IMPROVEMENT BY INHERITANCE: MEASURING RECURSIVE SELF-IMPROVEMENT IN CODING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-improving coding agents are usually judged by the best agent retained during search. But a lineage can get better without getting better at self-improvement: a good edit to the improver can be selected once and then inherited unchanged. We distinguish an improver's *child quality*, how good the task agents it writes are, from its *successor quality*, how good the improvers it writes are. A lineage comparison mostly measures inheritance when the improver rarely edits itself; we formalize this decomposition and introduce *nested replay*, which crosses candidate improvers on common starting states and scores the improvers they produce on one common evaluation bank. On the released Hyperagents history, the selected self-edit raises child quality from 0.48 to 0.86 on common states but does not detectably raise successor quality: the next generation's gain matches the gain in child quality almost exactly (0.29 against 0.30), because the edited improver almost never edits itself. The assay is sensitive: a modifier built to act on evidence about the improvement procedure yields a clear gain in successor quality under the same design. Controlled experiments trace the gain in child quality to evidence routing: the edit sends the improver to evaluation results it otherwise ignores, and helps only when those results carry information the task lacks. Claims of recursive self-improvement need measurement at the recursive order they claim.

1 INTRODUCTION

A self-improving coding agent is a loop. A task agent solves benchmark instances. An improver, itself a coding agent, edits the task agent's code and possibly its own. The edited agent is evaluated, and an archive selects the next parent. Two prominent systems with public code and released histories are the Darwin Gödel Machine (Zhang et al., 2026a) and Hyperagents (Zhang et al., 2026b), and both report curves that rise from a weak initial agent to a substantially stronger one, with ablations showing that removing the archive or the improver's ability to edit itself hurts.

A lineage can get better without getting better at self-improvement. Suppose a self-edit turns the improver M_0 into M_1, which writes better task agents. If M_1 never edits itself, every later generation runs M_1 and the lineage keeps its gain, yet M_1 may be no better than M_0 at producing a new improver. The lineage then improves by *inheritance*: it carries a better improver forward without a better ability to improve improvers. We therefore separate an improver's *child quality*, how good the task agents it writes are (order 0), from its *successor quality*, how good the improvers it writes are (order 1). A rising archive curve measures neither directly: historical gains combine changes in the improver with changes in the states that selection presents to it.

We ask three questions of the one improver edit that the released Hyperagents history selected. *Did it make the improver better at writing task agents?* Common-state replay, which applies both versions to the same parents and evidence, says yes: $+0.38$ on twelve states, and positive on two further banks (Section 5). *Did it make the improver better at writing its successor?* Nested replay says not detectably: the next generation's gain is inheritance, and a positive control shows that the assay detects a real gain in successor quality (Section 6). *Why did the edit help at all?* It routes the improver to evaluation results that it otherwise ignores, which helps when those results carry information the task lacks and the improver would not read them unprompted (Section 7). Figure 1

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.

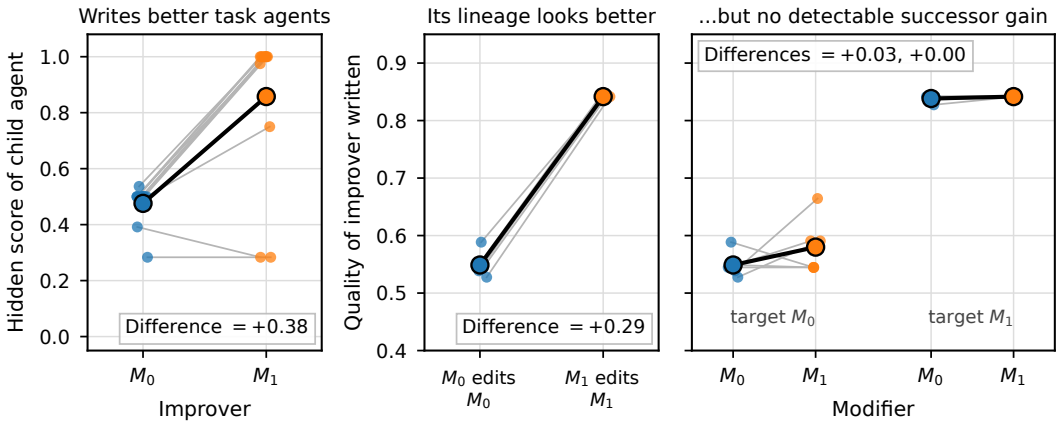

Figure 1: The selected self-edit writes better task agents but is not detectably a better improver. *Left:* hidden score of the children that the initial improver M_0 and the selected edit M_1 write from the same twelve task states (thin lines: states; large markers: means). *Middle:* under natural self-application the next generation's improver is better, because M_1 rarely edits itself and is inherited. *Right:* with the target held fixed, we detect no advantage for M_1 in the quality of the improvers it produces (six meta-states per target).

shows the headline result per state, and it separates what the archive curve merges: M_1 is a better improver of task agents, its lineage is better because M_1 is inherited, and M_1 is not detectably better at producing future improvers.

We make three contributions.

- **Measurement.** We distinguish child quality from successor quality, show that a lineage comparison measures mostly inheritance when self-edits are rare (Theorem 1), and introduce nested replay, which identifies successor quality (Theorem 2).
- **Finding.** In Hyperagents, the selected improver edit strongly raises child quality but not detectably successor quality; the lineage gain is inherited. A second, independently produced edit behaves the same way, and a positive control shows the assay is sensitive.
- **Mechanism.** The gain in child quality comes from evidence routing, whose benefit tracks how much the evidence adds to the task specification and how rarely the improver reads it unprompted.

2 What self-improvement means

An improver is a coding agent together with its code; we identify it with that code, the operator file the loop can edit, and write $M_0, M_1, \ldots$ for its versions. An improver m receives a task state X (a parent task agent and its development evidence) and randomness ξ, and produces a child task agent $m(X, \xi)$ with hidden score $S \in [0, 1]$. For a declared distribution ν_0 of task states, its child quality is

$$Q^{(0)}(m; \nu_0) = \mathbb{E}_{X \sim \nu_0, \xi}\, S(m(X, \xi)).$$

Child quality asks whether an improver writes better task agents. By contrast, successor quality asks whether it writes a better improver. A *meta-state* Z holds one opportunity to modify an improver: a repository with the current task agent, the current improver code, the development evidence and the tools. Running the loop under modifier m from Z leaves a successor improver $U(m; Z, \zeta)$; if m does not edit the improver, U returns it unchanged, a legitimate outcome rather than missing data. For a declared distribution ν_1 of meta-states, the successor quality is

$$Q^{(1)}(m; \nu_1, \nu_0) = \mathbb{E}_{Z \sim \nu_1, \zeta}\, Q^{(0)}\big(U(m; Z, \zeta); \nu_0\big),$$

and the definition extends recursively to higher orders $Q^{(r)}$, although our experiments measure only orders 0 and 1. For a self-edit $m \to m'$ write $\Delta_0 = Q^{(0)}(m') - Q^{(0)}(m)$ and $\Delta_1 = Q^{(1)}(m') - Q^{(1)}(m)$; the two need not share a sign (Proposition 1, Appendix D, which also holds all proofs).

Self-lineage comparisons. The natural evidence for recursive improvement is a lineage comparison: let the new improver modify itself and compare it with the old improver modifying itself. This comparison is harder to read than it looks, because self-application changes two things at once: the modifier, and the target that the next generation inherits whenever it is not edited. For a modifier M_a and a *target* M_b, the improver in the meta-state, let Y_{ab} be the mean child quality of the improvers that M_a leaves when started from meta-states whose improver is M_b, and let p_{ab} be the probability that it edits the target. The self-lineage contrast is $Y_{11} - Y_{00}$; the modifier effects hold the target fixed, $\delta_b = Y_{1b} - Y_{0b}$.

A simple example. Suppose M_0 has child quality 0.55 and a self-edit produces M_1 with child quality 0.84, roughly the values in Section 6. If neither ever edits itself again, self-applying M_0 leaves an improver of quality 0.55 and self-applying M_1 leaves one of quality 0.84. The lineage comparison is $+0.29$, although the two modifiers are equally ineffective at producing new improvers: both simply return the target they inherited. Theorem 1 generalizes this to the case where self-edits occur occasionally.

Theorem 1 (A lineage gain splits into inheritance and editing)**.** *Let μ_{ab} be the mean child quality of the improvers that M_a leaves when it does edit M_b. Each cell satisfies $Y_{ab} = Q^{(0)}(M_b) + p_{ab}\,[\mu_{ab} - Q^{(0)}(M_b)]$. Hence, for the self-edit $M_0 \to M_1$,*

$$Y_{11} - Y_{00} = \underbrace{Q^{(0)}(M_1) - Q^{(0)}(M_0)}_{\textit{inheritance } (\Delta_0)} + \underbrace{p_{11}\,[\mu_{11} - Q^{(0)}(M_1)] - p_{00}\,[\mu_{00} - Q^{(0)}(M_0)]}_{\textit{edit term}}.$$

If neither improver edits a copy of itself, the self-lineage contrast equals Δ_0; if no modifier edits either target, $\delta_0 = \delta_1 = 0$. A lineage can therefore gain the full order-0 improvement with no order-1 improvement.

Corollary 1 (Inheritance dominance)**.** *Since $Q^{(0)} \in [0,1]$, $|Y_{11} - Y_{00} - \Delta_0| \le p_{11} + p_{00}$ and $|\delta_b| \le p_{1b} + p_{0b}$. When self-edits are rare, a lineage contrast is dominated by inheritance and the order-1 effects are small, whatever the quality of the edits.*

Theorem 1 explains why self-application can mislead: when self-edits are rare, almost all of a lineage gain comes from inheriting the better target, whatever the improver would do if it edited. In other words, a lineage can improve even though the mechanism that modifies it does not.

3 How to measure it

Nested replay holds both stages fixed: candidate improvers act on the same meta-states, and every improver they produce is then evaluated on the same downstream bank.

Theorem 2 (Nested replay identification)**.** *Let $Z_1, \dots, Z_J$ be meta-states from a declared bank ν_1 and $X_1, \dots, X_K$ task states from a declared bank ν_0, both fixed in advance and independent of everything generated. Within each meta-state, assign modifiers $m \in \{m_0, m_1\}$ to execution slots at random and draw successor improvers $\tilde{o}_{jmr} = U(m; Z_j, \zeta_{jmr})$, $r = 1, \dots, R$. Evaluate every successor on the same bank, $\hat{q}(\tilde{o}) = K^{-1}\sum_k S(\tilde{o}(X_k, \xi_k))$ with fresh ξ_k. Then*

$$\hat{\Delta}_1 = \frac{1}{J}\sum_j \left[\tfrac{1}{R}\textstyle\sum_r \hat{q}(\tilde{o}_{jm_1r}) - \tfrac{1}{R}\sum_r \hat{q}(\tilde{o}_{jm_0r})\right]$$

is unbiased for $Q^{(1)}(m_1; \hat{\nu}_1, \nu_0) - Q^{(1)}(m_0; \hat{\nu}_1, \nu_0)$ on the empirical bank, and for the population contrast when the Z_j are drawn from ν_1. The independent unit is the meta-state; the K evaluation states and the R replicates are not independent treatment units.

Operationally, nested replay is two experiments in sequence. First, each candidate modifier runs on the same meta-states with the same target improver, producing candidate successor improvers. Second, ignoring how those successors were produced, each is evaluated on the same bank of task states. A modifier has higher successor quality only if the improvers it produces do better in this second stage. Holding the target fixed removes inheritance, and holding the evaluation bank fixed removes differences in the tasks presented downstream. Common-state replay is the order-0 case, which crosses improvers with the same task states. The distinction also runs deeper than order 1, since measuring order $r-1$ cannot establish improvement at order r.

Evaluation	What varies	What is held fixed	What it identifies
Best-so-far archive curve	improver, states, selection	nothing	archive performance
Parent–child change	improver and parent state	little	neither order
Common-state replay	improver	task states, evidence, budget	child quality (Δ_0)
Self-lineage comparison	modifier and inherited target	meta-states	inheritance plus edit term (Thm. 1)
Nested replay	modifier	target, meta-states, evaluation bank	successor quality (δ_b; Thm. 2)

Table 1: What common evaluations of self-improvement identify. Only the last two rows hold enough fixed to separate the two orders, and only nested replay isolates the modifier.

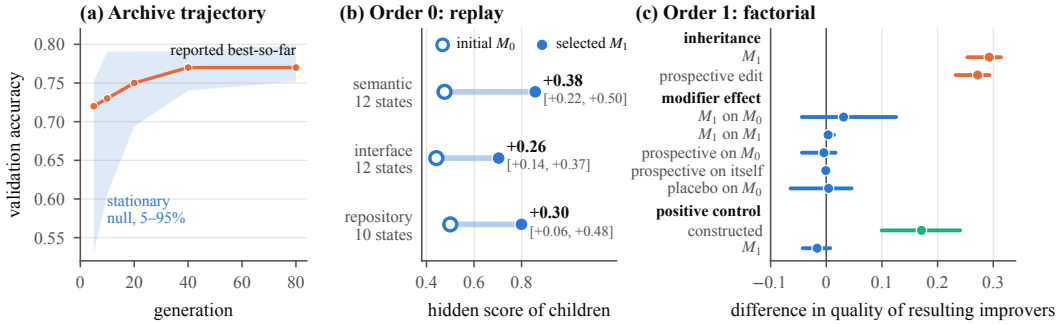

Figure 2: Three levels that an archive curve conflates. (a) The archive curve stays inside the band of a null in which the improver never changes. (b) M_1 improves child quality on all three banks (hollow: M_0, filled: M_1; 95% intervals). (c) The self-lineage contrast (orange) is large because M_1 is inherited; the modifier effects (blue) are near zero, while the positive control (green) is clearly detected.

Theorem 3 (Lower orders do not identify higher orders)**.** *For every integer $r \geq 1$ there exist improvers A, B with $Q^{(j)}(A) = Q^{(j)}(B)$ for every $j < r$ and $Q^{(r)}(A) \neq Q^{(r)}(B)$. Moreover, for any selection and promotion rule that depends on the history only through observed task scores, the self-modifying processes started from A and from B induce the same distribution over all task-level observables (child scores, promotions and archive trajectories) through every horizon $H < r$. Hence no measurement of order below r, and no such observable through horizon $H < r$, identifies the sign of an order-r contrast: in particular Δ_0 does not identify Δ_1.*

The key consequence is methodological: different recursive claims require different interventions (Table 1). An archive trajectory does not identify even Δ_0 (Propositions 1A and 1B): the same score trajectory can arise with or without a change in the improver.

4 What the Released Histories Show

We analyze all 71 released Hyperagents runs across 15 configurations, with evaluation reports, lineage, every generation's diff and 1,550 improver transcripts, together with the released DGM and ADAS histories (Appendix A). Let g_t be the expected child score of the improver in use at time t and ρ_t the distribution of states the archive presents to it. Historical child quality $V_t = \mathbb{E}_{\rho_t} g_t$ satisfies

$$V_t - V_s = \underbrace{\mathbb{E}_{\rho_s}[g_t - g_s]}_{\text{improver change}} + \underbrace{\mathbb{E}_{\rho_t} g_t - \mathbb{E}_{\rho_s} g_t}_{\text{state-distribution change}},$$

and a history observes only the sum. In the released histories, the second term does much of the work. First, on the balanced paper-review task an agent that predicts one label for every paper scores 0.50 regardless of its reviewing ability, and 163 of 369 fully evaluated nodes are collapsed

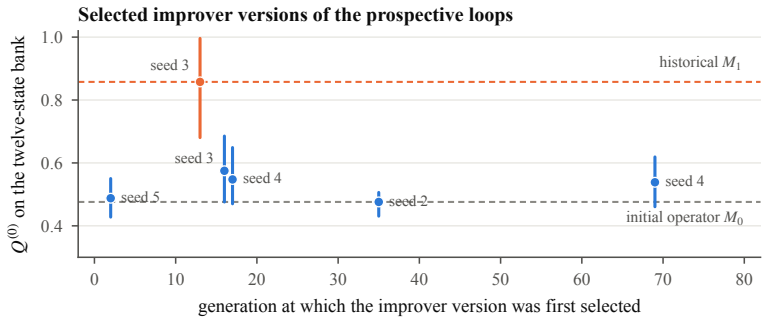

Figure 3: Prospective improver versions. Each version selected as a parent in five self-referential loops is scored on the twelve-state bank ($Q^{(0)}$ with a parent-bootstrap interval) and placed at the generation of its first selection. Dashed lines mark the initial improver and the historical selected edit under the same harness. Only the version that routes the improver to the evaluation results reaches the historical edit's level; a specification-only version introduced earlier in the same seed was selected next.

onto one label; transitions into collapse are almost all regressions and transitions out almost all improvements (Table 8). Second, the order of scores carries little information: in 119 of 141 run-by-checkpoint comparisons the observed best-so-far lies inside a 90% band that permutes the order of the run's scores, and a null in which the improver never changes, holding the realized parents fixed, covers it at 64 of 69 checkpoints. The released DGM and ADAS histories show the same limits. Taken together, these checks mean the curve cannot say whether the improver improved, so the next sections intervene on the improver directly. Figure 2 sets the three levels side by side: the archive curve, child quality on three banks (Section 5), and successor quality (Section 6).

5 Order 0: The selected edit writes better task agents

We apply the initial improver M_0 and its three proposed edits in the released source history, at generations 1, 3 and 12, to the same twelve parent/evidence states, with the same task setup and 40-call budget, two replicates and randomized execution order. Only M_1 was selected and has descendants. At generation 1 it replaced a one-line instruction with a briefing that names the path to the evaluation results and says to review them first.

M_1 raises hidden score from 0.476 to 0.858, a gain of 0.382 (family-clustered interval $[0.221, 0.500]$), positive for ten of twelve parents. It opens the report in every run; M_0 does so in at most one of 24. An unselected sibling, M_3, which also tells the improver to read the report, is comparably strong under replay (0.940), showing that the selected M_1 was not the only strong edit among its siblings. The effect also transfers to two deliberately different state banks. On ten SWE-bench Verified instances without issue text, M_1 raises the primary score from 0.500 to 0.800 ($+0.300$, repository-clustered interval $[0.062, 0.476]$) and resolves 11 of 20 applications against none; on twelve solvers for a 14-step state-tracking task, which fail because their output is truncated, it raises hidden accuracy from 0.441 to 0.704 ($+0.262$ $[0.138, 0.374]$). The lead persists over four steps of repeated application (Appendix C).

A prospective loop. The released history contains only one selected improver edit, so we also ran five new 80-generation self-improving loops on eight problem families disjoint from the evaluation bank, starting from M_0 and selecting parents by the released rule on development score (Appendix C). Six new improver versions were selected, and we scored each on the twelve-state bank (Figure 3). Only one substantially raised child quality: an independently generated edit that, like M_1, tells the improver to read the evaluation results, and that scores 0.858, the level of M_1. The other five score between 0.476 and 0.575, with intervals that cover M_0. The evidence-routing edit appeared in only one of the five seeds, and the other four archives nonetheless reached a perfect development score while still running M_0. A rising archive thus coexisted with little or no improvement of the improver itself.

Contrast	Improver edits	Estimate	95% interval	p
M_1 vs M_0, target M_0 (δ_0)	4/36 vs 6/36	$+0.031$	$[-0.044, +0.126]$	0.31
M_1 vs M_0, target M_1 (δ_1)	0/36 vs 3/36	$+0.003$	$[+0.000, +0.014]$	0.50
interaction $\Gamma = \delta_1 - \delta_0$		-0.028	$[-0.125, +0.044]$	0.38
self-lineage $Y_{11} - Y_{00}$ (inheritance)	0/36 vs 6/36	$+0.293$	$[+0.253, +0.314]$	0.03
prospective edit vs M_0, target M_0	0/36 vs 6/36	-0.004	$[-0.044, +0.017]$	0.88
prospective edit vs M_0, target prospective edit	0/36 vs 8/36	-0.001	$[-0.007, +0.002]$	1.00
two other prospective edits vs M_0, target M_0	0/72 vs 6/36	-0.004	$[-0.044, +0.017]$	0.88
placebo (renders M_0) vs M_0, target M_0	6/36 vs 6/36	$+0.004$	$[-0.064, +0.046]$	0.91
positive control: constructed modifier vs M_0	35/42 vs 11/42	$+0.171$	$[+0.099, +0.240]$	0.02
positive control: M_1 vs M_0	3/42 vs 11/42	-0.016	$[-0.042, +0.007]$	0.25

Table 2: Modifier $\times$ target factorial and positive control. Six semantic meta-states; positive control, seven states. Improver edits: runs that changed the instruction rendered by the operator file, first arm vs second. Quality on the four semantic evaluation states: M_0 0.545, M_1 0.842, prospective edit 0.821. Intervals invert the exact sign-flip test over meta-states.

6 Order 1: the selected edit does not detectably write better successors

Order-0 replay shows that M_1 writes better task agents. The order-1 question is different: does M_1 write better future improvers? Natural self-application cannot answer this, because M_1 is both the modifier and, unless edited, the improver the next generation inherits. We therefore cross modifier and target: M_0 and M_1 each modify copies of both M_0 and M_1 on the same six semantic-fault meta-states (Python functions that each miss one unstated convention of their specification), six times per cell under a pre-registered protocol, and every resulting improver is scored on the same four evaluation states (Appendix B). The meta-state is the unit of inference, and Table 2 reports the two modifier effects, the difference between them, and the self-lineage contrast.

The data show the pattern that Theorem 1 predicts. On these states M_1 raises child quality from 0.545 to 0.842, and the self-lineage contrast is $+0.293$, within 0.004 of that order-0 gain. Corollary 1 bounds the gap by $p_{11} + p_{00} = 6/36$ before any child is scored: the lineage gain is inheritance. The modifier effects are small, $+0.031$ $[-0.044, 0.126]$ on target M_0 and $+0.003$ on target M_1. M_1 never edited a copy of itself (0 of 36 runs), and M_0 edited M_1 in 3 of 36 runs, each time slightly lowering it. When M_1 did edit M_0 (4 of 36 runs) the resulting improvers were strong, so $\hat{\delta}_0$ is positive, but its interval includes zero.

What the order-1 result resolves. Most of the uncertainty in an order-1 contrast comes from whether a modifier edits the target at all, not from noise in rescoring the improvers it writes (Appendix B), so we use six modifier replicates per cell. With six meta-states, the exact sign-flip test reaches $p < 0.05$ only when all six differences share a sign, so the factorial resolves large changes in successor quality, not small ones. Its intervals exclude modifier effects above 0.13 on target M_0 and 0.014 on target M_1, far below the 0.29 lineage gain. Small improvements in successor quality remain possible, but not one large enough to explain the lineage improvement.

Positive control. On seven further meta-states whose evidence records that the current improver had repeatedly failed, a modifier explicitly asked to diagnose the improvement procedure as well as the task edits the improver in 35 of 42 runs, against 11 for M_0, mostly to send the next improver to the evaluation results, and improves successor quality by $+0.171$ $[0.099, 0.240]$, positive on all seven states. M_1, on the same states, edits the improver in 3 of 42 runs (-0.016).

Replication. An independently produced evidence-routing edit, the first improver edit selected by a prospective self-improving loop, again raises child quality on these states (to 0.821) but never edits an improver as a modifier (0 of 72 runs), and again shows no detectable gain in successor quality (-0.004 and -0.001, Table 2). Corollary 1 explains the pattern: order-1 effects are bounded by edit rates, and evidence-routing improvers direct the loop to the task and almost never edit the improver,

Experiment	Evidence channel	Information added	Read unprompted	Routing effect
Interface fault, MiMo	unparsed count	high	1/24	$+0.185$
Semantic fault, MiMo	expected/actual	high	0/24	$+0.303$
Semantic fault, DeepSeek	expected/actual	high	21/24	-0.009
Cross-model, non-readers (3)*	expected/actual	high	$\leq$2/12	$+0.19$ to $+0.34$
Cross-model, readers (2)	expected/actual	high	$\geq$10/12	-0.11 to 0.00
Information dose, low	expected/actual	high	0/12	$+0.372$
Information dose, medium	expected/actual	partly in spec	0/12	$+0.046$
Information dose, high	expected/actual	in spec	0/12	-0.067
Repositories, no issue	test tracebacks	high	0/20	$+0.307$
Repositories, with issue	test tracebacks	largely in issue	0/20	$+0.043$
Repositories 2×2, no issue	test tracebacks	high	0/20	$+0.253$
Repositories 2×2, with issue	test tracebacks	largely in issue	0/20	$+0.013$

Table 3: Evidence-routing cells on the two-determinant grid. Information added: what the report reveals beyond the specification or issue the improver already holds. Read unprompted: the share of blind runs that opened the report. Effects are paired forced-read (or briefing) minus blind contrasts on the hidden endpoint. Large effects occur only in the high-value, low-access cells. *Kimi excluded as provider-confounded.

even when the evidence implicates it. The positive-control modifier escapes the bound by editing in most runs.

7 Why did the edit help? Evidence routing

The selected edit primarily changes where the improver looks: it directs the improver to the evaluation results. In the 1,550 released transcripts, self-improving improvers opened those results before their first edit in 87% and 93% of generations, and fixed improvers in none of 800. We call this mechanism evidence routing, and state its predicted determinants before the experiments: the benefit of routing a piece of evidence should grow with how much it reveals beyond what the improver already has, and with the improver's probability of not reading it unprompted. Table 3 places every experiment on the grid of these two determinants. All experiments use a fixed set of parents, two replicates per parent and condition unless stated, MiMo-V2.5 as the underlying model with identical tools and budget across arms, and hidden evaluation items that never enter the repository.

Routing helps. We compare three instructions: the original one-line instruction, which we call *blind*; the M_1 briefing; and the one-line instruction plus a single sentence pointing to the report, which we call *forced read*. On the twelve state-tracking solvers, the M_1 briefing raises hidden accuracy by 0.185 over the one-line instruction (interval $[0.064, 0.323]$); briefed improvers raised the token budget in 11 of 24 runs, blind improvers once. On twelve semantic-fault solvers, briefed children pass 0.305 more hidden cases ($[0.201, 0.408]$; ten of twelve parents positive, none negative). None of the 24 blind improvers opened the report. The forced-read pointer, "Read `/repo/eval_results/dev/report.json` before making any edits", does as well as the full briefing ($+0.303$ against $+0.305$) with half the tool calls.

Does the evidence add information? A randomized experiment varies what the specification already says across the same twelve semantic parents: minimal context, the unresolved decision identified, or the evaluator's convention stated. Forced reading helps by $+0.372$ $[0.207, 0.522]$, $+0.046$ $[-0.179, 0.279]$ and -0.067 $[-0.200, 0]$ respectively (low-minus-high interaction $+0.439$, sign test $p = 0.021$). Real repositories show the same law. On ten SWE-bench Verified instances with the issue text withheld, forced-read children score 0.307 above blind children ($[0.182, 0.425]$) and resolve the bug in 10 of 20 runs against one. A same-instance 2×2 on 20 further instances crosses issue text with routing: without the issue, forced reading raises the score from 0.550 to 0.803; with it, forced reading adds $+0.013$ $[-0.135, 0.133]$, while the issue alone raises the blind arm by $+0.274$ (interaction $+0.241$, instance bootstrap $[0.038, 0.450]$, repository-clustered $[-0.066, 0.487]$; Figure 4c).

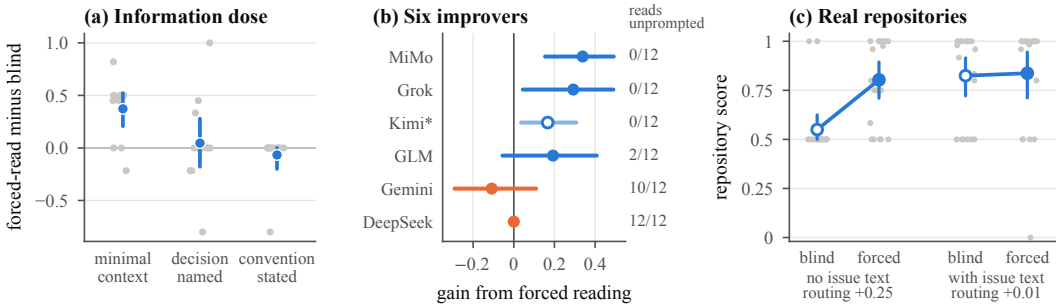

Figure 4: Evidence routing follows information redundancy. (a) Forced-read minus blind on twelve semantic parents as the specification supplies none, part or all of the diagnostic information (dots: parents; blue: mean and 95% interval). (b) Gain from forced reading for six improvers, with the share of blind runs that read the report unprompted; blue: models that rarely do, orange: models that usually do; *provider-confounded (Section 7). (c) Same-instance 2×2 on 20 repository instances (dots: instances; hollow: blind, filled: forced read): routing raises the score without the issue text and adds nothing with it; interaction $\Gamma = +0.24$ $[0.04, 0.45]$.

Would the improver read it anyway? The second determinant is whether the improver finds the evidence on its own. A four-arm panel (blind, forced-read, report inline, both) on the same twelve parents covers six underlying models. MiMo, GLM, Kimi and Grok open the report in at most two of twelve blind runs and gain 0.17 to 0.34 from forced reading (MiMo and Grok with adjusted $p \leq 0.015$; Kimi's contrast is confounded by provider routing, Appendix B); DeepSeek and Gemini open it in 12 and 10 blind runs and gain nothing (0.000 and -0.109). Once the report is inline, the pointer adds nothing. Taken together, routing helps when the evidence adds information and the improver would not otherwise read it, and not otherwise. The gain comes less from changing the improver's reasoning than from changing which information reaches it.

8 Discussion

Related work: self-improvement systems and benchmarks. Self-referential self-improvement descends from the Gödel machine (Schmidhuber, 2007). Recent language-model systems include the Darwin Gödel Machine (Zhang et al., 2026a), the Huxley–Gödel Machine (Wang et al., 2026b), Hyperagents (Zhang et al., 2026b), the Self-Improving Coding Agent (Robeyns et al., 2025), the Gödel Agent (Yin et al., 2025), the Self-Taught Optimizer (Zelikman et al., 2024), MetaRSI (Tan et al., 2026), the Red Queen Gödel Machine (Iacob et al., 2026) and Meta^n (Kim et al., 2026). Promptbreeder and MetaSkill-Evolve make parts of the improvement procedure itself evolvable (Fernando et al., 2024; Wang et al., 2026c), and Generalized Agent Iteration formalizes the modifier alongside the agent (Tang et al., 2026), while ADAS, AgentBreeder and AlphaEvolve search over agent designs, scaffolds or programs (Hu et al., 2025; Rosser & Foerster, 2025; Novikov et al., 2025). Recent benchmarks isolate improvement of experience, training algorithms and data strategies (Xue et al., 2026; Chi et al., 2026; Meng et al., 2026), and Wang et al. (2026a) study statistical limits on capacity growth under self-modification. MetaRSI performs a closely related common-target comparison: it forks the same successor system and compares an original and a meta-updated improver under the same budget (Tan et al., 2026), and AIE-Bench benchmarks how well a meta-agent improves a target agent, including itself (Mishra et al., 2026). In our terminology both measure an improver's ability to improve a target; nested replay asks one level further, whether an improver produces better future improvers.

Related work: selection, evaluation and variation operators. Selecting the maximum of noisy estimates produces the optimizer's curse (Smith & Winkler, 2006), correlating a baseline with its change creates mathematical coupling (Oldham, 1962; Tennant et al., 2023), agent scores may be unreliable (Rabanser et al., 2026; Patel et al., 2026; Ndzomga, 2026), staged evaluation resembles racing (Maron & Moore, 1993; Jamieson & Talwalkar, 2016; Li et al., 2018) with adaptive-data-analysis risks (Dwork et al., 2015), and architecture search has been checked against random base-

lines (Real et al., 2019; Li & Talwalkar, 2020; Yang et al., 2020). Self-adaptation, hyper-heuristics and learning-to-optimize separate a selected individual from a variation operator (Bäck, 1996; Burke et al., 2013; Andrychowicz et al., 2016); adaptive operator selection credits a fixed operator set from offspring performance (Burke et al., 2013), whereas in nested replay the outcome of the first intervention is itself an improver scored in a second common-state experiment.

Clade metaproductivity. The Huxley–Gödel Machine shows that benchmark performance is a weak guide to self-improvement potential and aggregates descendants' task performance under a search process (clade metaproductivity) (Wang et al., 2026b). Neither that quantity nor successor quality determines the other (Proposition 7): the first is a property of the modifier together with the search policy and the lineage it induces, the second of the modifier on declared states.

Implication for evaluation. A claim that an agent improves its task-solving procedure requires an order-0 comparison on common task states. A claim that it improves its own improvement procedure requires an order-1 comparison: hold the target improver and the meta-state fixed, vary the modifier, and score the resulting improvers on a common downstream bank. Natural self-application is not enough, because it changes the modifier and the inherited target together. More generally, the intervention must match the recursive order of the claim.

Implication for system design. Improving task agents and improving the improver are different optimization targets. A loop can become much better at the first by routing useful evidence to an unchanged improvement procedure, and selection on child scores rewards exactly that. Our experiments suggest that the second requires edits to the improver itself, judged by the improvers they produce: evidence-routing improvers rarely edit the improver, whereas the positive-control modifier edits it in most runs and yields a measurable order-1 gain.

Limitations. The order-1 factorial uses a small constructed meta-state bank and one improver backbone, so it resolves large rather than subtle changes in successor quality. Order 2 and beyond remain unmeasured, and the repository panels are small. We study short-horizon self-improvement in released coding-agent loops, not open-ended recursive self-improvement.

Conclusion. A self-improving lineage may contain better agents, a better improver of task agents, or a better producer of future improvers. These are different claims. In Hyperagents, the selected self-edit improves the second but not detectably the third: later generations benefit by inheritance. Evaluations of recursive self-improvement should therefore intervene at the recursive level they claim has improved.

Reproducibility statement

The appendix gives the design, analysis and outcome of every experiment, including the preregistered protocols and the correction record (Appendix B.1). The code, protocols, state and meta-state banks, run logs and transcripts will be released with the paper, and every number in the paper is regenerated by a released script from those files.

AI use statement

Language models are the object of study: MiMo-V2.5 and six other models serve as improvers and task solvers, and their outputs are the measured data. Separately, we used an AI coding assistant to help write the experiment and analysis code and to assist in writing the proofs. Additionally, we used generative AI tools to draft and edit text. We reviewed all AI-assisted work: every reported number was checked against the released run logs and is regenerated by a released script, and the harness is covered by invariant tests. We take responsibility for the final content of this work, including text, claims and code produced with the aid of generative AI.

REFERENCES

Marcin Andrychowicz, Misha Denil, Sergio Gómez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Thomas Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, 1996. doi: 10.1093/oso/9780195099713.001.0001.

Edmund K. Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, 2013. doi: 10.1057/jors.2013.71.

Yizhe Chi, Wenyi Li, Deyao Hong, Xiaoqiu Wang, Mingju Gao, Kaisen Yang, Bingxiang He, Youjie Zheng, Calvin Xiao, and Qinhuai Na. AI4AI-Bench: Benchmarking LLM agents in algorithmic design for recursive self-improvement. *arXiv preprint arXiv:2608.20318*, 2026.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. Preserving statistical validity in adaptive data analysis. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing*, pp. 117–126, 2015. doi: 10.1145/2746539.2746580.

Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *PMLR*, pp. 13481–13544, 2024.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations (ICLR)*, 2025.

Alex Iacob, Andrej Jovanović, William F. Shen, Daniel Burkhardt, Meghdad Kurmanji, Nurbek Tastan, Lorenzo Sani, Niccolò Alberto Elia Venanzi, Ambroise Odonnat, Zeyu Cao, Bill Marino, Xinchi Qiu, and Nicholas D. Lane. The Red Queen Gödel Machine: Co-evolving agents and their evaluators. *arXiv preprint arXiv:2606.26294*, 2026.

Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 51 of *PMLR*, pp. 240–248, 2016.

Zae Myung Kim, Young-Jun Lee, Seungyeon Jwa, and Dongyeop Kang. Meta^n: Recursive self-improvement through emergent depth. *arXiv preprint arXiv:2608.24735*, 2026.

Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *Proceedings of the 35th Uncertainty in Artificial Intelligence Conference (UAI)*, volume 115 of *PMLR*, pp. 367–377, 2020.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.

Oded Maron and Andrew W. Moore. Hoeffding races: Accelerating model selection search for classification and function approximation. In *Advances in Neural Information Processing Systems*, volume 6, 1993.

Fanqing Meng, Lingxiao Du, Qiguang Chen, Ziqi Zhao, Haocheng Lu, Mengkang Hu, and Michael Qizhe Shieh. RSIBench-Data: Benchmarking data-centric research for recursive self-improvement. *arXiv preprint arXiv:2607.25886*, 2026.

Abhishek Mishra, Selvam Palanimalai, Yogendra Manawat, Samuel Verboomen, Prannay Hebbar, Damir Vrabac, Deepak Nathani, Sumeet Motwani, Kunal Bhatia, and Vignesh Baskaran. AIE-Bench: Benchmarking agents that build agents. In *ICML 2026 Workshop on Combining Theory and Benchmarks: Towards A Virtuous Cycle to Understand and Guarantee Foundation Model Performance*, 2026.

Franck Ndzomga. Efficient benchmarking of AI agents. *arXiv preprint arXiv:2603.23749*, 2026.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

P. D. Oldham. A note on the analysis of repeated measurements of the same subjects. *Journal of Chronic Diseases*, 15(10):969–977, 1962. doi: 10.1016/0021-9681(62)90116-9.

Dhaval C. Patel, Kaoutar El Maghraoui, Shuxin Lin, et al. Beyond static leaderboards: Predictive validity for the evaluation of LLM agents. *arXiv preprint arXiv:2606.19704*, 2026.

Stephan Rabanser, Sayash Kapoor, Peter Kirgis, Kangheng Liu, Saiteja Utpala, and Arvind Narayanan. Towards a science of AI agent reliability. *arXiv preprint arXiv:2602.16666*, 2026.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4780–4789, 2019. doi: 10.1609/aaai.v33i01.33014780.

Maxime Robeyns, Martin Szummer, and Laurence Aitchison. A self-improving coding agent. *arXiv preprint arXiv:2504.15228*, 2025.

J Rosser and Jakob Foerster. AgentBreeder: Mitigating the AI safety risks of multi-agent scaffolds via self-improvement. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025. doi: 10.52202/085713-5003.

Jürgen Schmidhuber. Gödel machines: Fully self-referential optimal universal self-improvers. In Ben Goertzel and Cassio Pennachin (eds.), *Artificial General Intelligence*, Cognitive Technologies, pp. 199–226. Springer, 2007. doi: 10.1007/978-3-540-68677-4_7.

James E. Smith and Robert L. Winkler. The optimizer's curse: Skepticism and postdecision surprise in decision analysis. *Management Science*, 52(3):311–322, 2006. doi: 10.1287/mnsc.1050.0451.

Zihan Tan, Leixin Sun, Zitong Shi, Yitao Liu, Jiajun Wu, Nathaniel Brooks, Jiaru Qian, Xiaoran Shang, Suyuan Huang, Yi Ding, et al. MetaRSI/RSI^2: A meta-recursive self-improving system for recursive self-improving systems themselves. *arXiv preprint arXiv:2609.06396*, 2026.

Hongyao Tang, Yi Ma, Pengyi Li, and Yifu Yuan. Generalized Agent Iteration: One formal framework for iterative policy improvement and recursive self-improvement. *arXiv preprint arXiv:2609.13406*, 2026.

Peter W. G. Tennant, Georgia D. Tomova, Eleanor J. Murray, Kellyn F. Arnold, Matthew P. Fox, and Mark S. Gilthorpe. Lord's 'paradox' explained: the 50-year warning on the use of 'change scores' in observational data. *arXiv preprint arXiv:2302.01822*, 2023.

Charles L. Wang, Keir Dorchen, and Peter Jin. On the statistical limits of self-improving agents. *Transactions on Machine Learning Research*, 2026a. arXiv:2510.04399.

Wenyi Wang, Piotr Piękos, Nanbo Li, Firas Laakom, Yimeng Chen, Mateusz Ostaszewski, Mingchen Zhuge, and Jürgen Schmidhuber. Huxley-Gödel Machine: Human-level coding agent development by an approximation of the optimal self-improving machine. In *International Conference on Learning Representations (ICLR)*, 2026b.

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

Shuhan Xue, Zixin Ding, Yichen Shen, Yinjie Wang, Zhenfei Yin, Yingcheng Wu, Yuxin Chen, Mengdi Wang, and Ling Yang. PAST-Bench: Benchmarking the foundations of recursive self-improvement in personal agents. *arXiv preprint arXiv:2608.04003*, 2026.

Antoine Yang, Pedro M. Esperança, and Fabio M. Carlucci. NAS evaluation is frustratingly hard. In *International Conference on Learning Representations (ICLR)*, 2020.

Xunjian Yin, Xinyi Wang, Liangming Pan, Li Lin, Xiaojun Wan, and William Yang Wang. Gödel Agent: A self-referential agent framework for recursively self-improvement. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 27890–27913, 2025. doi: 10.18653/v1/2025.acl-long.1354.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-Taught Optimizer (STOP): Recursively self-improving code generation. In *Conference on Language Modeling (COLM)*, 2024.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Tjarko Lange, and Jeff Clune. Darwin Gödel Machine: Open-ended evolution of self-improving agents. In *International Conference on Learning Representations (ICLR)*, 2026a.

Jenny Zhang, Bingchen Zhao, Wannan Yang, Jakob Foerster, Jeff Clune, Minqi Jiang, Sam Devlin, and Tatiana Shavrina. Hyperagents. *arXiv preprint arXiv:2603.19461*, 2026b.

A Archive audit detail

Scoring effects and parent–child persistence. The paper-review task is a balanced accept/reject decision over 100 validation papers, so accuracy is bounded by $\min(p_a, g_a) + \min(p_r, g_r)$ for predicted mix (p_a, p_r) and ground truth (g_a, g_r), and an agent that predicts one label for every paper scores 0.50 regardless of its reviewing ability. We call a report *collapsed* if one label carries at least 0.95 of predicted mass or non-label strings exceed 0.05: 163 of 369 fully evaluated source nodes are collapsed, and matched parent–child pairs grouped by endpoint state (Table 8) show that transitions into collapse are almost all regressions and transitions out almost all improvements, with pooled correlation 0.81 between the delta and the change in ceiling. The edits are prompt rewrites that toggle a decision threshold ("default to reject," "look for reasons to accept"), and selection retains whichever toggle landed near 50%; the release's own non-label metric cannot see this. Restricting to balanced pairs, children retain some parent behavior (within-run slopes 0.04 to 0.62; Table 4) but agree with their parent only slightly more than with a random node from the same run, and the mean change is at or below zero in every family (Table 9) while solver-identical retests vary by only 0.005 to 0.038. The variation is behavioral, its mean is not positive, and selection over it can produce a rising curve.

Selection and archive trajectories. A best-so-far curve is invariant to parentage; what it can reveal is whether the order of scores carries information. We compare each run's observed best-so-far at fixed evaluated-node counts with a null that permutes the order of its full-evaluation scores (1,000 permutations), and regress child score on generation index within run, alone and with the parent's score as covariate.

Across 141 run-by-checkpoint comparisons, the observed best-so-far lies inside the 90% permutation band at 119, escaping mainly in the 200-generation continuation; child score has no trend in the source runs (-0.007, $p = 0.61$) and a $+0.06$ trend in the 200-generation runs that falls to $+0.02$ to $+0.03$ once the parent's score is controlled. The share of children promoted to full evaluation rises from 0.47 to 0.81 across generation quintiles under score-proportional selection and stays between 0.23 and 0.32 under random selection: selection retains rare favorable draws and shifts later parentage toward agents whose interfaces work, which raises the curve without improving modification outcomes at a fixed state.

What the curve identifies. The decomposition in Section 4 takes the earlier state distribution as its reference; averaging the two references gives $V_t - V_s = \frac{1}{2}(\mathbb{E}_{\rho_s} + \mathbb{E}_{\rho_t})[g_t - g_s] + \frac{1}{2}(\mathbb{E}_{\rho_t} - \mathbb{E}_{\rho_s})[g_t + g_s]$, and what follows holds for either. The analyses above concern the state-distribution term: label collapse, parent–child persistence, promotion and selection all change which states the operator is applied to, and can raise V_t with g_t fixed. Propositions 1A and 1B (Appendix D) are the formal statement that the trajectory alone cannot separate the terms; the same appendix proves the stationarity check, censoring bounds and distribution relativity used here. With $\tau(x) =$

		Persistence slope			Sibling	Excess agreement		
Family	Pairs	β	null 95%	p	ICC(1)	parent–child	random pair	retest
Hyperagents source	147	0.18	0.15	0.018	$0.41^{\dagger}$	0.16	0.12	0.43
DGM customized	69	0.35	0.22	0.008	$0.07^{\dagger}$	0.23	0.28	0.38
Improver edits selection	133	0.58	0.21	<0.001	$0.58^{\dagger}$	0.30	0.23	0.46
Random parent selection	19	0.04	0.46	0.46	$0.17^{\dagger}$	0.22	0.24	0.46
Transfer, transferred	144	0.40	0.14	<0.001	$-0.03^{\dagger}$	0.36	0.32	0.39
Continuation, source	569	0.59	0.09	<0.001	0.45	0.33	0.27	0.32
Continuation, transferred	736	0.62	0.08	<0.001	0.49	0.32	0.29	0.38
Strong-baseline init.	196	0.08	0.12	0.26	$0.18^{\dagger}$	0.31	0.30	–

Table 4: Parent–child persistence on balanced pairs. β is the within-run slope of child on parent accuracy; null 95% is the 95th percentile under shuffled parents. † marks an ICC not distinguishable from its shuffled null. Excess agreement is per-question agreement beyond that expected from the two accuracies.

$\mathbb{E}_{\xi}[S(M_1(x,\xi)) - S(M_0(x,\xi))]$, $\Delta(\nu') - \Delta(\nu) = \int \tau\,[d\nu' - d\nu]$: the $+0.382$ is a contrast on the twelve-state reference distribution, not a context-free property of M_1, and the same sign on other banks is a finding about those banks.

Stationary-operator nulls that hold the realized parent sequence fixed cover the observed best-so-far at 64 of 69 checkpoints, and the ten-question gate censors 5% to 77% of children, with imputations leaving the pooled mean change within 0.03 of zero (Appendix B); these limits are why Section 5 measures the operator directly.

Other released systems. DGM's histories show the same pattern without a one-label floor (Table 9, lower block): improvements and regressions nearly equal, mean change near zero, 45% to 62% of SWE-bench children never scored; in ADAS, the best search-fitness agent ranks first or second on test in two of five domains. Meta^{n} (Kim et al., 2026), whose meta-operation code is fixed by construction, is a positive control: over five seeds we ran ourselves, reevaluating 26 fixed chains on the full test shows that the validation-selected gain of 0.309 comes from replacing two initial programs that cannot be parsed; among executable starts it is 0.014 $[0, 0.047]$ (Appendix B).

B Secondary numerical detail

Table 7 lists every Hyperagents configuration covered.

Release acquisition. We read the released Hyperagents, DGM and Polyglot archives with length and checksum verification and executed no archived code; four loop source files were pinned at a fixed commit to interpret the logs. Release versions, acquisition scripts, integrity records and per-file hashes are in the code release.

Improver edits. Partitioning every diff by file, self-improving arms edited the improver in 1% to 15% of generations, and in the source runs only one such edit has descendants: the generation-1 briefing, which also prioritizes response parsing and prompting and raises the tool-call cap; the same instruction appears in all five packages selected for transfer. Reading evaluation results did not make historical edits net-positive (56 of 147 balanced pairs improved, 83 regressed). By arm, self-improving improvers edited the improver in 41 of 500, 75 of 500, 74 of 498, 6 of 500 and 46 of 1,996 generations. Four of the five transferred packages also add code that summarizes evaluation results into the improver's prompt. Fixed improvers spent 150 of 500 and 157 of 300 edits on the inert copy of their own file.

Transcripts. Self-improving improvers read the evaluation before their first edit in 433 of 500 source generations and 232 of 250 transferred generations, and first edited the solver in 399 and 181 of them; fixed improvers read it in 0 of 500 paper-review and 0 of 300 IMO-grading generations.

DGM and ADAS. On DGM's SWE-bench runs 45%, 54% and 62% of children were never scored; on Polyglot the released metadata scores are ten-task staged evaluations; 27 nodes also carry a released 50-task second-stage summary, which we do not use. ADAS consecutive changes on the search set split 11/9, 12/13, 10/12, 13/14 and 13/13 improved to regressed across five domains, with mean between -0.001 and $+0.003$; the agent with the best search fitness ranks 10th, 8th, 8th, 1st and 2nd on test, with a search-minus-test gap of 0.01 to 0.10.

Interventions. The matched experiments share one design: the same fixed parents, two replicates per parent per condition, identical tools, budget and feedback across arms, a development report the improver may read, hidden evaluation whose items never enter any repository (the repository experiment is the exception: its test patch is committed into the working tree), and parent-level analysis. Interface fault: 14-step state-tracking task, 150 hidden items, briefing effect $p = 0.017$, 8 of 12 parents positive; children with a raised token budget score about 0.9 in either arm. Semantic fault: eleven problem families, a 40-case report of input, expected, actual and status in both arms, 300 unseen hidden cases per family; briefed improvers inferred the convention in 18 of 24 runs against 3 of 24 blind, opened the report in 23 runs and changed the algorithm in one or two lines, and no child broke a case its parent passed; briefing $p = 0.002$, forced read $p = 0.004$ ($[0.183, 0.411]$). Repositories: both arms saw a report with the failing output of half the target tests; forced-read improvers opened a test file in 20 of 20 runs, blind improvers in 2 of 20; eight instances positive, none negative ($p = 0.008$); one forced-read run and no blind run broke a regression test. Repository eligibility required pure-Python projects, a difficulty label of at least fifteen minutes, a multi-hunk gold patch, at least two target tests, seeded shuffle and at most two instances per repository. Environments failed before the human patch and passed after it. The first three matched-parent experiments used deterministic alternating order, so their sign-flip tests require exchangeability of parent-level paired differences; replicates sharing a parent are never treated as independent. In the interface experiment, the development-set difference was 0.203, positive in both replicates, with a leave-one-parent-out range of 0.137 to 0.214. In the semantic experiment, forced-read minus briefed was 0.002 with interval $[-0.080, 0.083]$; no equivalence margin was specified.

B.1 Corrections and confirmatory reruns

Issue	Affected analysis	Correction	Reported result
Repository parser	SWE-bench intervention	Corrected parser and complete four-arm rerun	Confirmatory panel only
Report path	Cross-model panel	Corrected path and a fresh 336-run panel	Superseded by the corrected-harness panel
Test adapter	Meta^{n} audit	Corrected adapter and full recomputation	Corrected scores and null
Virtual-root symlink	Panels run after the report-path fix	Corrected substitution; replay, rollouts, dose and cross-model panels rerun	Corrected-harness runs reported; earlier runs archived

Table 5: Corrections affecting experimental results. Superseded runs are kept in the archived run logs.

Corrected harness. Table 5 lists the corrections. Every affected panel was rerun in full under the corrected harness, whose file hashes are recorded, using its original assignment; the other-bank panels, nested replay, the 2×2 and the prospective loop ran under it from the start. The three matched experiments and the original ten-instance repository panel predate the affected code path. Superseded runs and per-panel incidence counts are kept in the archived run logs.

Nulls and censoring. Null A fits child $= a + b \cdot$parent within each run and permutes fitted residuals while holding the realized parent sequence fixed (64 of 69 checkpoints inside its 90% band); Null B jointly simulates the parent trajectory, promotion gate, archive and curve (43 of 64, every miss above the band). Null A uses 2,000 fitted-residual permutations per run; two random-selection runs with fewer than ten promoted pairs are omitted. Null B combines a binomial ten-question

stage, an estimated family-specific promotion probability, the observed never-evaluated share, and the family's parent-selection rule. It reproduces 43 of 64 checkpoints; all 21 misses are observed curves above its band, and four runs with badly mispredicted promotion counts account for seven misses. The censoring scenarios treat the ten-question score as an unbiased full-score proxy. Their tipping point exceeds promoted-child accuracy by 0.02–0.25 in source runs and lies within 0.09 elsewhere; these values show the sensitivity but do not identify missing outcomes.

Parent productivity. For each parent with fully evaluated children we record the mean child score and the share of children that raised the run's best-so-far, avoiding any parent-minus-child quantity. Mean child score follows the parent's score (correlation 0.40 to 0.72 under score selection, 0.13 to 0.24 otherwise), which is parent–child persistence again; the share of best-raising children is 2% to 12% and uncorrelated or negatively correlated with parent score; and parents whose own edit touched a live improver file do not have more productive children once their own score is taken into account. Nothing indicates that the operator producing descendants became more productive over a run.

Meta^{n} audit. We ran the released system on `symptom2disease` for five seeds and 269 candidates. Per-run mean child-minus-parent changes range from -0.043 to -0.296. Where the parent range permits an estimate, the persistence slopes are -0.031 and -0.036, neither above its shuffled-parent null. Exact model versions, commits and per-run statistics are in the code release.

Before common-test evaluation, we selected 26 candidates: each initial and validation-selected chain, three deterministic iteration quantiles per seed, and their direct parents. Every candidate received the full 212-item official test and an independent repeat of its first 25 items. Three chains were non-executable on both evaluations and are reported separately. For the 23 executable chains, the repeat changes score by 0.014 in absolute value on average and disagrees on 0.035 of items. Averaged equally across seeds and candidates, validation exceeds common-test score by 0.072. The selected-chain validation/common-test means are $0.880/0.740$, a gap of 0.140.

Crossed operator replay. Before replay, we fixed the reconstruction to M_0 and the live meta-agent edits from generations 1, 3 and 12. All edits are children of M_0, and only M_1 has a child. Prompts use the same paths, filename substitutions, phase value and 40-call cap. Four operators, twelve parents and two replicates give 96 runs, with condition order randomized within block. Hidden means are $M_0 = 0.476$, $M_1 = 0.858$, $M_3 = 0.940$ and $M_{12} = 0.648$; family-clustered intervals for differences against M_0 are $[0.221, 0.500]$, $[0.407, 0.500]$ and $[0.062, 0.300]$, the parent bootstrap for $M_1 - M_0$ is $[0.251, 0.491]$, and $M_1 - M_3$ is -0.083 $[-0.187, 0.000]$. The replay was not designed to rank M_1 and M_3. The two parents with a negative M_1 effect belong to the family with two unstated conventions, on which the M_1 improver reads the report and then exhausts its reasoning budget without editing.

Operator replay on other banks. Both banks used the same two operators, randomized slots (seed 20092026), two replicates and their bank's original budget (interface: 40 calls, 4,096 tokens; repositories: 60 calls, 8,192 tokens). The only textual adaptation of M_1 on the repository bank replaces the priority bullet naming the task-agent file with "Source code of the library in the repository". Repository bank: M_0 0.500, M_1 0.800; seven instances positive, one negative and two tied (per-instance effects in the released logs); M_1 reached the 60-call cap in 6 of 20 runs and broke a regression test in 2; M_0 touched a gold-patch file in 1 of 20 runs, M_1 in 16. Interface bank: M_0 0.441, M_1 0.704; ten parents positive and two negative; leave-one-parent-out range $[0.237, 0.300]$; all 48 children compile; report content reached a tool response in 2 of 24 M_0 and 24 of 24 M_1 runs. The interface bank ran with the 24,000-character tool-output cap and the continuation nudges of the replay rather than the 6,000-character cap of the original interface experiment, identically for both operators.

Four-step rollouts. Each rollout reuses the replay's generation-one blocks for M_0 and M_1 on parents 1–6, two replicates each, and makes 72 new operator applications for generations two to four in randomized order. π_{dev} selects on the 40-case development score and retains the incumbent on ties; π_{replace} always takes the newest child. The endpoint is the terminal's 300-case hidden score. π_{dev}: contrasts 0.406, 0.484, 0.452; step-four terminal means 0.506 and 0.958; interval $[0.369, 0.500]$.

π_{replace}: 0.406, 0.365, 0.393; terminal means 0.506 and 0.899; interval $[0.190, 0.500]$. All 144 children compile.

Prospective operator loop. Eight new problem families with one unstated convention each, disjoint from the bank's eleven, were screened so that the initial solver fails 30 to 70 percent of development cases (initial development pass rate 0.494, hidden 0.500). Each repository holds a README describing the loop and an editable operator file whose instruction function and tool-call cap the harness executes. Each generation copies the selected parent, runs the improver under the operator's instruction, and evaluates the child on 160 development and 800 hidden cases; selection is the released score-proportional rule. Five seeds ran 80 generations each from the same initial repository; all 400 children compiled. Improvers edited the operator in 4, 5, 2, 8 and 2 generations of seeds 1 to 5; every seed reached a perfect development score, and mean child-minus-parent development change was negative in every seed (-0.014 to -0.047), as in the released histories. Selected versions were scored on the bank with their own instruction, the replay's budget, two replicates and randomized slots. Table 6 lists them.

Seed	Intro.	Sel.	Children	Child dev.	Edit	$Q^{(0)}$ on the bank
2	25	35	3	0.75	cap 30; adds "review SPEC.md and solution.py"	0.476 [0.431, 0.506]
3	11	13	40	0.99	cap 40; expert prompt that points to the evaluation-results directory and asks to read it	0.858 [0.681, 0.996]
3	6	16	18	0.88	cap 30; expert prompt on specification and edge cases, no evidence	0.575 [0.476, 0.685]
4	12	17	2	0.53	cap 30; text unchanged	0.547 [0.470, 0.649]
4	63	69	1	0.69	cap 50; text unchanged	0.538 [0.461, 0.619]
5	1	2	77	0.70	cap 25; "review and improve", specification focus	0.488 [0.428, 0.550]

Table 6: Operator versions selected as parents in the prospective loops. Introduced and selected are generations; children and their mean development score are loop-internal; $Q^{(0)}$ is the version's hidden score on the twelve-state bank with a parent-bootstrap interval. The initial operator scores 0.476 and the historical M_1 0.858 under the same harness. The harness fixes the improver's budget and ignores an operator's cap, so versions whose text is unchanged render the initial instruction and differ from 0.476 only by evaluation noise.

Information redundancy. We crossed twelve semantic-fault parents with blind and forced-read instructions at three specification levels. The low level leaves the evaluator's convention unstated, the medium level identifies the decision that requires a convention, and the high level states the convention. The 72 applications were assigned to a randomized order fixed in advance. Forced-minus-blind effects are $+0.372$ $[0.207, 0.522]$, $+0.046$ $[-0.179, 0.279]$ and -0.067 $[-0.200, 0]$ from low to high context; the paired low-minus-high interaction is $+0.439$ (sign test $p = 0.021$); all 72 children compile.

Same-instance SWE-bench 2×2. The 20 instances were screened for locally reproducible target and regression tests (six SymPy, five Django, four Sphinx, three pytest, one seaborn, one pylint). Each instance received all four arms once, arm order randomized within instance, with the repository panel's budget (60 calls, 8,192 tokens) and the same report, tools and evaluator. Arm means: blind 0.550 (2 of 20 resolved, 0 regression breaks), forced-read 0.803 (8, 1), issue-blind 0.824 (9, 2), issue-forced 0.837 (12, 3). Interaction $\Gamma = +0.241$, instance bootstrap $[0.038, 0.450]$, repository-clustered bootstrap $[-0.066, 0.487]$, leave-one-repository-out $[0.145, 0.343]$, sign test 12 positive, 4 negative, 4 tied ($p = 0.077$).

Nested meta-replay with target M_0. An earlier experiment held the target at M_0 on twelve meta-states (six semantic-fault solvers from the loop-suite families and six state-tracking parents, each with its report, a README describing the loop and the initial operator file), with two modifier replicates. Each distinct child operator was applied to twelve evaluation states (four semantic, four state-tracking, four repository) in two randomized replicates, averaged within each operator-state

pair. $\hat{\delta}_0 = +0.002$, two-way bootstrap $[-0.048, 0.045]$, with 40 of 48 children the unchanged M_0 and the randomization test giving $p = 0.97$. Child-operator qualities: unchanged operator 0.442; the eight edited operators 0.49 to 0.89 (mean gain 0.26), five from M_0 modifiers and three from M_1.

Modifier × target factorial. The protocol was fixed before any new run; amendments for budget, all made before any evaluation, restricted the design to the six semantic meta-states and the four semantic evaluation states. M_1 is one canonical operator file that renders exactly the instruction of every earlier M_1 treatment, and the placebo renders exactly M_0's instruction. Per cell (edits of 36 runs, mean quality of edited children, cell mean): M_0 on M_0 6, 0.57, 0.549; M_1 on M_0 4, 0.86, 0.580; M_0 on M_1 3, 0.80, 0.838; M_1 on M_1 0, –, 0.842; seed-3 edit on M_0 0, –, 0.545; M_0 on seed-3 edit 8, 0.82, 0.822; seed-3 edit on itself 0, –, 0.821; seed-2 and seed-5 edits on M_0 0 each; placebo on M_0 6, 0.60, 0.553. Intervals invert the exact sign-flip test over meta-states.

Resolution when edits are rare. An order-1 contrast moves by about the change in edit rate times the mean gain of an edited child (0.26 in the first experiment), and the edit indicator dominates the variance. We simulated this hurdle process with the observed edited operators, edit-rate heterogeneity across domains and the analysis's own estimator: with twelve meta-states, the sign-flip test detects an edit-rate rise from 0.17 to 0.50 with probability 0.51 at two modifier replicates and 0.93 at six, and is calibrated under identical arms. The variance decomposition makes this explicit. If a child equals the target, of quality Q_b, with probability $1 - p$ and is otherwise an edited operator of quality Q_e, then $\mathrm{Var}(Y) = p\,\mathrm{Var}(Q_e) + p(1-p)(\mathbb{E}Q_e - Q_b)^2$. With the eight edited operators of the first experiment (variance 0.019, mean gain 0.26) and $p = 8/48$, the second term, whether an edit occurs, is about three times the first, and rescoring a child reduces only evaluation noise, which neither term contains. Modifier runs cost about a cent and evaluation cost scales with the number of edits, so precision is cheapest to buy with modifier replicates. The six-replicate count was chosen for the registered design of twelve meta-states; budget amendments made before any evaluation reduced the factorial to six meta-states, for which the exact test can reject only when all six differences share a sign.

Positive control. Each of the seven semantic states' evidence includes a factual record of three attempts by the current operator, none of which opened the evaluation results or raised the development score. The constructed modifier asks the improver to diagnose the task solution and the improvement procedure separately and fix both, without naming the operator file; its wording was fixed after a separate feasibility pilot on other states, which enters no estimate. Two of its 35 edits failed to render and were scored as M_0.

Cross-model panel. Six improvers, twelve parents, four arms, one replicate, 288 runs. Forced-minus-blind nine-family intervals are MiMo $+0.339$ $[0.154, 0.492]$, GLM $+0.193$ $[-0.056, 0.408]$, Kimi $+0.167$ $[0.036, 0.308]$, Grok $+0.293$ $[0.044, 0.490]$, DeepSeek 0.000 (blind arm at 1.000) and Gemini -0.109 $[-0.292, 0.110]$. Randomization tests permute condition labels within parent; Holm-adjusted values within each model's three prespecified contrasts are < 0.001 (MiMo), 0.100 (GLM), 0.135 (Kimi), 0.015 (Grok), 1 (DeepSeek) and 0.443 (Gemini) for the forced contrast, and < 0.001, 0.100, < 0.001, 0.057, 1 and 0.615 for inline delivery. Blind-arm report receipt is 0, 2, 0, 0, 12 and 10 of 12. A seventh model, Qwen, could not be run because its provider rate-limited every attempt. Kimi's requests are routed to several providers, one of which returned empty replies and served nearly all blind-arm calls but a varying share of forced-arm calls, so Kimi's contrast is not interpretable.

Meta^n common-test result. The order null covers 19 of 20 checkpoints. The common-test follow-up gives an initial-to-selected mean of 0.309 $[0.008, 0.623]$ across all five seeds and 0.014 $[0, 0.047]$ for the three executable initial programs. The 15 matched parent–child contrasts average -0.079 $[-0.161, -0.005]$; the 14 executable pairs average -0.031 $[-0.092, 0.029]$. Per-chain results are in the released logs.

Intervals use 20,000 family bootstraps over the nine families (three families contribute two parents each). A transcript audit found no run in any reported panel that read a file outside its working repository. Correlations of gain with non-reading and headroom are exploratory. Per-arm means and raw records are in the released logs.

Stress tests. The persistence slope, excess agreement, promotion rates and the collapse mechanism are stable under leave-one-run-out and a run-level bootstrap; the direction-of-change share is not separated from one half in the transfer and continuation arms and is reported as a range.

Configuration	Runs	Full validation size	Runs with a balanced full evaluation
Hyperagents source	5	100	5
Without open-ended selection	5	10	0
Fixed improver, paper review	5	10	0
DGM baseline, paper review	5	10	0
DGM customized	5	100	5
Improver edits parent selection	5	100	5
Random parent selection	5	100	4
Transfer control, original improver	5	10 or 100	0
Transfer, transferred improver	5	100	5
DGM transfer control	5	10 or 100	0
DGM transfer, transferred	5	100	2
200-generation continuation, source	5	100	5
200-generation continuation, transferred	5	100	5
Strong-baseline initialization	1	100	1
Proof writing (no labels)	5	–	–

Table 7: Hyperagents configurations covered. Five configurations never left the ten-instance staged evaluation.

Endpoint state	Pairs	Improved	Tied	Regressed	Mean Δ
Balanced $\rightarrow$ balanced	147	56	8	83	-0.014
Balanced $\rightarrow$ collapsed	56	0	1	55	-0.124
Collapsed $\rightarrow$ balanced	57	52	2	3	$+0.101$
Collapsed $\rightarrow$ collapsed	99	23	33	43	-0.014

Table 8: Matched validation pairs in the five Hyperagents source runs, grouped by endpoint label-balance state.

Family	Pairs	Imp. / tied / reg.	Mean Δ	r(parent, child)	Retest SD
Hyperagents source	147	56 / 8 / 83	-0.014	0.52	0.025
DGM customized	69	27 / 6 / 36	-0.009	0.37	0.028
Improver edits selection	133	51 / 14 / 68	-0.007	0.77	0.021
Random parent selection	19	7 / 0 / 12	-0.016	0.25	0.005
Transfer, transferred	144	60 / 16 / 68	-0.003	0.35	0.019
Continuation, source	569	234 / 56 / 279	-0.008	0.57	0.037
Continuation, transferred	736	331 / 72 / 333	-0.007	0.58	0.038
Strong-baseline init.	196	65 / 19 / 112	-0.014	0.05	–
DGM SWE-bench	40	18 / 3 / 19	$+0.010$	0.69	–
DGM SWE-bench, no archive	25	10 / 3 / 12	$+0.001$	0.51	–
DGM SWE-bench, no self-imp.	25	13 / 2 / 10	-0.017	0.26	–
DGM Polyglot	50	11 / 18 / 21	-0.039	0.11	–
DGM Polyglot, no archive	20	8 / 2 / 10	-0.039	0.08	–
DGM Polyglot, no self-imp.	19	7 / 5 / 7	-0.011	-0.29	–

Table 9: Direction of change and raw parent–child correlation. Balanced pairs for Hyperagents; common-task matched pairs for DGM. Retest SD is the standard deviation of the score change for children whose diff touched no solver or shared code file.

C FURTHER ORDER-0 RESULTS

Four-step productivity under two downstream protocols. $Q_H^{(0)}$ depends on the downstream policy π, so we extend the replay on the first six families through four steps under two policies, chaining from the replay's generation-one children. Under π_{dev}, a child replaces the incumbent only if its 40-case development

score is strictly higher; hidden scores never affect the archive. The $M_1 - M_0$ contrasts are 0.406, 0.484 and 0.452 at steps one, two and four. At step four, terminal hidden means are 0.506 for M_0 and 0.958 for M_1; the parent-bootstrap interval is $[0.369, 0.500]$, the leave-one-parent-out range is $[0.443, 0.493]$, and all six family effects are positive. Under π_{replace}, chosen as the policy most likely to break the advantage, the newest child always becomes the incumbent and no selection acts, so regressions propagate. The contrasts are 0.406, 0.365 and 0.393; at step four, terminal hidden means are 0.506 for M_0 and 0.899 for M_1 (interval $[0.190, 0.500]$; five families positive, one negative where a late regression propagated). Repeated application of M_1 keeps or reaches the ceiling on most families, including the two-convention family on which one application fails, while M_0 chains barely move. The descendant advantage on this bank therefore does not depend on the selection rule, though without selection it is exposed to individual regressions.

Prospective operator versions. The released history contains one selected operator edit, so progressive operator improvement, $Q^{(0)}(M_t; \nu)$ rising across selected versions, cannot be observed in it. We therefore ran a self-referential loop that produces operator versions prospectively (Appendix B). Each repository holds a solver for an eight-function suite of new semantic-fault families, disjoint from the evaluation bank, and an operator file that the improver may edit and that the harness executes to build the next improver's instruction. The initial operator is M_0. Parents are chosen by the released score-proportional rule on development score; hidden scores are recorded and never used. Five seeds ran for 80 generations. A version is selected when it produced at least one child, and every selected version is measured on the twelve-state bank under the replay's common budget, two replicates per state, so that $Q^{(0)}(M_t; \nu_{\text{eval}})$ is directly comparable to the replay (Figure 3). Improvers edited the operator in 21 generations and six versions were selected (Table 6). One, selected at generation 13 of seed 3, tells the improver where the evaluation results are and to read them; it scores 0.858 $[0.681, 0.996]$ on the bank, the level of the historical M_1; its improvers opened the report in 39 of 40 generations and all 22 of its children scored perfectly on development. Every other selected version, each rewording the instruction around the specification or only raising the tool budget, scores between 0.476 and 0.575, with intervals that cover M_0's 0.476. Selection did not distinguish them: once the suite was saturated, a specification-only version introduced earlier in the same seed, at generation 6, was selected at generation 16 and scores 0.575. The archives themselves reached a perfect development score in every seed, between generations 2 and 37, while the operator was still M_0 in four of them: the M_0 improver received the report in 22 of 259 generations, those children scored far above their parents, and selection retained them. The archive curve rose through state improvement and selection with the child-producing operator unchanged, which is the first term of the decomposition standing still while the second moves. The loop reproduces the released pattern rather than progressive improvement: the one operator edit that raises held-out operator quality is evidence routing, and selection on child score found it in only one of five seeds.

D PROOFS

Setting. $\mathcal{O}$ is a measurable space of operators; for each $m \in \mathcal{O}$, K_m is a Markov kernel from meta-states to $\mathcal{O}$ (the law of the child operator), and $Q^{(0)} : \mathcal{O} \to [0, 1]$ is measurable. With ν_1 fixed, $Tf(m) = \mathbb{E}_{Z \sim \nu_1} \int f(o) K_m(Z, do)$ is a positive linear operator on bounded measurable f, and $Q^{(r)} = T^r Q^{(0)}$.

Theorem 1 and Corollary 1. Under $K_{M_a}(Z^b, \cdot)$ the child operator equals M_b with probability $1 - p_{ab}$, and then its quality is $Q^{(0)}(M_b)$; otherwise its quality has mean μ_{ab}. Averaging over $Z \sim \nu_1$ gives $Y_{ab} = (1 - p_{ab})Q^{(0)}(M_b) + p_{ab}\mu_{ab}$, which rearranges to the stated form; subtracting the $(0, 0)$ cell from the $(1, 1)$ cell gives the decomposition, and $\delta_b = p_{1b}[\mu_{1b} - Q^{(0)}(M_b)] - p_{0b}[\mu_{0b} - Q^{(0)}(M_b)]$ vanishes when $p_{1b} = p_{0b} = 0$. Since μ_{ab} and $Q^{(0)}(M_b)$ both lie in $[0, 1]$, each bracket has absolute value at most 1, which gives the corollary. □

Theorem 3. Fix $r \geq 1$ and take $\mathcal{O}$ countable with elements $A, B, a_1, \ldots, a_r$ and $b_1, \ldots, b_r$. Let every kernel be deterministic and independent of the meta-state: $K_A(\cdot) = \delta_{a_1}$, $K_{a_j} = \delta_{a_{j+1}}$ for $j < r$, $K_{a_r} = \delta_{a_r}$, and likewise for B and the b_j. Set $Q^{(0)}(A) = Q^{(0)}(B) = c_0$ and $Q^{(0)}(a_j) = Q^{(0)}(b_j) = c_j$ for $j < r$, and $Q^{(0)}(a_r) = 1$, $Q^{(0)}(b_r) = 0$. Induction on j gives $Q^{(j)}(A) = Q^{(0)}(a_j) = c_j = Q^{(j)}(B)$ for $1 \leq j < r$ and $Q^{(0)}(A) = Q^{(0)}(B)$, while $Q^{(r)}(A) = 1 \neq 0 = Q^{(r)}(B)$. For the observational claim, let task scores be Bernoulli with mean $Q^{(0)}$ of the operator in use and let the archive process select parents and promote children by any rule that depends on the history only through observed scores (the score-measurability assumption in the statement; a rule that inspected operator code could of course distinguish a_t from b_t). Under A the operator in use at generation $t \leq r - 1$ is a_t and under B it is b_t, with identical score laws $\text{Bernoulli}(c_t)$, so the joint law of all scores, promotions and selections through generation $r - 1$ coincides; the laws first differ at generation r. Any statistic measurable with respect to observables through horizon $H < r$ therefore has the same distribution under A and B, and cannot identify the sign of $Q^{(r)}(A) - Q^{(r)}(B)$. Replacing a_r by an operator of quality 0 and b_r by one of quality 1 reverses the sign with the same lower-order agreement. □

Theorem 2. Condition on the fixed banks. For each j, m, r, the child $\tilde{o}_{jmr}$ has law $K_m(Z_j, \cdot)$, and for each k the evaluation randomness ξ_k is independent of $(\tilde{o}_{jmr}, Z_j)$, so $\mathbb{E}[S(\tilde{o}_{jmr}(X_k, \xi_k)) \mid \tilde{o}_{jmr}] = \mathbb{E}_\xi S(\tilde{o}_{jmr}(X_k, \xi))$ and, averaging over the K fixed states, $\mathbb{E}[\hat{q}(\tilde{o}_{jmr}) \mid Z_j] = \int Q^{(0)}(o; \hat{\nu}_0) K_m(Z_j, do)$ with $\hat{\nu}_0$ the empirical evaluation bank. Averaging over r and taking the paired difference within j gives $\mathbb{E}[\hat{\Delta}_1 \mid Z_{1:J}] = J^{-1} \sum_j \left[\int Q^{(0)} K_{m_1}(Z_j, do) - \int Q^{(0)} K_{m_0}(Z_j, do) \right] = Q^{(1)}(m_1; \hat{\nu}_1, \hat{\nu}_0) - Q^{(1)}(m_0; \hat{\nu}_1, \hat{\nu}_0)$; if the X_k are drawn from ν_0 and the Z_j from ν_1, a further expectation gives the population contrast. Randomizing modifier labels among execution slots within a meta-state makes the two modifiers exchangeable with respect to slot effects and licenses a randomization test under the sharp null that enumerates that scheme. Because the K states are shared by every child and the R replicates share Z_j, the $\hat{q}$ values within a meta-state are dependent; the meta-state is the exchangeable unit. □

Proposition 1 (Order reversal)**.** *There exist self-edits $m \to m'$ with $\Delta_0 > 0$ and $\Delta_1 < 0$, and self-edits with $\Delta_0 < 0$ and $\Delta_1 > 0$.*

Proposition 1. Take $\mathcal{O} = \{m, m', c, c'\}$ with $Q^{(0)}(m) = 0.4$, $Q^{(0)}(m') = 0.6$, $Q^{(0)}(c) = 0.9$, $Q^{(0)}(c') = 0.1$, and meta-state-independent kernels $K_m = \delta_c$, $K_{m'} = \delta_{c'}$. Then $\Delta_0 = 0.2$ and $\Delta_1 = Q^{(0)}(c') - Q^{(0)}(c) = -0.8$. Exchanging the roles of c and c' gives $\Delta_0 = 0.2$ and $\Delta_1 = 0.8$; exchanging m and m' instead gives $\Delta_0 = -0.2$ with $\Delta_1 = 0.8$. □

Earlier propositions. The following results are used in Sections 4 and 5: 1A/1B (historical non-identifiability: an archive trajectory does not identify Δ_0, which is logically distinct from Theorem 3), 2 (common-parent identification, the order-0 case of Theorem 2), 3 (stationarity check), 4 (censoring), 5 (horizon non-implication), 6 (distribution relativity) and 7 (non-equivalence to descendant productivity, used in Section 8).

Propositions 1A and 1B. Fix a process with initial agent C_0, stationary operator M, selection rule π and randomness $\xi_1, \ldots, \xi_T$, giving parents $P_t = \pi(\mathcal{A}_{t-1})$ and children $C_t = M(P_t, \xi_t)$. Define a second process with every parent equal to C_0 and $M'_t(C_0, \xi_t) = M(\pi(\mathcal{A}_{t-1}), \xi_t)$, where $\mathcal{A}_{t-1}$ is the archive the first process would have built from the past randomness. This is a legitimate history-dependent operator with no selection acting on it. By construction $S(C'_t) = S(C_t)$ for every t and realization, proving 1A without claiming that M'_t improves. For 1B, additionally suppose $S(P_t)$ is nondecreasing almost surely and $h(s) = \mathbb{E}[S(M(P, \xi)) \mid S(P) = s]$ is nondecreasing. Then $\mathbb{E}[S(M'_t(C_0, \xi)) \mid \text{past}] = h(S(P_t))$ is nondecreasing. Score-based selection alone does not supply the first condition. □

Proposition 2. For fixed states X_i sampled from ν, independent applications give $\mathbb{E}[Y_{ijz} \mid X_i] = \mathbb{E}[S(M_z(X_i, \xi)) \mid X_i]$. The within-state contrast is unbiased for the conditional effect, and its mean is unbiased for the empirical-reference contrast (and for the population contrast when the states are sampled from ν). If operator labels are randomized among experimental units within state, inference may enumerate that actual assignment scheme under the sharp null. Merely randomizing execution order while applying every operator does not make an arbitrary sign-flip test design-exact. □

Proposition 3. With known f, conditioning on the parent sequence fixes $f(S(P_t), \text{run})$, and exchangeability makes every error permutation equiprobable, so the permutation distribution of any statistic is exact. Estimating f from the same outcomes generally destroys this finite-sample symmetry; finite-sample exactness then requires a scheme justified for the estimator, such as appropriate refitting or sample splitting. Time-varying evidence, heteroskedasticity, changing task mix, models or tools also void exchangeability. □

Proposition 4. By iterated expectation and conditional independence, $\mathbb{E}[Y \mid X] = \mathbb{E}[Y \mid X, D{=}1]$; positivity is also needed so this conditional mean is estimable throughout the target support. Without these assumptions, $\mathbb{E}[Y] = \Pr(D{=}1)\mathbb{E}[Y \mid D{=}1] + \Pr(D{=}0)\mathbb{E}[Y \mid D{=}0]$, and the second mean lies only within the outcome range. Substantive imputations narrow this range by assumption and must be labeled sensitivities. □

Proposition 5. Consider a point-mass initial distribution. Let M_a produce state a with score 1 and M_b produce state b with score 0, so $Q^{(0)}(M_a) > Q^{(0)}(M_b)$. Let the fixed downstream protocol map a to a terminal state with score 0 and b to one with score 1. Then $Q_2^{(0)}(M_a) < Q_2^{(0)}(M_b)$. The construction extends to any $H > 1$ by inserting score-preserving intermediate states. □

Proposition 6. Define $\tau(x) = \mathbb{E}_\xi[S(M_1(x, \xi)) - S(M_0(x, \xi))]$. Linearity of integration gives $\Delta(\nu) = \int \tau(x) d\nu(x)$ and therefore $\Delta(\nu') - \Delta(\nu) = \int \tau(x)[d\nu'(x) - d\nu(x)]$. If τ has both signs, distributions concentrated on the corresponding regions reverse the operator ranking. □

Proposition 7 (descendant productivity). Let a search process start from operator m, present the operator in use with the meta-states its own lineage induces, apply each child operator it produces, and select by any rule that depends on the history only through observed task scores; let $\mathrm{CMP}_\pi(m)$ be the expectation, under policy π and horizon $H \geq 2$, of a functional of the task scores in m's clade, such as their mean or maximum (Wang et al., 2026b). Then (i) there exist operators A, B with $Q^{(0)}(A) = Q^{(0)}(B)$ and $Q^{(1)}(A) = Q^{(1)}(B)$ but $\mathrm{CMP}_\pi(A) \neq \mathrm{CMP}_\pi(B)$; and (ii) there exist A, B with $\mathrm{CMP}_\pi(A) = \mathrm{CMP}_\pi(B)$ for every such π and H but $Q^{(1)}(A) \neq Q^{(1)}(B)$. *Proof.* Take two meta-states: z_1, the declared bank ($\nu_1 = \delta_{z_1}$), and z_2, the state the lineage presents. Let task scores be Bernoulli with mean the $Q^{(0)}$ of the operator in use, set $Q^{(0)}(A) = Q^{(0)}(B) = q \in (0, 1)$, and let the auxiliary operators a, b, c never edit themselves. For (i), let $K_A(z_1, \cdot) = K_B(z_1, \cdot) = \delta_c$, so $Q^{(1)}(A) = Q^{(1)}(B) = Q^{(0)}(c)$, and $K_A(z_2, \cdot) = \delta_a$, $K_B(z_2, \cdot) = \delta_b$ with $Q^{(0)}(a) = 1$, $Q^{(0)}(b) = 0$; from the second generation on, the two clades' scores are Bernoulli(1) and Bernoulli(0), so their mean and maximum differ in expectation. For (ii), let $K_A(z_2, \cdot) = \delta_A$ and $K_B(z_2, \cdot) = \delta_B$, so both clades produce Bernoulli(q) scores at every generation and every score-measurable functional has the same law, while $K_A(z_1, \cdot) = \delta_A$ and $K_B(z_1, \cdot) = \delta_b$ give $Q^{(1)}(A) = q \neq 0 = Q^{(1)}(B)$. The two quantities can agree only when the declared meta-state distribution is the one the search induces and the metric weights children as the policy does. □