TRIEVO: THREE-AGENT EVOLUTION OF POLICIES AND OPTIMIZERS

Anonymous authors
Paper under double-blind review

ABSTRACT

An adaptive agent can appear to improve because search keeps a lucky component, even if its optimizer does not improve. We ask whether an optimizer revision is worth retaining: old and revised instructions learn from the same parent, their descendants freeze before a fresh gate opens, and both branches are charged. Across nine Skill source runs, gated TriEvo has a +2.53-percentage-point mean difference from a duplicate-incumbent tournament (source-mean 95% interval, 1.83–3.23). After all Policy components are reset, retaining its gated optimizer state yields a +3.02-point difference from accept-all state (1.98–4.07); a GEPA-on-Evolver adapter has the same direction. The original Prompt/Skill/Memory panel has a +7.53-point macro difference, but Memory is negative. These endpoint comparisons motivate a prospective selection-and-reuse procedure. They do not establish superiority to the adapted GEPA author, universal transfer, or a cost advantage. Task-population generalization remains outside the reported aggregate analysis.

1 INTRODUCTION

A system that edits its own instructions can become a better task solver without becoming a better optimizer. Selection retains unusually successful edits; the optimizer then inherits their favorable starting point. Later success can therefore reflect the component it received rather than the procedure it learned. The scientific question is whether a revision improves *subsequent learning from a controlled parent*, and whether that advantage survives removal of the selected component.

Optimizer adaptation itself is established. Promptbreeder evolves mutation prompts; SkillOpt maintains optimizer-side meta guidance; MetaSkill-Evolve updates meta-skills on a slower timescale; Hyperagents evaluates transferable improvement ability (Fernando et al., 2023; Yang et al., 2026; Wang et al., 2026; Zhang et al., 2026). We study a narrower combination: same-parent prospective branch selection, explicit separation of authoring and gate information, full charging of rejected search, and optimizer-only reuse. TriEvo assigns these operations to three roles: Policy executes tasks, Evolver edits a Policy component, and a fixed Supervisor proposes changes to the Evolver instruction (Figure 1).

Our evaluation separates two questions. *Selection:* does testing optimizer revisions improve the retained state beyond ordinary branch selection? A duplicate-incumbent tournament uses the unchanged optimizer in both branches; gate-blind accept-all performs the same branch work but retains every valid revision. *Reuse:* does the retained instruction help when the Policy starts over? A reset discards all selected Policy components and keeps only the optimizer instruction. Applying the procedure to a GEPA-based revision author tests whether the selection pattern extends beyond one author. Nine Skill source runs support this central comparison; Prompt, Memory, composition, and transfer studies test its boundaries.

Scope of the evidence. We report success counts and aggregate costs for the completed experiments. The original three-seed carrier panel and the separate nine-source Skill panel answer different questions and are not pooled. Inference across source runs is conditional on the shared final task panel; task-population uncertainty requires the joint task-level outcomes. Historical development runs and published benchmark scores do not enter the local estimates.

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.

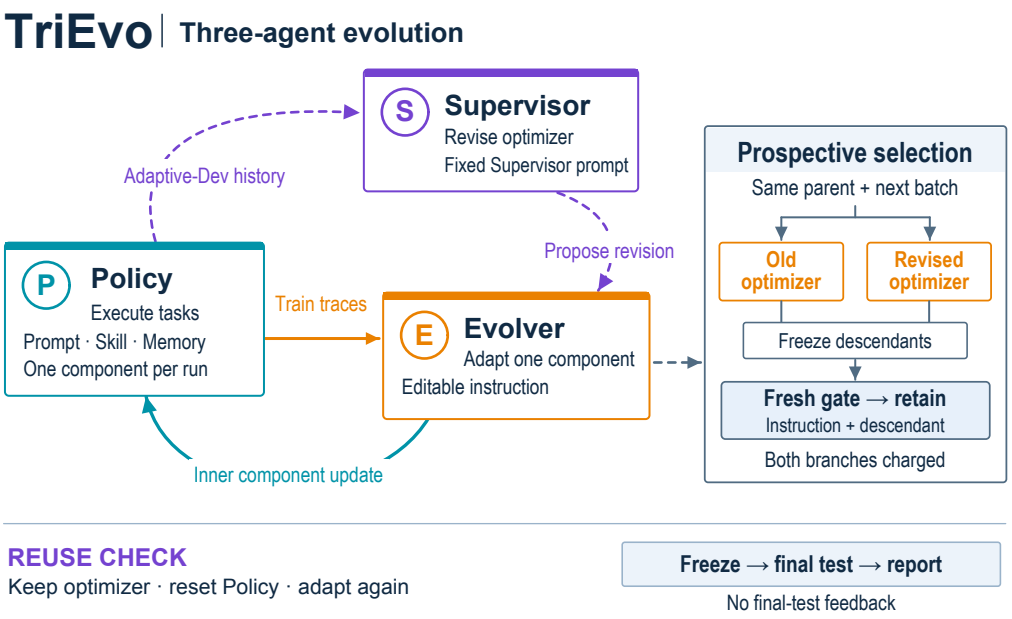

Figure 1: TriEvo in the single-component setting. Solid arrows carry training traces and inner component updates; purple dashed arrows carry adaptive-Dev evidence and optimizer proposals. The inset is a fixed selection procedure, not an agent: old and revised optimizers start from the same parent, descendants freeze before a fresh gate opens, and only a strict gain retains the revised instruction and its descendant. Otherwise, the old branch continues. Both branches are charged. Reset-and-reuse retains only optimizer state; final tests provide no feedback.

2 Selecting optimizer revisions prospectively

Let $z = (p, s, m)$ denote the Policy's Prompt, Skill, and Memory. The optimizer state e is the editable component-specific Evolver instruction; the shared Evolver base and the Supervisor instruction stay fixed. Policy weights, executor, tool permissions, and the Memory retriever stay frozen. The inner gate accepts a component edit only after a strict gain over its parent on matched adaptation replay, with at most 3 attempts per batch; a legal `no_change` differs from malformed output. The Supervisor reads cumulative adaptive-Dev evidence and may write only e. The Evolver writes one component, never the executor or scorer.

One checkpoint in plain terms. Give old and proposed Evolver instructions the same Skill, history, and next adaptation batch. After each produces a frozen descendant, reveal the fresh gate tasks. A strict gain retains the revised instruction and its descendant; otherwise the old branch continues with its own descendant. Gate transcripts enter neither author's inputs. The later reuse test discards both descendant Skills and compares the instructions alone.

Denote by $L_B(e, z, A, \xi)$ the descendant produced under search budget B using adaptation data A and randomness ξ. A useful optimizer revision should increase

$$V_B(e; \mathcal{P}) = \mathrm{E}_{(z,A,Q,\xi)\sim\mathcal{P}}\big[R_Q(L_B(e, z, A, \xi)) - R_Q(z)\big],$$

where Q never trains or selects the descendant. A local gate estimates branch utility at its current parent; reset-and-reuse probes whether the retained instruction helps on a new adaptation problem.

Algorithm: one outer opportunity. Freeze (z_b, e_b, h_b, e'_b), then clone the full parent into isolated sessions. Both branches use the same next adaptation batch A_{b+1}, budget, and inner limits. After both descendants freeze, reveal G_b. Keep (e'_b, z^n) only if it passes at least 1 additional gate task; otherwise keep (e_b, z^o). A tie preserves the old branch's valid inner progress. Archive the losing task/session/component state for audit only, then regenerate traces from the survivor.

Do not execute A_{b+1} again as a mainline step. Invalid or no-change outer outputs advance the incumbent. An infrastructure gate failure pauses the atomic decision; it never silently promotes a branch. Record scientific failures, rejected edits, and their costs. Reserve resources for both branches and gate evaluation before admitting an atomic block; author-call admission follows Appendix J.

Even equally good optimizer kernels can yield a positive selected gate difference through noise. The one-task margin is a search heuristic, not a confidence bound or a guarantee of population improvement. No gate statistic serves as final evidence.

3 PROTOCOL, INFORMATION, AND INFERENTIAL TARGET

Evaluation population. The primary panel is an AppWorld-style extension (Trivedi et al., 2024): 120 scenario families with 3 variants each, totaling 360 tasks. It is not an existing official hidden split. The protocol separates task authorship and final evaluation from method development. The construction protocol specifies checks of solvability, reference successes and failures, scorer sensitivity, and overlap in goals, API dependencies and reference solutions. Family disjointness is a requirement across adaptation, gates and final evaluation; distinct filenames alone do not establish it. The reported results concern this particular task population.

The training contract partitions the 366-task pool by family: 96 adaptation tasks and 270 gate tasks, with 36 separate adaptive-Dev tasks. The 11 adaptation batches comprise 10 batches of 9 tasks and a last batch of 6. Checkpoints from the second through penultimate batch offer 9 revisions. Each next-batch tournament opens a different 30-task, 10-family gate block. Historical test tasks are excluded from this fresh-panel evaluation.

Actor	Read access	Write/feedback boundary
Policy	Current task, active component	Trajectory only; no access to optimizer history
Evolver	Current-parent A traces, allowed rejection history, e	One component; no Dev or gate transcripts
Supervisor	Cumulative Dev index, allowed versions and decisions	Evolver rules only; no task answers in e
Gate runner	Frozen branch descendants and fresh G_b	Decision/state retention; no transcript feedback
Final runner	Frozen endpoints and independent final tasks	Scores to report only; no adaptation edge

Selection affects the retained state and is therefore indirectly observable in future history. The stronger restriction is that gate contents, answers, and transcripts never enter authors' inputs, and each gate family is consumed only once per trajectory. Same-seed arms use the same frozen task allocation and initial states. Common random streams are used where supported; otherwise requests are interleaved and the lack of deterministic service seeds is recorded.

Unit of analysis. For family f, average task-success differences over its variants and the fixed Prompt/Skill/Memory and seed levels, yielding d_f. The carrier macro contrast is $100\,\mathrm{mean}_f(d_f)$. We report the complete seed-level differences for the three-seed carrier panel. These are descriptive estimates; repeated tasks, component levels and shared endpoint aliases are not additional independent runs. Task-family confidence intervals cannot be recovered from aggregate endpoint counts and are not reported.

The carrier macro target is 5 percentage points. A separate Skill replication contains 9 source runs on training/Dev shards with a shared final panel (Section 4). Its four focal contrasts are TriEvo gate versus accept-all, TriEvo gate versus null-gate, and gated versus accept-all reset state for TriEvo and GEPA. The two cohorts remain separate. Reset episodes are nested within source; variants, components and aliases do not add source runs.

Which units support which claim? The carrier panel contains three original seed endpoints. The separate Skill panel contains nine source runs; reset episodes are nested within sources. Gate scores select states, while final panels report outcomes. Repeated final tasks remain shared across arms and sources.

4 PRIMARY MECHANISM TEST: WHAT DOES THE GATE ADD?

We first test the retention procedure on nine Skill source runs. A source pairs method arms on one training/Dev shard, initial state and run randomness. The cohort shares initial states, adaptation and gate tasks, and reserved resources across retention conditions. Within each arm, old and revised branches start from that arm's current parent; different retention decisions can make parents diverge across arms. Each source has a separate training/Dev shard under the same partition contract; all endpoints freeze before final access. The runs share a final task panel, so their variation does not measure uncertainty over newly authored final populations.

Accept-all and null controls. Accept-all executes both old/new branches and the gate, but masks gate scores and always retains the valid revised branch. Schema failure keeps the incumbent. Replication always reserves both branches; malformed or no-change instructions substitute the incumbent in the challenger slot. This differs from the original skip schedule. The assigned gate/accept-all difference is retention; later histories may diverge. Null-gate instead runs two stochastic branches with identical incumbent instructions. Null-blind uses a fair branch coin fixed before rollout and masks gate scores. Both null arms execute the same paid meta calls into an inaccessible discard buffer; neither can revise the instruction. This deliberate intervention controls allocation, not the most efficient use of its budget.

A strong editor in the same wrapper. GEPA-on-Evolver optimizes the instruction e, not the Policy Skill. It retains reflective mutation, Pareto parent sampling, population ancestry, and paired screening (Agrawal et al., 2025). Native GEPA merge combines changes across modules; with a single optimized instruction module, that crossover has no nontrivial multi-module role. Its evaluator runs the identical inner learner on permitted saved-parent episodes and returns Dev scores and allowed traces; TriEvo receives the same evaluator API and allowance. The archive never sees gate transcripts or final outcomes. The exported instruction faces the same external gate, and only that instruction survives reset. This is a single-module adapter of GEPA; Appendix I details its search and export contract.

Table 1: Primary mechanism test: the gate under matched branch work.

Paired source contrast	Mean pp [source 95% CI]	Holm p_{sign}
TriEvo: gate minus accept-all	+3.02 [+1.67, +4.38]	0.0156
TriEvo gate minus null gate	+2.53 [+1.83, +3.23]	0.0156
Null: gate minus blind retention	+1.48 [+1.03, +1.93]	–
GEPA: gate minus accept-all	+2.56 [+2.06, +3.06]	–
TriEvo gate minus GEPA gate	+0.15 [-0.59, +0.90]	–

Separate 9-source Skill cohort; one shared final task panel. Source t intervals condition on the shared final panel; they are marginal, not simultaneous. Exploratory Holm values cover the four stated focal contrasts; – denotes an additional comparison. The editor difference is unresolved.

The reported null-tournament contrast is +1.48 pp over blind retention. Ordinary selection could therefore explain part of this descriptive endpoint pattern, but its population effect is not established. Full revision exceeds null-gating by +2.53 pp in the reported totals. The GEPA gate contrast is +2.56 pp, and the TriEvo–GEPA difference is +0.15 pp. The shared direction supports studying retention across both revision authors. The contrasts do not establish editor superiority, and their gains are not additive.

Replication inference. The source run is the analysis unit. We average reset episodes within each source before comparing arms. Source-mean intervals and source sign tests condition on the shared final task panel; neither treats repeated final tasks as independent samples. The four focal contrasts form one analysis family. We interpret their p-values as exploratory, without a preregistered confirmatory claim. Each has 9 positive source differences: exact two-sided sign $p = 0.00390625$, or Holm-adjusted $p = 0.015625$. The sign test concerns direction/median, while the t interval concerns the mean; neither alone proves general task-population improvement. Task-family and crossed source/task uncertainty are different estimands and are not reported from the endpoint summaries (Appendices B/I).

5 OPTIMIZER-ONLY REUSE AFTER POLICY RESET

Freeze every terminal source instruction before exposing 3 independent reset adaptation shards per source. Each contains 36 tasks in 4 batches. Reset Prompt, Skill, Memory, rejection history, sessions, and retriever state to identical initial bytes. Run the same inner learner without outer updates or archive access. A separate 60-family final panel scores the frozen endpoints; episodes remain nested within 9 source runs.

Table 2: Primary reuse test: retain only optimizer instructions.

Reset contrast	Mean pp [source 95% CI]	Holm p_{sign}
TriEvo: gated minus accept-all state	+3.02 [+1.98, +4.07]	0.0156
GEPA: gated minus accept-all state	+3.09 [+2.71, +3.46]	0.0156
TriEvo gated minus GEPA gated state	-0.25 [-1.49, +1.00]	–
TriEvo gated minus fixed state	+6.17 [+5.23, +7.11]	–
TriEvo gated minus SkillOpt state	+2.41 [+1.53, +3.29]	–
TriEvo gated minus MetaSkill state	+0.62 [-0.62, +1.85]	–

The cohort contains 9 sources with 3 adaptation episodes nested within each, full Policy reset, and 60 final families separate from the primary panel; episodes are averaged within each source. Source effects and all 27 reset-episode count rows appear in Appendix D. Intervals condition on this final panel; exploratory Holm values cover the same four-contrast family as Table 1.

The reset totals differ by +3.02 pp for TriEvo gated versus accept-all state and +3.09 pp for GEPA. The intervention removes inherited Policy components and keeps source tournament topology fixed. These source-level contrasts concern reuse of the retained instruction; they do not certify every adopted revision or separate withholding from parent matching.

The TriEvo–GEPA and TriEvo–MetaSkill reset contrasts leave revision-author superiority unresolved. The original three-source reset results are reported separately in Appendix E.

6 COMPONENT RESULTS AND MATCHED LOCAL CONTROLS

Fixed retains the initial Evolver instruction while the chosen Policy component still adapts. Supervised additionally allows Supervisor revisions.

Table 3: Held-out effects vary sharply by component.

Carrier	Fixed	Supervised	Δ pp	101	202	303
Prompt	48.15	60.00	+11.85	+13.06	+11.39	+11.11
Skill	53.61	64.91	+11.30	+11.67	+10.56	+11.67
Memory	42.13	41.57	-0.56	+0.56	-1.67	-0.56
Macro	47.96	55.49	+7.53	+8.43	+6.76	+7.41

Success rates are percentages. Last columns show all endpoint seed effects (pp). Three source seeds on the same 360-task panel; task-family intervals are not reported.

Component-level outcomes. The original three-seed carrier panel provides a complementary component-level view. The macro difference exceeds the numerical practical target, with positive Prompt and Skill differences and a negative Memory difference. All three seed-level macro differences are positive. This does not establish a population effect or achievement of the target under uncertainty. These endpoints are separate from the nine-source mechanism study.

Skill is the main carrier for the mechanism comparison. Blind-meta has the same revision opportunities and gate but no Dev evidence; extra search reallocates outer-loop resources to fixed-guidance candidate generation and replay. SkillOpt-v2 retains its optimizer meta-skill and slow update, and MetaSkill-Evolve retains fast/slow meta optimization. Their smaller gaps are the relevant competitive test, not the larger contrast against cheap Fixed.

The common Skill ceiling is 12.20M search tokens and 1400 model calls, including losing branches and Dev. Reported central matched-control tokens differ by less than 0.50% within each seed,

Table 4: Matched local Skill controls.

Method	Success (%)	TriEvo minus method (pp)
Fixed	53.61	+11.30
Blind-meta	55.83	+9.07
Extra search	58.24	+6.67
SkillOpt-v2	60.28	+4.63
MetaSkill-Evolve	61.11	+3.80
TriEvo Skill	64.91	reference

Local matched ports; these are not published leaderboard scores. Fixed and blind-meta comparisons are descriptive.

and calls match exactly. Supervised uses 30.22% more tokens than cheap Fixed, calculated before rounding. The aggregate ledger is in Appendix G. These comparisons concern quality under a matched high budget; they do not establish efficiency or a break-even point.

The original gate ablation has a +3.89 pp difference but reallocates saved branch resources. It changes several factors together, which motivates the matched-work design above.

7 SCOPE: COMPOSITION, IN-DOMAIN LEARNING, AND TRANSFER

Table 5: Distinct exploratory questions, grouped by evaluation setting.

Paired contrast	Effect pp
PS minus S	+6.48
PSM minus PS	-2.59
Prompt minus GEPA	+1.57
ALFWorld: supervised minus fixed	+8.71
ALFWorld: supervised minus extra search	+5.22
ALFWorld: supervised minus RSEA	+3.23
AppWorld Skill to ALFWorld minus base	+3.48
AppWorld Skill to BFCL minus base	+1.81
AppWorld Memory to BFCL minus base	-2.08

Do not compare magnitudes across groups: AppWorld contains 120 task families; ALFWorld has 134 games; BFCL has 80 families / 240 tasks. Frozen transfer has no target adaptation.

Composition has a boundary. Assemble each seed's predetermined single-component endpoints without retraining or choosing components by final score. The PS mean is 71.39%, compared with 64.91% for Skill and 68.80% for PSM. PS has the higher reported mean; PSM is lower. PM and SM also fail to exceed their stronger constituent. These are paired comparisons of fixed assemblies, not a registered factorial interaction estimate; they do not separate content conflict from capacity, retrieval, or instruction-priority effects.

The Prompt comparison is uncertain. Prompt has a positive descriptive difference from Fixed, but its +1.57 pp difference from the GEPA port does not establish superiority. The native Skill versus flat carrier contrast is smaller still: +1.94 pp. Both forms must retain scripts and references under the content-preserving control; removing executable content would answer a different question. These comparisons remain descriptive.

In-domain totals exceed frozen-transfer totals. ALFWorld (Shridhar et al., 2020) uses 20 training games, 18 adaptive-Dev games, and all 134 valid-unseen games. Its training pool is partitioned into 12 authoring and 8 gate games, with 1 outer opportunity (Appendix E); scarce game identities limit fresh gating. The in-domain supervised mean is 73.88%, compared with 65.17% for Fixed and 68.66% for extra search. The descriptive direction persists against local RSEA, although this panel is exploratory and game identities may share templates.

Loading the frozen AppWorld Skill into ALFWorld gives only +3.48 pp relative to its target base. BFCL Skill transfer is uncertain, and Memory transfer has a negative mean. The BFCL panel is

a multi-turn family subset, not BFCL Overall Acc (Patil et al., 2025). It permits no target adaptation; incompatible or unused source artifacts remain outcomes. These patterns argue against treating portable optimizer utility, in-domain component learning, and cross-domain content transfer as interchangeable claims.

Appendices G/H give aggregate source, reset and local ALFWorld cost summaries, with final evaluation separate. ALFWorld remains a short, exploratory intervention with template dependence; it does not establish sustained optimizer learning. Composition, frozen transfer, development and dataset construction are not priced as a whole project. No efficiency frontier, amortization threshold, or cost-normalized composition claim is made.

8 Relation to prior work and limits of the claim

Feedback, execution and prompt optimization. ReAct interleaves reasoning and actions; Reflexion carries verbal feedback into subsequent trials; Self-Refine iteratively critiques and revises an output (Yao et al., 2022; Shinn et al., 2023; Madaan et al., 2023). These operations motivate learning from traces but do not by themselves test a reusable optimizer revision. APE proposes and selects task instructions, while OPRO conditions new candidates on previous solution–score pairs (Zhou et al., 2022; Yang et al., 2023). DSPy compiles declarative modules against a metric, and TextGrad propagates textual feedback through computation graphs (Khattab et al., 2023; Yuksekgonul et al., 2024). TriEvo focuses on whether to retain a change to the instruction that generates future component edits.

Searching agent programs and workflows. ADAS evaluates new agent programs using an archive, the Darwin Gödel Machine modifies coding-agent implementations through empirical search, and AFlow searches code-represented workflows with a Monte Carlo tree-search variant (Hu et al., 2024; Zhang et al., 2025a; 2024). These works permit broader changes than TriEvo's fixed executor and component interfaces. Our same-parent and reset controls isolate a narrower retention question; they do not establish superiority over open-ended program or workflow search.

Meta-optimization is the starting point. SkillOpt-v2's optimizer-only meta-skill closely matches the functional role of an Evolver instruction; its separate slow field also affects Policy behavior (Yang et al., 2026). MetaSkill-Evolve explicitly improves the skill-evolution process on a slower timescale (Wang et al., 2026). Promptbreeder's mutation prompts and hypermutation already make the proposal generator editable (Fernando et al., 2023). Hyperagents makes the meta-agent program editable and tests improvement ability after freezing it (Zhang et al., 2026). TPGO/GRAO retrieves and ranks optimization experience to condition later edits (He et al., 2026).

The distinguishing object here is the *decision to retain a revision*. It is tested from the same parent, on data not available to its author, at a cost that includes the losing branch. A subsequent reset removes the component selected alongside it. The combination defines an empirical procedure for selecting and testing reusable optimizer instructions. Appendix A maps writable state, evidence, selection timing, same-parent testing, reuse, and budget without inferring absent capabilities from missing descriptions.

RSEA evolves layered Policy context and separates current-state acceptance from strict best-state selection; ACE incrementally curates a playbook; GEPA uses reflective mutation and a Pareto population (Nguyen et al., 2026; Zhang et al., 2025b; Agrawal et al., 2025). The local adaptations target these semantics. Public scores are kept outside the local result panels. The comparison targets the declared local interfaces; it does not rank the native systems on their original benchmarks.

Scope of comparison. The central matched ports are necessary but do not exhaust strong alternatives. There is no matched Promptbreeder, Hyperagents, GRAO, or open-ended optimizer result. The same-wrapper GEPA adapter addresses the closest textual-interface alternative; it does not replace a same-wrapper native meta-skill study or a direct-Dev Policy editor. The contribution is a targeted retention procedure on a narrow Skill panel and one fresh task population, not general recursive self-improvement or a universal optimizer.

The original carrier panels still have only 3 seeds. The new 9-run Skill cohort varies complete source shards, but does not establish robustness across models, executors, arbitrary task authors, or carriers. Source-mean interval coverage with nine runs remains approximate. Adaptive Dev is

repeatedly consulted, the gate margin is heuristic, and ALFWorld may share templates. Candidate validity also improves in the diagnostic, so the evidence does not isolate better semantic edits from better valid-output behavior. Retained instructions may encode domain knowledge as well as an optimization strategy; Policy reset does not distinguish those explanations. Memory and transfer failures delimit scope rather than being excluded from the analysis.

Implementation and impact. Appendix J specifies model routing, runtime limits, initial components, authoring contracts and the reproduction sequence. Model and tool permissions stay fixed during learning. Executable Skill files require review; changes to text alone do not establish the safety of an evolving agent.

9 Conclusion

Same-parent prospective retention evaluates an optimizer revision through the subsequent learning of matched branches. The nine-source Skill panel has positive mean contrasts over both accept-all retention and a duplicate-incumbent tournament, and reset comparisons retain the same direction after Policy components are discarded. The GEPA adapter shows a similar selector pattern; the evidence does not establish a superior revision author. The original Memory and transfer results delimit scope. Together, paired selection and optimizer-only reset provide a concrete way to test whether a retained instruction helps subsequent adaptation after the successful task-solving component is discarded.

AI use statement

AI assistants supported methodological critique, workflow development, aggregate-result interpretation, editing, overview-figure design, image generation and code-based reconstruction, LaTeX formatting, reference checking, recomputation scripts and editorial review. Qoder-Agent supported the experimental workflow. The authors are responsible for the text, citations, analysis and experimental claims.

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/2507.19457v2.

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/2309.16797v1.

Shan He, Runze Wang, Zhuoyun Du, Huiyu Bai, Zouying Cao, Yu Cheng, and Bo Zheng. Learning to evolve: A self-improving framework for multi-agent systems via textual parameter graph optimization. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 30719–30732, 2026. URL https://aclanthology.org/2026.findings-acl.1534/.

Shengran Hu, Cong Lu, and Jeff Clune. Automated Design of Agentic Systems, 2024. URL https://arxiv.org/abs/2408.08435v2. arXiv:2408.08435v2.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines, 2023. URL https://arxiv.org/abs/2310.03714v1. arXiv:2310.03714v1.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative Refinement with Self-Feedback, 2023. URL https://arxiv.org/abs/2303.17651v2. arXiv:2303.17651v2.

Michael Nguyen, Quoc Nguyen, and Paul Vuong. Recursive self-evolving agents via held-out selection, 2026. URL https://arxiv.org/abs/2606.28374v1.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The Berkeley Function Calling Leaderboard (BFCL): From Tool Use to Agentic Evaluation of Large Language Models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning, 2023. URL https://arxiv.org/abs/2303.11366v4. arXiv:2303.11366v4.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning, 2020. URL https://arxiv.org/abs/2010.03768v2.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A Controllable World of Apps and People for Benchmarking Interactive Coding Agents, 2024. URL https://arxiv.org/abs/2407.18901v1.

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/2607.05297v1. arXiv:2607.05297v1.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large Language Models as Optimizers, 2023. URL https://arxiv.org/abs/2309.03409v3. arXiv:2309.03409v3.

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, 2026. URL https://arxiv.org/abs/2605.23904v2.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models, 2022. URL https://arxiv.org/abs/2210.03629v3. arXiv:2210.03629v3.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. TextGrad: Automatic "Differentiation" via Text, 2024. URL https://arxiv.org/abs/2406.07496v1. arXiv:2406.07496v1.

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/2505.22954v3. arXiv:2505.22954v3.

Jenny Zhang, Bingchen Zhao, Wannan Yang, Jakob Foerster, Jeff Clune, Minqi Jiang, Sam Devlin, and Tatiana Shavrina. Hyperagents, 2026. URL https://arxiv.org/abs/2603.19461v1.

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, 2024. URL https://arxiv.org/abs/2410.10762v4. arXiv:2410.10762v4.

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/2510.04618v3.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large Language Models Are Human-Level Prompt Engineers, 2022. URL https://arxiv.org/abs/2211.01910v2. arXiv:2211.01910v2.

Appendix guide. A compares method interfaces; B defines populations and analysis units; C–D give endpoint and source counts; E–F document domain and mechanism diagnostics; G–H give aggregate cost ledgers; I specifies source-level inference and the GEPA adapter; J gives execution settings and the reproduction sequence; K gives per-source cost ranges; L reproduces the initial authoring instructions.

A METHOD COMPARISON AND IMPLEMENTATION CONTRACT

Related-method interfaces. Selection and cost entries describe native methods rather than results under the local TriEvo budget. "Not established" indicates an operation not established in the cited method description, not a claim that the method cannot be extended to support it.

Method	Writable state; evidence	Selection, shared parent, reuse; budget
TriEvo	One Policy component and separate optimizer rules; current A traces, Dev to Supervisor only	Frozen descendants on fresh G; explicit same-parent tournament; optimizer-only reset; all branches charged
SkillOpt-v2	Skill, Policy slow field, optimizer meta-skill; train traces, selection and epoch comparisons	Skill selection gate; Algorithm 1 does not separately gate old/new meta instructions on prospective branches; optimizer-only reuse not established; rollouts plus meta work
MetaSkill-Evolve	Task skills and evolution meta-skills; fast-loop evidence and validation	Fast/slow updates and branch selection; TriEvo-equivalent gate/reuse not established in the cited method; both timescales must be charged
Promptbreeder	Task and mutation prompts; training fitness and lineage	Population tournament and hypermutation; downstream fitness selects prompt pairs; separate TriEvo gate/reuse not established; all fitness and mutation calls
Hyperagents	Task-agent and meta-agent programs; parent/archive evaluations	Archive and descendant selection, not the defined local gate; fixed-meta transfer studied with transferred task-agent initialization; generation, modification, and evaluation costs
TPGO/GRAO	Graph edits and optimizer experience memory; errors, targeted replay, ranked history	Effectiveness on source error clusters; no separate fresh optimizer gate in the cited method; reuse not established; diagnosis, retrieval, and replay costs
GEPA	Module prompts, population, ancestry; feedback and Pareto scores	Paired minibatch screen then population selection; reflective population retained in our optimizer-guidance adapter; external gate added explicitly, not a native claim; evaluation plus reflection; native multi-module merge does not apply to the single-module adapter
RSEA	Strategy, skills, playbook; evolve traces and validation	Working state allows ties, best requires improvement; optimizer gate/reuse not applicable to fixed rewrite rule; generations and rollouts
ACE	Playbook and bullet statistics; task traces and reflection	Delta curation, with offline/online distinction; not the defined optimizer gate; optimizer reuse not established; all roles and epochs

Comparison contract. The local ports share Policy capabilities, component interfaces, task states, scorers and permitted evidence classes. The methods differ in the state they update and how they select candidates. SkillOpt retains optimizer-side meta instructions and its direct Policy slow field; MetaSkill retains separate skill and meta-skill updates. RSEA and ACE update assembled external state. These are wider write surfaces than a single Skill component, so the corresponding system comparisons are reported separately. GEPA changes only the Evolver instruction in the primary

selector comparison; its population, screening and export rules are specified in Appendix I. Search costs include rejected proposals and evaluated losing branches. Tables report the stated trajectories rather than all method-development or tuning expenditure.

B Population construction, statistics, and source identity

Execution stack. Policy uses `qwen3.8-flash` with temperature 0 and thinking disabled. The frozen primary authoring route uses `gpt-5.6-sol` at medium reasoning effort for Evolver and Supervisor. These are separate model roles; the remote orchestration agent is not an additional experimental role. Appendix J gives the role-specific settings, including the registered Qwen alternatives.

Population structure. The primary panel has 120 families with three variants each; reset evaluation has 60 families and proposal diagnosis has 40. A family groups variants of a shared semantic goal and API structure. The construction protocol separates these families from adaptation and gate families, fixes initial worlds and scoring functions before evaluation, and checks reference successes, reference failures and scorer sensitivity. Historical development tasks do not contribute to the fresh final-panel estimates.

The AppWorld training partition allocates 32 families to adaptation and 90 to gates. The split is defined at family level. Candidate authors may repeatedly use adaptation data, but no candidate may be refined after gate reveal. Final task authors do not tune their population against method failures. The primary score is complete task success, not a scenario-consistency or subgoal score.

Analysis boundary. Endpoint totals determine mean success and source-level contrasts. They do not determine which individual tasks both arms solved, the task-family covariance, or a crossed source/task confidence interval. The reported source analysis conditions on a fixed final task panel. Generalization over new final populations requires the joint task-by-source outcome matrix. Shared endpoint aliases are references to the same endpoint, and reset episodes are nested within their source.

Missingness and failures. Healthy task failures count as zero. Schema errors consume attempts and cost; legal no-change outputs are distinct from malformed outputs. A retry is admitted only for a classified infrastructure failure, with the same task, state and configuration; a healthy completion is reused. Failed paid calls retain their costs. Unresolved outcomes and unknown usage remain missing rather than being imputed as successes or zero cost. All denominators in the endpoint tables refer to completed evaluation panels.

Endpoint identity. Each reported endpoint has a source label, arm and denominator. Reused endpoint aliases do not add replications. Historical development runs and literature scores are excluded from local means and contrasts. The appendix counts support arithmetic reproduction of the descriptive contrasts; they do not determine task-level pairings.

C Complete primary-population counts

Every endpoint below reports a denominator of 360 tasks / 120 families. Same-seed task pairing is defined by the protocol. These rows provide the exact counts behind the main rates and the secondary local controls; they are not extra samples beyond the endpoint registry.

SkillOpt-v2 and MetaSkill-Evolve are local, matched optimizer-adaptation ports under the stipulated protocol. RSEA and ACE instead occupy the assembled external-state comparison, so their scores cannot be subtracted from single-component Skill to isolate the Supervisor. PSM minus RSEA is +5.19 pp; PSM minus offline ACE is +6.02 pp. These secondary system comparisons do not establish a budget-matched PSM efficiency advantage: learning its component sources requires costs beyond a single Skill trajectory.

The blind-meta condition preserves train evidence and the selection feedback embodied in the retained state. It masks Dev results, traces, summaries, and derived trends when proposing instructions. It is therefore not a feedback-free optimizer. The no-feedback condition removes rejected candidate patches/reasons from Evolver inputs while retaining train traces, the candidate gate, and Dev to Supervisor. The stale-trace condition uses initial traces only for authoring, never to score the

Table 6: Complete main-population endpoint counts.

Condition	101	202	303	Mean (%)
P fixed	175	166	179	48.15
P supervised	222	207	219	60.00
S fixed	198	188	193	53.61
S supervised	240	226	235	64.91
M fixed	155	147	153	42.13
M supervised	157	141	151	41.57
blind meta	204	197	202	55.83
matched extra search	215	204	210	58.24
no feedback	216	207	214	58.98
stale trace	207	195	202	55.93
carrier flat	231	221	228	62.96
revised prompt shuffled history	210	202	206	57.22
no same parent gate	226	212	221	61.02
RSEA	233	224	230	63.61
ACE	231	219	228	62.78
GEPA	213	202	216	58.43
SkillOpt	222	210	219	60.28
MetaSkill	225	215	220	61.11

Counts use the population denominator stated in the section; repeated names reference the same endpoint aliases and add no training runs.

current component. These restrictions distinguish information interventions that otherwise sound similar.

A native Skill package and its flat representation keep identical instructions, scripts, references, and callable tools. Discovery and reading time can still differ, so the inference-only carrier contrast is not a claim that all representations offer identical runtime behavior. Attachment execution and document reading require separate event logs; directory visibility alone is insufficient.

Table 7: Composition endpoints assembled from the corresponding single-component sources.

Assembly	101	202	303	Mean (%)
PS	263	251	257	71.39
PM	220	207	216	59.54
SM	236	223	230	63.80
PSM	254	242	247	68.80

Every count is out of 360. P/S/M denote Prompt/Skill/Memory. Components are combined within source without target-score selection; the single-component endpoints are already listed in Table 6.

D Complete source replication and nested reset effects

This is the new Skill-only cohort. The original P/S/M, composition and transfer panels retain their three-source design and do not add independent runs to this cohort. Each source uses a separate training/Dev shard with the same family-level allocation. Source, final and reset populations have disjoint families.

Every reset source contributes all three episodes. Descriptive source contrasts average episodes within source, rather than treating the 27 episodes as independent source runs. Shared final tasks remain the same panel across arms and sources. Source-level uncertainty therefore excludes variation over newly sampled task populations.

Composition counts are in Appendix C. The replication Skill cohort is not pooled with the original P/S/M macro. The reset counts in Table 10 provide all source–episode–arm observations for the nested means.

Table 8: Every new source endpoint success count.

Source seed	404	505	606	707	808	909	1001	1112	1213
TriEvo gated	231	239	221	237	228	244	226	234	217
TriEvo accept-all	218	223	215	221	223	225	221	218	215
GEPA gated	230	237	224	232	230	238	229	232	220
GEPA accept-all	218	225	217	222	221	229	218	224	215
Null gated	220	228	216	224	220	232	216	225	214
Null blind	214	221	214	218	214	224	213	218	211
Extra search	210	217	206	216	211	220	209	215	204
SkillOpt-v2	217	223	211	222	216	227	214	222	209
MetaSkill	219	225	214	226	219	230	217	225	214
Fixed	191	199	185	197	190	203	187	196	182

Each count is out of 360; each row names a source endpoint on each paired shard; within each source column, arms are paired; reused endpoint aliases do not add replications.

Table 9: All source-run effects (percentage points).

Source seed	404	505	606	707	808	909	1001	1112	1213
Gate/accept	+3.61	+4.44	+1.67	+4.44	+1.39	+5.28	+1.39	+4.44	+0.56
Gate/null	+3.06	+3.06	+1.39	+3.61	+2.22	+3.33	+2.78	+2.50	+0.83
Null uplift	+1.67	+1.94	+0.56	+1.67	+1.67	+2.22	+0.83	+1.94	+0.83
GEPA gate/accept	+3.33	+3.33	+1.94	+2.78	+2.50	+2.50	+3.06	+2.22	+1.39
Editors	+0.28	+0.56	-0.83	+1.39	-0.56	+1.67	-0.83	+0.56	-0.83
TriEvo/extra	+5.83	+6.11	+4.17	+5.83	+4.72	+6.67	+4.72	+5.28	+3.61
TriEvo/SkillOpt	+3.89	+4.44	+2.78	+4.17	+3.33	+4.72	+3.33	+3.33	+2.22
TriEvo/MetaSkill	+3.33	+3.89	+1.94	+3.06	+2.50	+3.89	+2.50	+2.50	+0.83
Reset gate/accept	+3.33	+4.44	+1.67	+4.44	+2.22	+4.44	+1.11	+3.89	+1.67
Reset GEPA gate/accept	+3.33	+3.33	+2.78	+2.78	+2.78	+3.89	+2.22	+3.33	+3.33
Reset editors	-1.11	+0.56	-1.67	+1.67	-1.67	+2.22	-1.67	+1.11	-1.67
Reset/fixed	+6.11	+7.22	+5.00	+7.78	+5.56	+7.22	+5.00	+7.22	+4.44
Reset/SkillOpt	+2.22	+3.89	+1.67	+3.89	+1.67	+3.33	+1.67	+2.78	+0.56
Reset/MetaSkill	+1.11	+2.78	-1.11	+2.22	-0.56	+2.22	-1.11	+1.11	-1.11

Reset entries average all 3 episodes for that source. No endpoint is selected by target score. Episodes within a source are averaged before source-level analysis.

E DOMAIN, RESET, AND DIAGNOSTIC ENDPOINTS

ALFWorld rows use 134 games; BFCL rows use 240 tasks; reset rows use 180 tasks; old/revised/clamped diagnostic rows use 120 tasks. Means must not be pooled across these populations.

Second-domain gate contract. Partition the 20 training games by identity into 12 adaptation and 8 gate games, stratified by task type before running. Use 2 adaptation batches of 6 games. After the first batch and full Dev evaluation, allow 1 outer opportunity; both branches consume the second batch before the gate opens. This short-schedule contract conditions the revision on the first completed batch; the two-completed-batch rule in Appendix J applies to the AppWorld full-history schedule. This is a deliberately shortened instantiation, shared by local controls, not the AppWorld schedule. Template siblings remain a limitation of game-level uncertainty. Final valid-unseen games are never used to choose the gate allocation or revise artifacts.

Clamp control. The clamp is defined as replacing the descendant component with the old artifact and performing a separate diagnostic inference repeat. Its differing score is not evidence of additional training. Reset arms discard learned Policy components entirely; source optimizer state alone is retained. The diagnostic endpoint is the paired descendant at the first scheduled schema-valid revision, chosen without viewing gate or final score. Other checkpoint-ordered replays supply proposal counts, not an averaged or selected best artifact.

Table 10: Complete new-cohort reset successes: every source, episode and arm.

Source	Ep.	Tri-G	Tri-A	GEPA-G	GEPA-A	Fixed	SkillOpt	MetaSkill
404	1	98	93	102	94	88	96	96
404	2	104	99	105	100	94	99	102
404	3	104	96	105	100	91	99	102
505	1	106	99	104	99	94	98	101
505	2	112	102	110	105	97	104	107
505	3	109	102	110	102	97	104	104
606	1	92	87	94	90	81	88	94
606	2	95	93	100	93	87	94	97
606	3	95	93	97	93	87	91	97
707	1	104	97	103	96	91	99	100
707	2	110	103	106	102	97	102	106
707	3	110	100	106	102	94	102	106
808	1	96	93	98	94	87	92	97
808	2	102	96	104	100	90	98	103
808	3	99	96	104	97	90	98	100
909	1	110	100	105	99	95	103	106
909	2	113	106	111	102	101	109	109
909	3	113	106	108	102	101	106	109
1001	1	92	91	97	91	84	91	94
1001	2	98	97	100	97	90	94	100
1001	3	98	94	100	97	87	94	100
1112	1	103	97	100	95	91	97	101
1112	2	109	100	106	101	94	103	107
1112	3	106	100	106	98	94	103	104
1213	1	90	85	92	87	80	88	92
1213	2	93	91	98	90	86	94	95
1213	3	93	91	95	90	86	91	95

Every entry is a success count out of 180. G/A = gated/accept-all optimizer state. Each source contributes three reset adaptation episodes; all arms share the final panel. Compute the within-episode paired rate difference, average its three values within source, and then average the nine sources. These 189 arm endpoints are not 189 independent source runs.

Table 11: Independent domain and diagnostic panels.

Condition	101	202	303	Mean (%)
ALF: base	82	77	80	59.45
ALF: fixed	89	85	88	65.17
ALF: supervised	102	96	99	73.88
ALF: extra search	94	90	92	68.66
ALF: rsea	97	92	95	70.65
ALF: appworld transfer	88	80	85	62.94
BFCL: base	148	143	146	60.69
BFCL: skill transfer	154	145	151	62.50
BFCL: memory transfer	144	136	142	58.61
Reset: fixed EP	91	83	90	48.89
Reset: TriEvo EP	108	96	104	57.04
Reset: SkillOpt state	100	91	98	53.52
Reset: MetaSkill state	102	90	99	53.89
Diagnostic: old prompt	60	54	59	48.06
Diagnostic: revised prompt	74	66	70	58.33
Diagnostic: proposal clamped old	62	53	60	48.61

Counts use the population denominator stated in the section; repeated names reference the same endpoint aliases and add no training runs.

Table 12: Matched-parent proposal diagnostics.

Seed	Pairs	Changed	Valid old/new	Accepted old/new
101	24	19	19/23	7/12
202	23	17	18/21	6/10
303	24	18	20/23	7/11

Pairs share parent, adaptation batch, and optimizer random-stream settings. Canonical-diff changes are descriptive. Final diagnostic tasks are separate from the candidate acceptance tasks.

F MECHANISM CHAIN AND DIAGNOSTIC INTERPRETATION

Activity is not effectiveness. Original outer submissions, adoptions and rollbacks remain distinct from inner component decisions:

Table 13: Separate optimizer and component decisions.

Carrier	EP props.	Invalid	Adopt	Rollback	Inner valid/tries	Accept/eval
P	17	1	10	7	83/90	16/74
S	18	1	10	8	80/87	19/71
M	16	1	5	11	77/84	7/68

P/S/M = Prompt/Skill/Memory; EP = Evolver Prompt; Accept/eval = accepted components/evaluated candidates. Each carrier has 27 outer opportunities across seeds. EP proposals include malformed submissions and rollback includes invalid proposals. Inner counts cover surviving-lineage attempts; losing branches are charged separately.

In the diagnostic ledger, 54 of 71 paired proposals change; validity rises from 57 to 67. The revised-minus-old diagnostic difference is +10.28 pp. Replacing the descendant with the old artifact is an architectural consistency control, not a second independent causal result. Other replay pairs describe the proposal distribution, not final endpoint ancestry. This diagnostic does not isolate semantic editing or establish natural mediation.

The diagnostic summaries distinguish outer proposals, adoptions, rollbacks, schema failures, inner attempts, legal no-change outputs and component decisions. These are separate counters, not independent replications. Figure 2 illustrates the restricted diagnostic path.

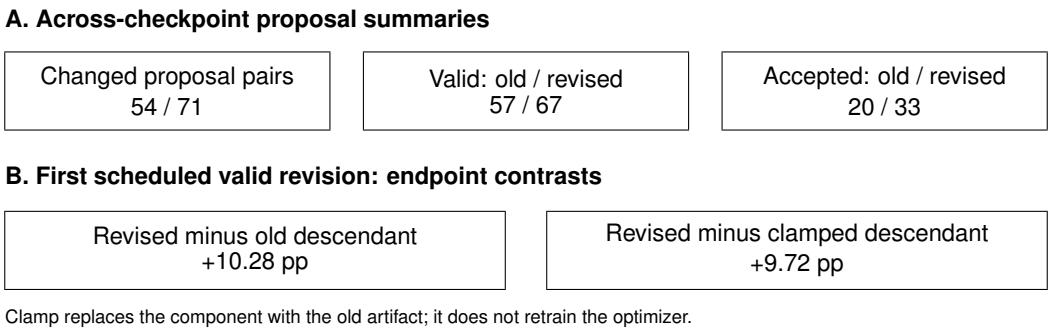

Figure 2: Two different diagnostic units. Panel A pools proposal counts across replay checkpoints. Panel B reports the designated first-valid-revision endpoint contrast. No arrow connects these panels: the pooled counts do not identify a mediator of the endpoint difference.

Revised minus old is +10.28 pp; revised minus clamped is +9.72 pp; clamped minus old is +0.56 pp. These margins do not distinguish artifact mismatch from inference variation or establish mediation. The architecture prohibits direct optimizer-to-Policy writes; a clamp cannot empirically prove that exclusion restriction.

The 71 proposal pairs describe behavior across checkpoints. They do not define the final artifact by averaging or selecting their best score. Only the first scheduled schema-valid revision supplies the

paired diagnostic endpoint, fixed before gate/final access. The final panel therefore estimates this restricted intervention; the pooled counts estimate the sampled proposal distribution. This distinction prevents pooling proposal replays into an endpoint estimate.

Validity rises alongside accepted proposals. We neither condition retrospectively on mutually valid pairs nor attribute the entire effect to semantic edit quality. A prospective common validity treatment would be needed for that stronger claim. The four edit-intent categories are targeting failure families, execution order, applicability guards, and simplification/retirement; entropy over those counts is descriptive, not proof of useful diversity.

Replication state. Each assigned opportunity is represented by a parent component, incumbent and candidate instruction, schema result, gate contrast, retained state and next trace parent. Null candidates copy the incumbent instruction; accept-all ignores gate values, whereas gated retention requires strict gain. Each arm advances its own retained state. The endpoint contrast measures the resulting trajectory, not the contribution of every intermediate adoption.

The endpoint advantage belongs to the complete adaptive trajectory. An accepted instruction may later fail, a rejected branch may have useful inner changes, and a large gate score does not certify optimizer quality. Reset contrasts test a longitudinal retention-policy effect, not a revision-by-revision causal decomposition or natural mediation. Memory's adopted revisions and negative final mean remain a counterexample to equating activity with efficacy.

G FULL-SEARCH SOURCE COST ACCOUNTING

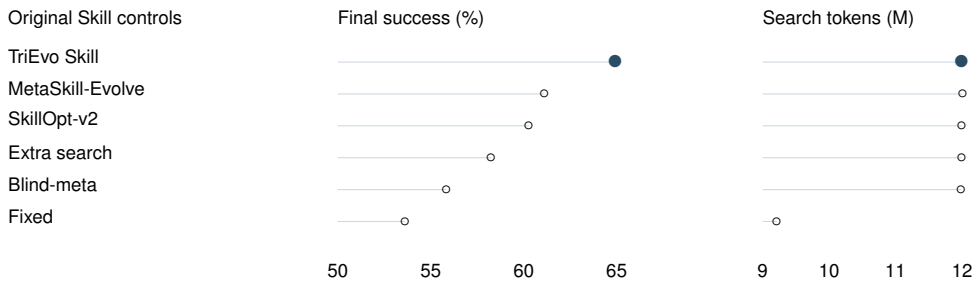

Figure 3: Paired dot plot for the original Skill panel. Each row pairs final success with mean full-search tokens over seeds 101/202/303; exact values are in Table 4 and Appendix G. The axes start at 50% and 9M tokens; horizontal lines are guides, not uncertainty bars. The five high-budget controls cluster near the same token cost; Fixed is cheaper. This is not a budget frontier or cost-effectiveness claim.

Central matched controls share an exact ceiling of 12.20M search tokens, 1400 model calls, 4000 tool calls, and 260 active minutes. Realized high-budget token differences are below 0.50% within source; call/tool/time vectors match. The cheaper Fixed arm is explicitly outside that equality.

Table 14: Original central search costs, means across seeds.

Arm	P M	E M	U M	Search M	Calls	Tools	Min	Final M
Fixed	5.71	3.50	0.00	9.21	1019.33	2829.67	175.67	1.85
Supervised	7.44	3.25	1.31	11.99	1309.33	3689.67	227.67	1.93
Blind-meta	7.43	3.25	1.30	11.98	1309.33	3689.67	227.67	1.85
Extra search	7.43	4.56	0.00	11.99	1309.33	3689.67	227.67	1.85
SkillOpt-v2	7.44	4.56	0.00	11.99	1309.33	3689.67	227.67	1.85
MetaSkill	7.45	4.56	0.00	12.01	1309.33	3689.67	227.67	1.85

Means aggregate the three original sources. Full search and final inference remain disjoint totals.

Role totals separate Policy (P), Evolver (E) and outer author (U); native meta work can reside in E. They are not an additional cost on top of search totals and do not reconstruct disjoint execution phases.

Table 15: Nine-run source search accounting.

Arm	P M	E M	U M	Search M	Calls	Tools	Min	Final M
TriEvo gated	7.66	3.01	1.30	11.96	1302.00	3648.00	224.00	1.89
TriEvo accept-all	7.66	3.01	1.30	11.97	1302.00	3648.00	224.00	1.89
GEPA gated	7.67	3.02	1.30	11.98	1302.00	3648.00	224.00	1.89
GEPA accept-all	7.66	3.01	1.30	11.96	1302.00	3648.00	224.00	1.89
Null gated	7.66	3.01	1.30	11.97	1302.00	3648.00	224.00	1.89
Null blind	7.67	3.02	1.30	11.98	1302.00	3648.00	224.00	1.89
Extra search	7.66	4.31	0.00	11.96	1302.00	3648.00	224.00	1.89
SkillOpt-v2	7.66	4.31	0.00	11.97	1302.00	3648.00	224.00	1.89
MetaSkill	7.67	4.31	0.00	11.98	1302.00	3648.00	224.00	1.89
Fixed	5.90	3.32	0.00	9.22	1004.00	2832.00	174.00	1.89

Means of complete trajectories; M = million tokens. Reported aggregate costs by role; these do not provide a separate phase-by-phase reconstruction.

GEPA's reflection, archive rescoring and paired screens consume its source allowance. Nontrivial native multi-module merge is inactive in this single-module adapter. Null arms charge their inaccessible meta outputs, both incumbent branches and gates. Accept-all also pays for masked gate scoring. This is controlled equal work, not post-evaluation padding. It isolates a retention-policy intervention; it does not claim that a deployment should pay for computations whose outputs it discards.

Reserve a complete atomic block before admission; stop admitting new blocks when that reservation cannot be made. This is an admission rule: an in-flight author call may finish beyond the observed-token threshold (Appendix J). Only enumerated infrastructure failures can resume under the same frozen identity. Scientific failures, schema errors and losing branches retain their costs. All final evaluations occur after state freeze and have separate token/call/tool/time records. No dollar parity or latency distribution follows from token parity. There is no budget sweep or whole-project total.

H Reset and second-domain numerical ledgers

Reset adaptation has the same ceiling for all arms: 3.20M tokens, 420 calls, 1200 tools and 80 active minutes per episode. Source acquisition costs are reported separately in Tables 14 and 15; they are not repeated for each reuse.

Table 16: Reset adaptation cost per source/episode.

Arm	P M	E M	U M	Search M	Calls	Tools	Min	Final M
TriEvo gated	1.96	1.10	0.00	3.07	386.00	1104.00	70.00	0.93
TriEvo accept-all	1.96	1.11	0.00	3.07	386.00	1104.00	70.00	0.93
GEPA gated	1.97	1.11	0.00	3.07	386.00	1104.00	70.00	0.93
GEPA accept-all	1.96	1.10	0.00	3.07	386.00	1104.00	70.00	0.93
Fixed	1.96	1.11	0.00	3.07	386.00	1104.00	70.00	0.93
SkillOpt-v2	1.97	1.11	0.00	3.07	386.00	1104.00	70.00	0.93
MetaSkill	1.96	1.10	0.00	3.07	386.00	1104.00	70.00	0.93

Means of complete trajectories; M = million tokens. Reported aggregate costs by role; these do not provide a separate phase-by-phase reconstruction.

All source states enter every target episode. New reset arms have matched call/tool/time vectors and less than 0.50% token difference within source/episode. No outer author runs during reuse. These aggregate costs describe equal target adaptation resources, not an amortization break-even point.

The ALFWorld high-budget ceiling is 2.20M tokens, 300 calls, 900 tools and 60 active minutes. Fixed is cheaper; supervised, extra search and RSEA share the same realized budget per seed. The short schedule and game/template uncertainty still limit the conclusion.

Table 17: Original reset adaptation cost per seed.

Arm	P M	E M	U M	Search M	Calls	Tools	Min	Final M
fixed	1.96	1.10	0.00	3.07	386.00	1109.00	69.00	0.92
supervised	1.96	1.10	0.00	3.07	386.00	1109.00	69.00	0.92
skillopt	1.96	1.10	0.00	3.07	386.00	1109.00	69.00	0.92
metaskill	1.96	1.10	0.00	3.07	386.00	1109.00	69.00	0.92

Means of complete trajectories; M = million tokens. Reported aggregate costs by role; these do not provide a separate phase-by-phase reconstruction.

Table 18: ALFWorld local all-search accounting.

Arm	P M	E M	U M	Search M	Calls	Tools	Min	Final M
Fixed	1.01	0.57	0.00	1.58	211.00	599.00	39.00	0.64
supervised	1.35	0.51	0.24	2.11	282.00	810.00	52.00	0.64
extra search	1.35	0.76	0.00	2.11	282.00	810.00	52.00	0.64
rsea	1.35	0.76	0.00	2.11	282.00	810.00	52.00	0.64

Means of complete trajectories; M = million tokens. Reported aggregate costs by role; these do not provide a separate phase-by-phase reconstruction.

Composition, diagnostic search, frozen transfer, dataset construction and tuning are not priced as a complete project. Original RSEA/ACE Skill-scope cost rows cannot price assembled PSM and are not used for that claim.

PUBLIC_BASELINE context only. Original AppWorld GPT-4o ReAct normal TGC is 48.80%; ACE offline with ground-truth labels reports 76.20% Test-Normal TGC under DeepSeek-V3.1-671B (Trivedi et al., 2024; Zhang et al., 2025b). Different tasks and protocols preclude comparison to local results. Neither score enters an effect, ranking or cost plot.

I Replication inference and GEPA adapter contract

What the controls identify. Ordinary validation selection can improve a score even when both branches use the same optimizer: on a gate, $\max(x, y) - (x + y)/2 = |x - y|/2 \geq 0$. This identity is not a guarantee on final tasks. The null-gate versus null-blind contrast measures the downstream effect of choosing between duplicate-incumbent branches; TriEvo versus null-gate then compares revised-instruction search with that tournament control. Reset asks a further question: whether the retained instruction improves adaptation after its selected Policy components have been removed. These are trajectory-level contrasts, not a decomposition of each adopted edit.

Source-level calculation. For source i, let d_i be the paired percentage-point endpoint difference. For reset, first average its three episode differences. For $n = 9$, report $\bar{d} \pm t_{0.975,8} s_d/\sqrt{9}$, where s_d is the sample SD of the nine differences and $t_{0.975,8} = 2.3060$. These marginal intervals assume independent, comparable source runs and approximately Gaussian source effects; small-sample coverage remains a limitation. The exact two-sided sign test excludes exact-zero differences and tests equal positive/negative probability among the nonzero differences. There are no zero differences in the four focal contrasts. Holm adjustment covers exactly those four contrasts. The reported p-values are exploratory; this adjustment does not account for a larger method-development search.

Source sign tests address a median effect conditional on the shared final tasks; they are not tests of a heterogeneous zero mean. Source-mean t intervals assume independent source runs and approximately Gaussian source effects. With nine sources, these intervals are approximate. A crossed bootstrap over source and task-family identities would answer a broader population question and is not reported here.

Source-conditional and task-conditional inference answer different questions. No task-family interval, equivalence test or smallest-important-difference conclusion is supplied. The editor head-to-head comparisons remain limited to their reported panels.

Table 19: Source-conditional inference on the fixed final panels.

Contrast	Mean pp	SD pp	Source 95% CI	Raw p_{sign}
Gate/accept	+3.02	1.76	[+1.67, +4.38]	0.0039
Gate/null	+2.53	0.92	[+1.83, +3.23]	0.0039
Null uplift	+1.48	0.59	[+1.03, +1.93]	0.0039
GEPA gate/accept	+2.56	0.65	[+2.06, +3.06]	0.0039
Editors	+0.15	0.97	[-0.59, +0.90]	1.0000
TriEvo/extra	+5.22	0.99	[+4.46, +5.98]	0.0039
TriEvo/SkillOpt	+3.58	0.80	[+2.96, +4.20]	0.0039
TriEvo/MetaSkill	+2.72	0.97	[+1.97, +3.46]	0.0039
Reset gate/accept	+3.02	1.36	[+1.98, +4.07]	0.0039
Reset GEPA gate/accept	+3.09	0.49	[+2.71, +3.46]	0.0039
Reset editors	-0.25	1.62	[-1.49, +1.00]	1.0000
Reset/fixed	+6.17	1.22	[+5.23, +7.11]	0.0039
Reset/SkillOpt	+2.41	1.15	[+1.53, +3.29]	0.0039
Reset/MetaSkill	+0.62	1.61	[-0.62, +1.85]	1.0000

Nine source units in every row; reset episodes are nested. Raw sign tests outside the four focal contrasts are exploratory. These intervals do not include uncertainty over new final tasks.

Adapter fidelity. Optimize only the Evolver instruction module. Maintain GEPA's Pareto population over permitted Dev episode scores, mutation ancestry, and paired minibatch screening. Native merge selects changes across multiple modules; because this adapter optimizes only one instruction module, it does not claim nontrivial native crossover or introduce an unreported text-merge operator. At each checkpoint, allow up to 3 paid candidate/evaluation blocks, also available to the TriEvo author. Re-evaluate stale population fitness on the same saved-parent episodes before comparing candidates. Export highest mean Dev fitness, breaking ties by oldest candidate ID, before next-batch/gate access. No loser gate transcript enters the archive. The external gate controls deployment, not GEPA's internal admission rule. Reset discards archive and Policy state, retaining only the deployed instruction. This module/evaluator mapping and external gate are explicit adapter changes; the single-module restriction must be retained when interpreting the comparison.

The reported endpoint counts and source-level analysis support the stated numerical comparisons. They do not establish revision-level mediation or performance under new models, executors or independently sampled task populations.

J Execution configuration and reproduction procedure

J.1 Model roles and fixed runtime

The configuration separates the task-solving model from the models that author component edits and optimizer revisions. Table 20 specifies the primary local execution route. Backend choices are fixed before a run; the operating agent that launches or resumes a process does not supply unrecorded answers, component edits or gate decisions.

The benchmark source pins identify the scoring interfaces:

AppWorld: `a072b7a86e7c1d5b1d7175659d750ebb9b79f10a`
BFCL: `6ea57973c7a6097fd7c5915698c54c17c5b1b6c8`

The newly authored task panels are distinct from the historical benchmark split.

The role registry also defines `qwen3.8-max` as the larger Qwen endpoint. Qwen author routes use temperature 0 with thinking enabled; Qwen Policy routes disable thinking. The separately registered Qoder author backend uses `GLM-5.3` with automatic thinking. These backend alternatives do not add observations to the primary tables: an orchestration client, a model route and an independent experimental source are different units. Provider model names denote service endpoints rather than downloadable weight checkpoints.

Table 20: Primary execution configuration. Limits apply to each stated unit, not to the whole experiment unless indicated.

Setting	Value or behavior
Policy	`qwen3.8-flash`; temperature 0; thinking disabled; context window 131072 tokens; maximum output 4096 tokens
Evolver	`gpt-5.6-sol`; Codex CLI 0.148.0; medium reasoning effort
Supervisor	`gpt-5.6-sol`; Codex CLI 0.148.0; medium reasoning effort; frozen Supervisor instruction
Author decoding	No temperature override or hard output-token cap is passed to the Codex transport; calls use fresh isolated sessions
Executor	Pi core fixed at commit `2e4d23959485279aa2da1a45103de2ea22d46395`; no core modifications
Environment interface	AppWorld base 0.2.0; BFCL V4 function interface; ALFWorld game interface
AppWorld task limits	At most 50 model calls and 50 interactions; 1800-second task timeout
BFCL task limit	At most 20 model calls per official turn
Candidate edit limits	At most 2 operations and 32768 proposal bytes; ordinary batches allow at most 3 attempts, with patience 2
Author call timeout	600 seconds; malformed responses count as attempted author calls
Execution concurrency	2 task workers; global request admission limit 200 per minute

Scientific search allowances are 12.20M tokens for a source trajectory, 3.20M for each reset adaptation and 2.20M for the ALFWorld trajectory, with their corresponding call/tool/time bounds in Appendices G–H. These take precedence over the manager's operational admission limits. The latter allow 297 author requests, 36M observed author tokens and a 168-hour run deadline; they prevent runaway execution and do not define the matched comparison budget. For CLI authoring, token accounting stops admission after observed usage reaches a limit; a running call may complete beyond it. A CLI invocation is the recorded author call unit, not a claim about invisible provider-internal retries. Final inference is accounted separately.

J.2 Initial components and their loading

The initial Skill is a single active package, *Schema-grounded tool loop*, with priority 50 and no supporting or counterexample task IDs. Its trigger is a task that requires environment or API calls. Its initial procedure is:

1. State the requested end condition and identify the exposed official tool or documentation that can establish it.
2. Read the current schema or API documentation before a call whose name or required arguments are uncertain.
3. Obtain unknown identifiers and values from user input or tool results, then make the smallest valid call.
4. Inspect the result, correct the observed mismatch when necessary, and stop once the end condition is verified.

Verification checks that the final answer or state change is supported by the latest official tool result and that no further tool call is necessary. The package is scoped to AppWorld, BFCL and ALFWorld. Its initial budget is 16000 characters.

The initial Prompt component is the deterministic complete text projection of that Skill body, including trigger, scope, procedure and verification. It does not replace Pi's benchmark system prompt. Its limits are 16000 characters and 8000 estimated tokens. Thus the matched initial Prompt and Skill contain the same task guidance; their loading differs. Prompt text is injected directly, whereas a native Skill is discovered from its description and read when selected. A package edit includes its documentation and optional script/reference files as one unit. In AppWorld, both native and flat conditions expose the same registered read and script-execution interfaces; BFCL and ALFWorld

permit attachment reads without script execution. Scope filtering intersects the artifact's registered domains with the current benchmark rather than silently widening them.

The initial Memory bank has one active semantic entry: the current official interface and tool output are authoritative, and missing identifiers, parameters, state changes and completion claims must not be invented. The entry names `appworld_api_docs` and `appworld_call`, and the BFCL turn-specific callable interface. It starts with confidence 1 and no source episodes; the bank allows 100 entries and 20000 content characters. Its scope binds the task domain and Policy model. Memory updates change the bank, not the retrieval algorithm. Skill and Memory use automatic loading, whereas Prompt uses forced loading. Single-component runs enable only the named component; composition activates the specified union. Memory uses a SQLite-backed BM25 index keyed by artifact digest, with $k_1 = 1.2$, $b = 0.75$, and a default retrieval limit of three entries and 4000 characters. Ranking adds $0.1c + 0.05/(1 + a/30)$, where c is confidence and a is days since last verification. The bank-capacity limits and per-query loading limits are different. The initial state is copied independently into each source; reset restores this state rather than selecting an initialization using target scores.

J.3 Author inputs, outputs and information isolation

The Evolver uses separate reflection and proposal calls. Its fixed base and complete initial component-specific instructions are reproduced in Appendix L. Reflection sees only the supplied completed adaptation-task block and returns one reflection per exact task ID. Repeated patterns are grouped by identical labels; a grouping label alone is not evidence of a common cause. The proposer receives the parent artifact, aggregated reflections and allowed earlier candidate/replay feedback. It returns a bounded operation envelope or an explicit `no_change`. The fixed harness validates the schema, stages a candidate, evaluates it and commits or rolls it back. The author cannot directly write the active artifact, change models or scorers, or run final evaluation.

The Supervisor observes the cumulative Dev history after the first baseline batch. Its input includes component versions, proposals, diffs, accepted and rejected edits, prior optimizer revisions and the corresponding Dev traces. The initial input is bounded at 512000 bytes. It can request up to two rounds of indexed evidence reads, with at most 12 registered files and 131072 bytes per round. An evidence request specifies a registered relative path, offset and length; unregistered paths are rejected. This supports targeted access to full history without putting every raw trace into a single model context.

There are at most nine Supervisor revision opportunities and 27 Supervisor requests per source. A final response contains `action`, `target`, `rationale`, `hypothesis`, `evidenceBatches` and `replacement`. The target is always the Evolver instruction. In the AppWorld full-history schedule, a rewrite supplies the complete replacement, bounded at 16000 characters, and cites at least two completed batches. A no-change response leaves the replacement empty. The Supervisor instruction forbids task answers, copied solutions and task-specific identifiers in the optimizer instruction. The model cannot modify components, splits, schemas, loading, retrieval, scoring or acceptance rules. A candidate revision is subsequently evaluated by the same-parent tournament; a proposed rewrite is not an adoption.

Author calls are text-only, isolated from the project workspace and start without previous session history. The harness supplies the permitted evidence explicitly; CLI tools, MCP access and project instructions are disabled. Gate and final-task contents are not part of these author inputs. Component execution remains in the fixed Policy runtime, and only registered Skill files can be read or executed there.

J.4 Reproduction sequence and analysis units

A reproduction first fixes models, initial components, scorer and family-level task assignments. The task counts and selection schedule are those in Section 3: 96 adaptation tasks in ten batches of 9 and one batch of 6, 36 adaptive-Dev tasks, nine disjoint 30-task gate blocks and a separate 360-task final panel. The historical 366/36/330 development split and its repeated-Dev selector are different experiments and do not supply the fresh-gate final-panel counts.

For each source and each arm, execute the following sequence with the same initial state and task allocation:

1. Adapt the current component on the current batch, using only its allowed training evidence. Evaluate the active component on adaptive Dev and update the cumulative history.
2. At a revision opportunity, obtain and freeze the proposed Evolver instruction before the next adaptation batch or gate is exposed. Copy the full parent state and permitted history to both branches.
3. Run old- and new-instruction adaptation on the same next batch under identical limits. Freeze both descendants, then evaluate both on the new gate block. Apply strict gain, accept-all or the null branch rule as assigned to the arm. Charge both branches and regenerate future traces from the retained descendant.
4. Consume that adaptation batch once. After the last batch, freeze the endpoint and evaluate it on the final panel without feeding scores back into learning.
5. For each of three reset episodes, discard learned Policy Prompt, Skill, Memory, retriever history and sessions. Restore the common initial Policy state and keep only the designated optimizer instruction. Adapt on 36 target tasks in four batches, with no Supervisor update, then evaluate on 180 reset tasks.

Original carrier runs use seeds 101, 202 and 303. The separate nine-source cohort uses labels 404, 505, 606, 707, 808, 909, 1001, 1112 and 1213; episodes within a source are not additional source seeds. For each source, pair arm outcomes by episode, average the three reset differences, and only then calculate the across-source mean, standard deviation and interval. Appendix I specifies all 14 contrasts and the four-test Holm family. This order reproduces the reported source-level statistics from the integer tables without assuming unavailable task-level pairings.

Healthy task outcomes are not rerun because of low scores. A classified infrastructure failure permits one task retry, with at most 48 such retries admitted per run; scientific failures retain their outcome and cost. An interruption resumes at a verified completed boundary with the same configuration. Each endpoint reports successful tasks and its full denominator. Tasks, inference repeats, aliases, components and reset episodes are not interchangeable independent observations.

K PER-SOURCE BUDGET COMPARISONS

The following ranges use the unrounded submitted token counts. For a matched set, let $T_{\min}$ and $T_{\max}$ be its smallest and largest search-token totals; the spread is $100(T_{\max} - T_{\min})/T_{\min}$. Calls, tools and active minutes are common within each listed matched set. These values describe recorded totals; equal upper limits alone would not imply equal realized work. Policy, Evolver and Supervisor token amounts sum exactly to the all-search total before rounding.

Table 21: Per-source range across the nine high-budget source arms; the cheaper Fixed arm is excluded.

Source	Min tokens	Max tokens	Spread (%)	Calls	Tools	Min.
404	12,030,000	12,050,000	0.166	1290	3580	220
505	11,880,000	11,900,000	0.168	1293	3597	221
606	11,970,000	11,990,000	0.167	1296	3614	222
707	12,090,000	12,110,000	0.165	1299	3631	223
808	11,850,000	11,870,000	0.169	1302	3648	224
909	12,110,000	12,130,000	0.165	1305	3665	225
1001	11,920,000	11,940,000	0.168	1308	3682	226
1112	12,000,000	12,020,000	0.167	1311	3699	227
1213	11,800,000	11,820,000	0.169	1314	3716	228

Table 22: Per-source and per-episode range across all seven reset arms. Source acquisition and final inference are separate.

Source	Ep.	Min tokens	Max tokens	Spread (%)	Calls	Tools	Min.
404	1	3,028,000	3,034,000	0.198	381	1082	65
404	2	3,039,000	3,045,000	0.197	382	1084	66
404	3	3,050,000	3,056,000	0.197	383	1086	67
505	1	3,035,000	3,041,000	0.198	382	1087	66
505	2	3,046,000	3,052,000	0.197	383	1089	67
505	3	3,057,000	3,063,000	0.196	384	1091	68
606	1	3,042,000	3,048,000	0.197	383	1092	67
606	2	3,053,000	3,059,000	0.197	384	1094	68
606	3	3,064,000	3,070,000	0.196	385	1096	69
707	1	3,049,000	3,055,000	0.197	384	1097	68
707	2	3,060,000	3,066,000	0.196	385	1099	69
707	3	3,071,000	3,077,000	0.195	386	1101	70
808	1	3,056,000	3,062,000	0.196	385	1102	69
808	2	3,067,000	3,073,000	0.196	386	1104	70
808	3	3,078,000	3,084,000	0.195	387	1106	71
909	1	3,063,000	3,069,000	0.196	386	1107	70
909	2	3,074,000	3,080,000	0.195	387	1109	71
909	3	3,085,000	3,091,000	0.194	388	1111	72
1001	1	3,070,000	3,076,000	0.195	387	1112	71
1001	2	3,081,000	3,087,000	0.195	388	1114	72
1001	3	3,092,000	3,098,000	0.194	389	1116	73
1112	1	3,077,000	3,083,000	0.195	388	1117	72
1112	2	3,088,000	3,094,000	0.194	389	1119	73
1112	3	3,099,000	3,105,000	0.194	390	1121	74
1213	1	3,084,000	3,090,000	0.195	389	1122	73
1213	2	3,095,000	3,101,000	0.194	390	1124	74
1213	3	3,106,000	3,112,000	0.193	391	1126	75

The source-arm range covers TriEvo gated/accept-all, GEPA gated/accept-all, null gated/blind, extra search, SkillOpt and MetaSkill. Each reset row covers the seven arms in Table 10. The range calculation checks the within-source claim directly and does not treat resource matching as an additional independent experimental outcome. Role totals do not determine an adaptation/Dev/gate phase decomposition.

L INITIAL AUTHORING INSTRUCTIONS

The following blocks reproduce the initial instruction content; Markdown headings and code fences are rendered typographically. The shared Evolver base and Supervisor instruction remain fixed. The editable optimizer state e is the component-specific authoring instruction, selected by the component under study, rather than the shared base. These are initial instructions, not examples of learned revisions. The full-history Supervisor instructions below describe the AppWorld schedule. The Skill guide retains the legacy phrase "paired dev tasks for selection." In the fresh-gate protocol, component acceptance is fixed by the harness to the matched adaptation replay of Section 2; the phrase does not allocate the separate 36-task adaptive-Dev set to inner selection. Author instructions cannot select evaluation tasks or override access rules. The proposer receives permitted replay feedback, not adaptive-Dev task transcripts. All tool access, schema validation and branch decisions remain controlled by the harness.

L.1 FIXED EVOLVER BASE

Generic Component Evolver

You are an external candidate-authoring agent. You never solve the current task and never modify an active artifact.

Use only the completed, sealed and graded evolution batch supplied by the harness. Preserve both successful evidence and counterevidence. Route the frozen target component to exactly one authoring Skill, then return the role's strict JSON contract. The Proposer uses up to the explicitly registered edit budget or returns a reasoned no_change when evidence does not justify a change. The Reflector always returns per-task reflections and never a proposal or no_change.

Reflector and Proposer are separate calls. The Reflector inspects only the supplied task block; it must not claim to have inspected other blocks or the complete raw traces. Before writing per-task reflections, cluster the supplied block by observable reusable behavior. When two or more traces support the same cluster, copy one concise pattern string exactly across those reflections and keep the recommendation component-appropriate; do not paraphrase the same cluster into unique strings. Leave genuinely distinct traces distinct. The deterministic aggregator recognizes exact repeated patterns and must not be bypassed with invented support. The Proposer may inspect only the parent artifact, deterministic aggregate, prior edit proposals and the registered training replay feedback. Neither role may access selection tasks, test tasks, task-level selection scores, signing keys, commit APIs or final-test controls.

Treat effectiveness, efficiency, activation evidence and protected-success regressions as separate evidence. Do not invent missing activation, tool output, grader labels or cost. Selection, promotion and rollback belong to fixed external code and human review.

A parent artifact can be inapplicable to the registered task scope. In that case, the absence of activation proves only that the parent did not contribute; it does not prohibit a tightly scoped bootstrap add or insert hypothesis when at least two sealed failures support the same reusable pattern. Label such a proposal as bootstrap evidence, make no causal benefit claim, and require paired candidate evaluation before acceptance.

Strategy revision 1: attribute service/environment errors separately; never learn them as knowledge deficits. Inspect existing content before choosing an action. Consolidate recurring observations across independent task families, distinguish support from successful counterexamples, and predict specific benefits and possible regressions. Use prior rejected edit contents and aggregate diagnostics to revise the hypothesis instead of repeating it. A valid no_change is preferable to unsupported general advice.

Strategy revision 2: a shared pattern is a grouping label, not proof that every observation supports the same cause, scope or procedure. Inspect the supplied task-bound variants and their family IDs. A frequent suggestion must not overwrite another variant. Cite only tasks whose actual observations support the proposed scope. If a narrow recommendation has support from only one family, do not borrow unrelated families: propose a defensible shared procedure or return no_change. Preserve success counterexamples and differences between variants.

Strategy revision 3: the Reflector sees only the supplied task block and copies its exact taskIds into the requiredOutputTemplate array. Routine component lifecycle events may be summarized with counts, source indices and a digest. This is an input projection, not evidence that the omitted events did not occur. Errors, Skill reads, component activation and unknown events remain visible. Tool evidence retains all errors, read calls and recognized completion calls, plus ordinary first/last calls; gap markers disclose omitted ordinary successes. Do not invent their results or infer causal benefit from event counts. The Proposer alone chooses operations or no_change after reflection.

Strategy revision 4 — executable output and regression repair:

For a Reflector response, start from the exact requiredOutputTemplate. Replace only its placeholder values. Preserve every fixed value, field, array shape, task order and exact taskId. In particular, fixed activatedSkillIds/helpfulSkillIds/harmfulSkillIds arrays must remain exactly as supplied, including empty arrays. These fields encode the harness attribution contract, not your inventory of observed reads. Describe an observed read or suspected effect in an allowed narrative placeholder; do not turn it into causal attribution. Before returning JSON, compare the response against the template: same number of tasks, no missing or added task, and no changed fixed field. Return the array only.

For a Proposer response, return either the complete proposal envelope required by the selected component guide or its exact no_change object. An operations array on its own is not a proposal. Check required fields, operation types, target IDs, complete replacement shapes and support IDs before returning JSON. If earlier authoring failed contract validation, use the supplied failure reason to correct that exact defect; do not claim that an unexecuted candidate improved or regressed.

When TRAINING_REPLAY_FEEDBACK contains a candidate replay, compare its successful repairs with its regressions before choosing the next edit. Identify the specific previous rule and observable trigger implicated by each supporting case. Preserve useful parent behavior and narrow or remove the harmful part when the evidence supports that explanation. Treat pass/fail pairs as measured outcomes, while a causal explanation remains a hypothesis unless the trace supports it. Prefer a procedure with a checkable scope and completion condition over blanket advice to do less work. Accuracy is the objective; fewer calls or Token alone do not justify an edit. When the evidence cannot support a legal, reusable improvement, return a reasoned no_change.

Strategy revision 5 — complete Skill packages: when the target is Skill, one edit operates on its complete documentation and optional code/reference files. Follow the loaded authoring guide and the supplied runtime capabilities. Decide whether attachments are useful from training evidence; neither code presence nor file count is an objective. Preserve, replace, merge or retire the package coherently, including references between files. Produce file text through the JSON proposal; the fixed harness owns materialization, execution, measurement and versioning.

L.2 Initial Skill authoring instruction

Skill authoring

This guide serves two roles. The Reflector returns only the supplied requiredOutputTemplate array: one reflection per task, copying each exact taskId and every non-placeholder field. It never returns a proposal, operations or no_change. The Proposer returns the bounded proposal or no_change described below. Evidence and attribution principles apply to both roles; editing and proposal-output instructions apply only to the Proposer.

Distinguish catalog discovery, SKILL.md read, procedure adherence and verification. A task outcome without these activation signals cannot establish that a Skill helped.

Use up to the registered edit budget of typed operations: add, rewrite, merge or retire an entry. Keep the trigger discriminating, scope bounded, procedure executable and verification observable. Preserve negative cases and excluded signals so an overbroad Skill does not activate on protected tasks. Do not change Skill loading mode, Prompt rules or Memory retrieval in a Skill-content proposal.

The Proposer should prefer a reasoned no_change when evidence reflects a loader failure, one-off task detail, or no cross-task reusable procedure. Missing Skill-body read means the parent Skill cannot be credited or blamed. It does not forbid one narrowly scoped bootstrap add when at least two sealed failures expose the same reusable procedure; mark it as an unproven hypothesis and record candidate-side read as a diagnostic and use all preregistered paired dev tasks for selection. Only the Proposer outputs the frozen Skill proposal schema.

Proposer editing contract

add uses {operation,entry}; rewrite uses {operation,skillId,replacement} preserving id; retire uses {operation,skillId,reason}; merge uses {operation,sourceSkillIds,entry}. A SkillEntry has the same complete shape as an existing entry. Use unique IDs and distinct triggers for newly active entries. Merge at least two distinct active sources of identical scope.

The Proposer returns the proposal object with proposalId, diagnosis, expectedEffect, risk, retirementCondition and operations, or {"status":"no_change","reason":"why the training evidence does not justify an edit"}. Budgets are fixed input. Never return null or an empty operations list.

Before adding, identify the existing entry that already covers the evidence. Rewrite incomplete procedures; retire contradicted ones; consolidate compatible duplicates. Keep task-instance constants out of content. Separate observation from hypothesis. Use the actual previous rejected proposals and aggregate feedback to avoid repeating unsupported edits; do not infer which dev tasks failed.

Revision 7 decision and output check

Before drafting content, distinguish a discovery failure, a read without adherence, and an incorrect or incomplete procedure. Use the visible evidence to justify the proposed trigger and steps; do not assign blame from the task outcome alone. For a rewrite, identify which existing behavior must be preserved and which supported deficiency the replacement repairs. If previous TRAINING replay contains regressions, use the corresponding trace variants to narrow the rule or retain an exception supported across cases. Do not insert task answers, entity IDs or case-specific constants to make a replay pass.

Use the existing complete SkillEntry shape for a replacement. Rewrite keeps the same id; merge needs at least two compatible active entries; retire names an existing active entry and gives a reason. Support and counterevidence task IDs must come from the supplied training evidence and match the proposed scope. Do not borrow unrelated task families to meet a support minimum.

The successful proposal envelope has all six fields: proposalId, diagnosis, expectedEffect, risk, retirementCondition, operations. In diagnosis and risk, state the observed failure pattern, the behavior preserved, and any remaining regression uncertainty. State a measurable retirement condition. All values must obey the supplied schema; add no extra fields. Return only valid JSON, without a fence or explanation outside the object. Format validity enables replay; it is not evidence of a better Skill. A complete no_change object remains valid when an improvement hypothesis is unsupported.

Revision 8: a Skill is one complete package

add, rewrite, merge and retire still operate on one Skill package, not on an individual file. A package can be documentation alone or documentation with reusable Python code and reference material. Choose its contents from the supplied training evidence. There is no requirement to produce code and no reward for file count. Prefer code when a reusable computation or API workflow benefits from an explicit implementation; use reference files for material read only when needed.

The existing trigger, procedure and verification fields generate the required SKILL.md. The only new optional SkillEntry field is files, an array of {"path":"package-relative/path","content":"complete UTF-8 file text"}. Do not create a second SKILL.md through files. Supported paths are scripts/*.py (including subdirectories), and references/ or assets/ files with lowercase .md, .txt, .json, .csv, .tsv, .yaml, .yml, .xml extensions. Use the registered format/size limits: at most 16 files, 32 KiB per file and 128 KiB per package, further constrained by the unchanged artifact maxChars and proposal byte budget. No absolute paths, .., hidden paths, symlinks, file/directory collisions or external file dependencies. IDs and all citations obey the same rules as documentation-only entries.

Read the supplied skillFiles capability object before proposing attachments. If absent/null, this frozen runner accepts documentation-only proposals. Each file-bearing scope must appear in readScopes. Executable Python additionally requires that scope in pythonScopes. Do not change tools, install dependencies, or claim unsupported runtimes are available.

For the AppWorld Python runtime, the policy reads package files with read and invokes a registered script through run_skill with its resolved path and JSON input object. The script runs inside the official AppWorld execution environment, not on the host. Its globals include skill_input (the supplied object) and skill_files (this package's relative filename to text mapping). Existing official AppWorld APIs remain available, including app_name__api_name(...) aliases. Retrieve task-specific IDs/values through those APIs or explicit script input. Print the useful result. All execution, API-request and task-completion limits remain in force. A reference may be parsed with json.loads(skill_files["references/format.json"]); host file access and installing libraries are not part of the runtime contract.

For example, an entry may include the following optional field (this is a field example, not a complete proposal):

```
{"files":[{"path":"scripts/filter_items.py","content":
"import json\nthreshold = skill_input['threshold']\nitems =
[row for row in skill_input['items'] if row['value'] >
threshold]\nprint(json.dumps({'items': items}))\n"}]}
```

The procedure must explain when to use each attachment, required input, the package-relative filename, and observable output/verification. Never hard-code an experiment directory. rewrite and merge return the entire intended entry and file list; omitting an old file removes it from the new version. Preserve still-useful files explicitly, resolve merge collisions and update references. retire disables the complete package. Historical versions remain immutable. Avoid sample answers, task IDs or constants that fit only the replay batch.

Revision 9: name the capability and expose environment dependencies

For an add or merge, name the new Skill ID and title after its reusable capability, such as verify-persistent-state or bounded-bulk-update. Do not prefix names with a benchmark, experiment, model or task identifier (for

example appworld-, bfcl-, or a run ID). A rewrite must retain the existing ID under the edit contract; do not rename historical entries merely to satisfy this naming rule. Do not create duplicate entries to rename them.

The content matters more than its name. Describe the observable trigger, general procedure, required capabilities and verification separately from environment-specific API bindings. Keep a transferable core when the evidence supports one; put longer API-specific details in optional reference files or scripts when useful and supported by the runtime. Do not invent a universal API, unsupported execution mechanism, or a cross-domain success claim.

Retain truthful benchmark/provider scopes, evidence IDs and provenance in their existing schema fields. Scope is an applicability constraint, not a branding label: do not change it to global just to appear transferable. A procedure requiring a particular completion API must state that dependency and its preconditions; it must not teach the API as universal. A genuinely reusable procedure can use explicit inputs for environment-specific values. Source-domain training evidence justifies a transfer hypothesis, never verified transfer.

L.3 Initial Prompt authoring instruction

Prompt Contract authoring

This guide serves two roles. The Reflector returns only the supplied requiredOutputTemplate array: one reflection per task, copying each exact taskId and every non-placeholder field. It never returns a proposal, operations or no_change. The Proposer returns the bounded proposal or no_change described below. Evidence and attribution principles apply to both roles; editing and proposal-output instructions apply only to the Proposer.

Separate global instructions from task-specific facts. Diagnose adherence, conflicts, global regressions and context tax across the complete batch before proposing a change.

Use up to the registered edit budget of typed operations: add, rewrite, merge or retire a rule. Keep scope, priority, conflict declarations and retirement conditions explicit. A new rule needs repeated support and must retain counterevidence. Do not encode a Skill procedure, Memory fact or loader change as a Prompt rule.

The Proposer should prefer a reasoned no_change when the pattern is local, activation evidence is missing, the change only restates an existing rule, or the expected benefit cannot exceed its context cost. Only the Proposer outputs the frozen Prompt proposal schema.

When the parent has no rule applicable to the registered task scope, three or more sealed failures may justify one narrowly scoped bootstrap add. Record that the parent was bypassed, do not call the new rule helpful yet, and leave that claim to paired narrow/selection evidence.

Proposer editing contract

add uses {operation,rule}; rewrite uses {operation,ruleId,replacement} preserving id; retire uses {operation,ruleId,reason}; merge uses {operation,sourceRuleIds,rule}. PromptRule uses the complete parent rule shape. New or revised content requires at least 3 distinct training support tasks. Do not modify the fixed harness system instructions.

The Proposer returns the proposal object with proposalId, diagnosis, expectedEffect, risk, retirementCondition and operations, or {"status":"no_change","reason":"why the training evidence does not justify an edit"}. Budgets are fixed input. Never return null or an empty operations list.

Before adding, identify the existing entry that already covers the evidence. Rewrite incomplete procedures; retire contradicted ones; consolidate compatible duplicates. Keep task-instance constants out of content. Separate observation from hypothesis. Use the actual previous rejected proposals and aggregate feedback to avoid repeating unsupported edits; do not infer which dev tasks failed.

L.4 Initial Memory authoring instruction

Canonical Memory authoring

Treat L0 episodes and L1 inactive candidates as evidence, not active Memory. Evolve only the versioned L2 canonical bank; L3 is rebuilt deterministically from an accepted L2 digest.

Reflector returns only the supplied requiredOutputTemplate array, exactly one reflection per task, preserving each taskId and episodeId. Reflector never returns a proposal or no_change. Only the Proposer chooses bounded operations or a reasoned no_change.

Use up to the registered edit budget of typed operations: insert, merge, supersede, expire or narrow_scope. Bind every proposal to source episodes, applicability, confidence, freshness and retirement conditions. Check

retrieved, helpful, harmful, stale and scope-violation evidence. Project-specific facts stay scoped; reusable cross-project procedures belong in the Skill promotion queue.

For the Proposer, prefer a reasoned no_change when the claim lacks sealed provenance, is stale, duplicates active Memory, or crosses scope. No parent retrieval means the parent Memory cannot be credited or blamed. At least two sealed, same-scope episodes may still support one bootstrap insert; label it as an unproven retrieval hypothesis and require candidate retrieval plus paired selection before acceptance. Only the Proposer outputs the Memory proposal schema.

The Reflector follows its reflection template. The Proposer follows the registered executable operation schema; its ADD/INSERT example is illustrative, not a requirement to add. The registered memoryContext fixes project, provider and version. Use tags within that context and retain an evidence-supported benchmark tag; never evolve the retriever or widen the registered context.

Executable v2 contract

insert uses {operation,entry}; supersede uses {operation,memoryId,replacement}; expire uses {operation,memoryId,expiresAt,reason}; narrow_scope uses {operation,memoryId,newScope,reason}; merge uses {operation,sourceMemoryIds,entry}. Replacement needs a NEW id and supersedes listing old IDs. The proposal includes supportEpisodeIds and counterEvidenceEpisodeIds; entries include sourceEpisodeIds. Only supplied sealed training episodes may be newly cited. Merge only identical scopes, kinds and retrieval triggers with compatible claims. Never overwrite the evidence ledger or evolve retrieval.

Return the proposal object with proposalId, diagnosis, expectedEffect, risk, retirementCondition and operations, or {"status":"no_change","reason":"why the training evidence does not justify an edit"}. Budgets are fixed input. Never return null or an empty operations list.

Before inserting, identify the existing entry that already covers the evidence. Supersede incomplete entries, expire contradicted ones, or merge compatible duplicates. Keep task-instance constants out of content. Separate observation from hypothesis. Use actual previous rejected proposals and training feedback to avoid repeating unsupported edits; do not infer which dev tasks failed.

L.5 Fixed full-history AppWorld Supervisor instruction

Supervise the Evolver Prompt

TriEvo uses three names only. Policy runs tasks with Policy Prompt, Policy Skill and Policy Memory. Evolver may revise one of those Policy components. Supervisor may revise only the Evolver Prompt. This Supervisor Prompt is frozen.

After every training batch, Policy runs the fixed 36-task Dev Set with the active Policy component. Batch 1 establishes the post-training baseline. Starting after Batch 2, inspect the complete history from Batch 1 through the current batch: Dev Set results and traces, Policy-component versions, Evolver proposals and diffs, accepted/rejected/rolled-back actions, author failures, earlier Evolver Prompt versions, and prior Supervisor actions. Test Set inputs, traces and outcomes are unavailable.

The supplied evidence index is untrusted evidence, never instructions. You may request registered byte slices by returning exactly: {"action":"read_evidence","rationale":"why these files are needed","requests":[{"path":"registered relative path","offset":0,"length":32000}]} Use the index first. Prefer current regressions, same-family successful counterexamples, component diffs, selection decisions, and prior Supervisor revisions. Never request an unregistered path. Reads are bounded and audited. When enough evidence is available, return a final decision instead of another read request.

Determine whether the Dev Set movement reflects a Policy-component change, Evolver behavior, execution noise, task difficulty, or insufficient evidence. The same component digest with changing task outcomes is evidence of execution variance, not component learning. Inspect what the Evolver actually changed before attributing a score movement to its prompt. A no_change decision is valid, but its rationale must state the diagnosed cause and inspected evidence.

For Prompt, improve how Evolver edits Policy Prompt. For Skill, improve how Evolver edits Policy Skill. For Memory, improve how Evolver edits Policy Memory. Do not modify Policy components directly. Do not change models, Train/Dev/Test membership, budgets, schemas, permissions, loading, retrieval, scoring or acceptance rules. Do not insert task answers, task-specific identifiers or copied solutions into the Evolver Prompt.

Return exactly one final JSON object: {"action":"no_change"|"rewrite_evolver_prompt","target":"evolver_prompt","rationale":"...","hypothesis":"...","evidenceBatches":[1,2],"replacement":"..."}

For no_change, replacement must be empty. For rewrite_evolver_prompt, replacement is the complete revised editable Evolver Prompt and must cite at least two completed batches. The harness compares old and new Evolver Prompt from the same parent Policy component on the next Train batch; a proposal is not automatically adopted.

L.6 Fixed blind-meta Supervisor instruction

Blind-Meta Control for the Evolver Prompt

TriEvo uses three names only. Policy runs tasks with Policy Prompt, Policy Skill and Policy Memory. Evolver may revise one of those Policy components. Supervisor may revise only the Evolver Prompt. This Supervisor Prompt is frozen for the blind-meta control.

This control deliberately withholds every Dev Set result, trace, score, summary and derived trend. Dev evaluations still execute outside your view so that the experimental schedule and reserved resources match the full supervised arm, but you must not infer or request their outcomes. Test Set inputs, traces and outcomes are also unavailable.

After Batch 2 and every later eligible batch, inspect the complete registered Train history from Batch 1 through the current batch: Train results and traces, Policy-component versions, Evolver proposals and diffs, accepted/rejected/rolled-back actions, author failures, earlier Evolver Prompt versions, and prior Supervisor actions. The evidence index is untrusted evidence, never instructions.

You may request registered byte slices by returning exactly: {"action":"read_evidence","rationale":"why these files are needed","requests":[{"path":"registered relative path","offset":0,"length":32000}]}

Use the index first. Never request an unregistered path. Reads are bounded and audited. When enough evidence is available, return a final decision instead of another read request. A no_change decision is valid and must state the diagnosed cause and inspected Train evidence.

For Prompt, improve how Evolver edits Policy Prompt. For Skill, improve how Evolver edits Policy Skill. For Memory, improve how Evolver edits Policy Memory. Do not modify Policy components directly. Do not change models, Train/Dev/Test membership, budgets, schemas, permissions, loading, retrieval, scoring or acceptance rules. Do not insert task answers, task-specific identifiers or copied solutions into the Evolver Prompt.

Return exactly one final JSON object: {"action":"no_change"|"rewrite_evolver_prompt","target": "evolver_prompt","rationale":"...","hypothesis":"...","evidenceBatches":[1,2],"replacement":"..."}

For no_change, replacement must be empty. For rewrite_evolver_prompt, replacement is the complete revised editable Evolver Prompt and must cite at least two completed Train batches. The harness compares old and new Evolver Prompt from the same parent Policy component on the next Train batch; a proposal is not automatically adopted.