.31×10^{-10}	5.25×10^{-10}

D SELECTED-SKILL LENGTH AND DEPLOYMENT ACCOUNTING

We measure every selected skill in the original deployment audit with the Qwen/Qwen3-8B tokenizer (Hugging Face `AutoTokenizer.encode`, `add_special_tokens=True`) to obtain a common length proxy across target models. Table 9 shows that OEO does not gain its optimization-time interaction advantage by producing uniformly shorter artifacts. Its selected skill is longer in 7 of 8 paired cells with SKILLOPT (median paired ratio $1.34\times$) and 5 of the 6 audited cells with GEPA (median $1.78\times$). The two later full-envelope SpreadsheetBench GEPA artifacts are not included in this length audit. Exact sealed-evaluation input/output totals are recoverable for all audited OEO and GEPA cells, but not for historical SKILLOPT runs because their target-token aggregate mixes optimization and final evaluation. We therefore report skill length as a deployment-overhead proxy, not a provider-billed token count or a three-method deployment-cost ranking.

Table 9: **Selected-skill length audit.** Token counts use the same offline tokenizer for all artifacts and are not provider-billed token counts. Final-evaluation coverage reports cells with an exactly separated target-token total.

Method	Cells	Proxy-token range	Median	Exact final-eval coverage
OEO	8	1,415–4,987	3,114.0	8/8
SKILLOPT	8	1,429–4,255	2,282.5	0/8
GEPA	6	1,016–2,739	1,958.0	6/6

E DETAILED EXPERIMENTAL PROTOCOL

Algorithm 1 Contract-enforced Open-Ended Optimization

Require: contract $\mathcal{C}$, initial skill s_0, optimizer M_{opt}; sealed split D_{final} held by the runner
$s \leftarrow s_0$; history $H \leftarrow \emptyset$
while the budget admits another optimizer step **do**
  present $(s, H, \text{remaining}(B))$ to M_{opt} and receive request q
  validate the operations in q against $\mathcal{C}$
  execute requested inspections, artifact reads, and target rollouts
  if q contains WRITESKILL(s') **then**
    $s \leftarrow s'$; persist s
  end if
  append q, returned evidence, and cost to H
  if q contains FINISH **then**
    break
  end if
end while
return s; evaluate once on D_{final}

E.1 BENCHMARKS, SPLITS, AND EVALUATION

Optimization data supply task evidence, and the permitted selection split—usually named `valid_seen`—may guide candidate or checkpoint selection. This selection-data boundary specifies which evidence can guide the returned skill. The sealed `valid_unseen` or test split is used only after selection and supplies the primary pass-rate metric under each benchmark's frozen evaluator. Unless a result is explicitly aggregated across seeds, each numerical score corresponds to one completed optimization run under the recorded configuration; blocked cells contain no score. The sealed evaluations contain SearchQA ($n = 1{,}400$), SpreadsheetBench ($n = 280$), OfficeQA ($n = 172$), and LiveMath ($n = 124$) examples. The OfficeQA items come from OfficeQA Full, which combines the 133 Pro questions with 113 easier questions, rather than from the Pro-only subset (Opsahl-Ong et al., 2026). Within a benchmark–target pair, all compared methods use the same dataset snapshot and split, target endpoint, initial skill, and evaluator.

E.2 FRONTIER COMPARISON AND RESOURCE ACCOUNTING

The primary experiment uses GPT-5.5 as the optimizer-side model for all three procedures. Each comparison contains the same 8 benchmark–target-model settings. The aligned GEPA condition evolves only the common persistent skill, starts from the byte-identical artifact, disables optional merging, and returns one validation-selected candidate. All 8 GEPA rows now use the native full-envelope procedure; the SpreadsheetBench runs replace earlier post-hoc checkpoint diagnostics.

We record optimizer tokens separately from target-model tokens spent during optimization and checkpoint-selection rollouts. SKILLOPT's configured allocations define the common reference budget, and OEO and GEPA report realized use relative to it. Model calls are atomic, so a call that begins within the budget can produce a small overshoot when it completes. The tables therefore compare sealed performance and realized token use separately rather than claiming exact realized-cost matching.

E.3 ONE-SHOT CONTROL

The zero-interaction rewrite receives the byte-identical initial skill and the same static step-0 information as OEO: benchmark and target identity, task and interface specification, objective, budget fields, evaluator and output requirements, and tool-availability description. Tool access is then disabled. It cannot inspect examples, execute the target, observe trajectories or scores, or revise more than once. Each cell contains one GPT-5.5 completion and zero target-model optimization rollouts.

E.4 OPTIMIZER-CAPABILITY CONTROLS

The main-text capability ladder combines two pre-specified controls with the aligned frontier rows. P1-W tested whether a Qwen3.5-4B optimizer could operate OEO after one benchmark-agnostic, syntax-only repair. P1-M compared Qwen3.5-27B-driven OEO and SKILLOPT under identical configured caps on LiveMath and SearchQA. The GPT-5.5 rows reuse the canonical frontier runs with the same Qwen3.5-4B target.

P1-W remained operationally blocked: neither the original response nor one syntax-only repair produced an executable action on either task. LiveMath and SearchQA each used 2 optimizer calls (3,274 and 14,110 tokens), with no target call, skill revision, or score. Under the same weak optimizer, SKILLOPT completed with scores of 0.5565 and 0.6579. The contrast establishes an executability boundary under this interface, not a comparison of downstream optimization quality.

Table 10: **Complete P1-M optimization accounting.** Scores and realized optimization use under shared configured caps, with Qwen3.5-27B as optimizer and Qwen3.5-4B as target.

Benchmark	Method	Correct	Score	Opt. calls	Opt. tokens / cap	Target calls	Target-opt. tokens / cap
LiveMath	OEO	74/124	0.5968	32	347,724/365,066	205	284,952/767,168
LiveMath	SKILLOPT	73/124	0.5887	29	408,198/365,066	256	288,354/767,168
SearchQA	OEO	944/1,400	0.6743	40	384,481/1,191,110	155	372,832/23,291,112
SearchQA	SKILLOPT	987/1,400	0.7050	112	1,199,906/1,191,110	2,880	5,650,257/23,291,112

All successful optimizer calls are charged, including LiveMath OEO's 2 format-only repairs. No accepted P1-M run has a failed provider retry, unknown usage, or context rejection; SKILLOPT's bounded overshoots arise within native actions admitted below the cap.

Sealed evaluation is excluded above: LiveMath OEO/SKILLOPT use 124/248 calls and 218,980/296,369 target tokens; SearchQA uses 1,400/2,800 calls and 4,051,046/5,231,219 tokens. SKILLOPT totals include initial-baseline calls.

Using the primary analysis's $100{,}000$-replicate paired bootstrap and exact McNemar test, LiveMath has 18/17 OEO-only/SKILLOPT-only correct items ($\Delta = +0.81$ points; 95% CI $[-8.87, +10.48]$; $p = 1.0000$). SearchQA has 76/119 ($\Delta = -3.07$; CI $[-5.00, -1.14]$; $p = 0.0025$, or 0.0051 after Holm correction across the 2 comparisons).

Table 11: **Complete final item-level correctness overlap in the instrumented** OEO–SKILLOPT **pair.** Agreement is binary pass/fail agreement; J_C is correct-set Jaccard; J_G is gain-set Jaccard relative to the shared initial skill.

Benchmark	Target	OEO	SKILLOPT	Agreement	J_C	J_G
SearchQA	Qwen3.5-4B	0.7514	0.7221	0.8793	0.8486	0.5567
SearchQA	GPT-5.5	0.8714	0.8657	0.9629	0.9581	0.7500
SpreadsheetBench	Qwen3.5-4B	0.2464	0.1964	0.8000	0.3778	0.2653
SpreadsheetBench	GPT-5.5	0.7607	0.7607	0.9000	0.8767	0.8136
OfficeQA	Qwen3.5-4B	0.3081	0.2849	0.8023	0.5000	0.3000
OfficeQA	GPT-5.5	0.7267	0.7035	0.9070	0.8779	0.7273
LiveMath	Qwen3.5-4B	0.6532	0.6129	0.7823	0.7065	0.7500
LiveMath	GPT-5.5	0.6855	0.5000	0.7016	0.5978	0.3617

E.5 TRAJECTORY AND ITEM-LEVEL DIAGNOSTICS

For each run, we reconstruct the initial skill s_0 and every byte-distinct committed state in chronological order, ending at s_T. Let $\mathrm{Lev}(s, s')$ denote token-level Levenshtein distance. The normalized edit between adjacent

states is $\mathrm{Lev}(s_{t-1}, s_t)/\max(|s_{t-1}|, |s_t|, 1)$. A Markdown section is indexed by its full heading path; section breadth is the fraction of paths added, removed, or changed between adjacent states. We report the maximum of each measure along the run. Let $L = \sum_{t=1}^{T} \mathrm{Lev}(s_{t-1}, s_t)$. Revision churn is

$$\mathrm{Churn} = \begin{cases} 0, & L = 0, \\ 1 - \dfrac{\mathrm{Lev}(s_0, s_T)}{L}, & L > 0, \end{cases} \tag{1}$$

which measures how much stepwise editing does not survive in the terminal state. Path metrics use chronological states; final-text and behavioral metrics use the selected skills $s_O^\star$ and $s_S^\star$. Their normalized text distance is $d_{\mathrm{text}} = \mathrm{Lev}(s_O^\star, s_S^\star)/\max(|s_O^\star|, |s_S^\star|, 1)$. Pass/fail agreement is the fraction of sealed items with identical binary correctness, and $d_{\mathrm{behavior}} = 1 -$ agreement. For correct-item sets C_0, C_O, and C_S, define gains $G_O = C_O \setminus C_0$ and $G_S = C_S \setminus C_0$. We report $J_C = |C_O \cap C_S|/|C_O \cup C_S|$ and $J_G = |G_O \cap G_S|/|G_O \cup G_S|$; empty–empty Jaccard is one. Rejected and uncommitted proposals are excluded.