EVIDRA: EVIDENCE-GUIDED INTERVENTION SELECTION FOR RESEARCH AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Scientific discovery requires research agents to decide how to revise their process in response to experimental evidence. We present EVIDRA, a framework for evidence-guided intervention across prompts, workflows, context, memory, tools, and sampling. Each intervention links a testable hypothesis to an edit target, a recoverable baseline, and expected observations. The controller prioritizes interventions by expected improvement, information value, cost, and risk, then combines scoped execution, rollback checks, and independent assessment to build reusable research evidence. On four graph-optimization tasks, EVIDRA achieves an equal-task mean gap-closure score of 0.8736, compared with 0.8682 for the strongest baseline, and leads on Cluster Editing and Treedepth. Three mathematical studies yield an exact counterexample to Lonely Runner translation monotonicity, an improved Erdős–Straus sieve-weight certificate, and a tighter certified upper bound for a four-variable iid probability problem.

1 INTRODUCTION

AI systems are increasingly able to produce concrete research artifacts. AlphaTensor and AlphaDev discover matrix-multiplication and sorting algorithms (Fawzi et al., 2022; Mankowitz et al., 2023); FunSearch and AlphaEvolve combine language-model proposals with executable evaluation for mathematical and algorithmic discovery (Romera-Paredes et al., 2024; Novikov et al., 2025). The AI Scientist extends automation across idea generation, experimentation, analysis, and manuscript preparation (Lu et al., 2024; Yamada et al., 2025; Lu et al., 2026). These developments make a consequential question increasingly practical: *after observing an experimental outcome, how should a research agent decide what to change next, and what evidence would justify that change?*

This question concerns the research process as well as its output. A failed search may reflect an unproductive hypothesis, an unsuitable exploration strategy, missing context, or an uninformative tool interface. These explanations imply different interventions even at the same score. An agent must connect observations to proposed changes, test them under controlled conditions, and retain the resulting evidence.

Consider a search that repeatedly produces invalid constructions. One hypothesis is that the proposal omits a feasibility condition; another is that exploration remains confined to an unproductive region. The first suggests changing the instructions or evidence context, whereas the second suggests changing the workflow or sampling. A useful next experiment should distinguish these explanations. Its outcome can then inform both the current candidate and future decisions about where to intervene. This motivates representing the explanation, proposed change, and expected observation together.

Existing methods optimize program populations, language-model programs, and agent workflows (Cemri et al., 2026; Agrawal et al., 2026; Zhang et al., 2026a). Experience-based agents reuse reflections, while recent frameworks organize scoped changes and evidence chains (Shinn et al., 2023; Zhang et al., 2026b; XYZ Agentic Team, 2026; Meng et al., 2026). These capabilities broaden what an agent can change. The central design question is how to choose an intervention that tests an explanation of the available evidence, then use the outcome to guide further research. This requires connecting intervention selection, execution, and assessment within one experimental loop, while assigning distinct roles to development and assessment feedback (Dwork et al., 2015).

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.

We introduce **EVIDRA**, a framework for evidence-guided intervention selection across prompts, workflows, context, memory, tools, and sampling (Figure 1). Each proposal binds a testable hypothesis to an explicit edit target, a recoverable baseline, and a declared scope. The controller ranks eligible proposals using expected improvement and diagnostic value relative to cost and risk. Paired rollback checks an intervention against its baseline, and independent assessment evaluates the resulting candidate. Development observations guide subsequent search; both successful and unsuccessful trials are retained with their conditions and verification status for reuse.

We evaluate EVIDRA on four graph-optimization tasks and three mathematical case studies. With shared task interfaces and request limits, the four-task mean score is 0.8736, compared with 0.8682 for the strongest baseline, with the highest means on Cluster Editing and Treedepth. Component experiments compare the full system with w/o Process Adaptation and w/o Propose. The mathematical cases yield a Lonely Runner translation-monotonicity counterexample, an improved Erdős–Straus sieve-weight certificate, and a tighter certified probability bound.

Our contributions are threefold:

- A framework for hypothesis-bound interventions with explicit targets, recoverable baselines, and evidence-guided proposal ranking.
- An evaluation on four graph-optimization tasks against four baselines, with the highest aggregate score and two component ablations.
- Three mathematical results with explicit witnesses or certificates, documented search procedures, and exact verification.

2 Related Work

Program and agent optimization. Executable evaluation drives program discovery in FunSearch and AlphaEvolve (Romera-Paredes et al., 2024; Novikov et al., 2025), while OpenEvolve, AdaEvolve, and ShinkaEvolve develop population and parent selection strategies (Sharma, 2025; Cemri et al., 2026; Lange et al., 2026). DSPy, TextGrad, and GEPA optimize language-model programs (Khattab et al., 2024; Yuksekgonul et al., 2025; Agrawal et al., 2026); ADAS, AFlow, and the Darwin Gödel Machine optimize agents and workflows (Hu et al., 2025; Zhang et al., 2025; 2026a). EVIDRA focuses on the experiment surrounding an edit: its hypothesis, scope, baseline, and subsequent attribution and assessment.

Scientific experimentation and evidence. The AI Scientist connects experiments with manuscript preparation; its second version introduces staged tree search and repeated experiments (Lu et al., 2024; Yamada et al., 2025; Lu et al., 2026). Reflexion, ExpeL, and ACE reuse reflections and lessons (Shinn et al., 2023; Zhao et al., 2024; Zhang et al., 2026b); AI4AI and ScientistOne organize scoped improvement and evidence chains (XYZ Agentic Team, 2026; Meng et al., 2026). EVIDRA links this experience to executed changes and verification status. Counterexample-guided synthesis provides a precedent for iterative construction and checking (Solar-Lezama et al., 2006); empirical assessment additionally requires controlling adaptive feedback (Dwork et al., 2015).

3 Problem Formulation

A research contract specifies the task, allowed changes, evaluation rules, and execution limits. Let a denote the output being developed, such as solver code or a mathematical certificate, and x the research-process configuration. Its six intervention families are

$$\mathcal{O} = \{\mathrm{Prompt}, \mathrm{Workflow}, \mathrm{Context}, \mathrm{Memory}, \mathrm{Tools}, \mathrm{Sampling}\}. \tag{1}$$

Appendix A.1 defines their edit boundaries. An experiment targets either a component of x indexed by $\mathcal{O}$ or the output a, denoted by the additional target $\mathrm{Artifact}$.

At step t, state S_t holds candidate outputs, hypotheses, experimental evidence, retrieved experience, and statistics for actions and targets. A proposal q names a target $o(q) \in \mathcal{O} \cup \{\mathrm{Artifact}\}$, baseline $b(q)$, scoped change $\Delta(q)$, and testable hypothesis. The controller chooses

$$q_t = \operatorname*{arg\,max}_{q \in \mathcal{Q}(S_t)} U(q \mid S_t), \qquad o_t = o(q_t), \tag{2}$$

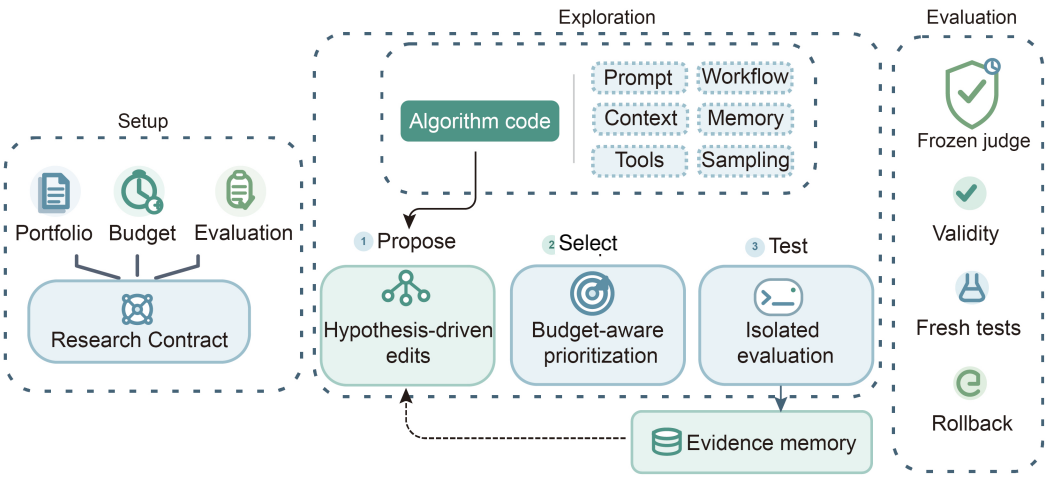

Figure 1: **EVIDRA.** The agent proposes scoped experiments, prioritizes them using available evidence, and evaluates the resulting changes. Development results and rollback checks inform future search; independent assessment determines acceptance without returning feedback.

where $\mathcal{Q}(S_t)$ contains proposals allowed by the task. For $o_t \in \mathcal{O}$, the intervention changes its declared process component: $x'^{(j)} = x^{(j)}$ for $j \neq o_t$. An Artifact intervention edits a.

4 EVIDRA

4.1 Proposal and Evidence-Guided Selection

The agent uses the current output, hypotheses, development feedback, and retrieved experience to propose experiments and predict their outcomes. The controller ranks eligible interventions by

$$U(q \mid S_t) = \frac{\lambda_g \widehat{g}_q \widetilde{p}_q + \lambda_I \widehat{I}_q - \lambda_r \max(\widehat{r}_q, r_{o(q)}) + \kappa/\sqrt{N_{o(q)}+1}}{\max(\widehat{c}_q, \varepsilon)} - \lambda_{\text{rep}} D_t(q). \tag{3}$$

The proposal estimates improvement $\widehat{g}_q$, information value $\widehat{I}_q$, risk $\widehat{r}_q$, and cost $\widehat{c}_q$. Here $r_{o(q)}$ is target-specific risk, and $N_{o(q)}$ counts completed attempts. The success estimate $\widetilde{p}_q$ combines the proposed likelihood with observed validity and improvement rates. $D_t(q)$ indicates a previously attempted proposal; within-batch duplicates are removed before ranking. The λ coefficients weight these terms; κ encourages trying less-explored targets, and $\varepsilon > 0$ stabilizes division by cost. This heuristic balances expected improvement with diagnostic value. Appendix A.1 gives the updates and parameter values.

4.2 Execution, Attribution, and Assessment

The controller checks that the actual change matches its declared scope, then evaluates the candidate under fixed rules. Development results guide search. To check attribution, a paired rollback restores the exact baseline and reevaluates it using the same seeds; the candidate's advantage must disappear under the task's comparison rule. This tests the specified edit as a whole. Trials retain their hypotheses, conditions, outcomes, and verification status for later retrieval. Independent assessment evaluates a fixed candidate on separate instances; its diagnostics are withheld from subsequent search.

4.3 Research Workflow and Task Execution

EVIDRA applies interventions to the research components in Equation 1. Prompt and context interventions revise instructions and evidence; workflow and sampling interventions adjust experiment generation and execution; memory and tool interventions change retrieval and editable interfaces.

For graph optimization, the controller assembles task-specific prompts, evidence, and retrieved experiment records. It uses these to repair candidates, extend promising branches, explore alternative parents, or restart search. The outputs are executable algorithms; mathematical tasks produce witnesses and certificates. Appendix A.1 gives the execution procedures and experimental settings.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Tasks and data. We use TaskCapsule to generate instances of one-sided crossing minimization (OCM), directed feedback vertex set (DFVS), Cluster Editing, and Treedepth. Their objectives minimize edge crossings, deleted vertex weight, edge edits, and elimination-forest depth, respectively. Three data groups use different generation seeds. Each contains small and medium development instances and independently generated evaluation instances. This separates the feedback used to guide research from the instances used to measure the selected algorithm's performance. Appendix G specifies instance counts, graph families, sizes, and seeds.

Shared protocol and baselines. All methods share the same DeepSeek model, task instances, initial and reference algorithms, evaluation protocol, and API-call budget. We compare EVIDRA with two evolutionary program-search methods, AdaEvolve (Cemri et al., 2026) and OpenEvolve (Sharma, 2025), and two automated research systems, AI Scientist V1 (Lu et al., 2024) and V2 (Yamada et al., 2025). These baselines provide comparisons with population-based algorithm improvement and agent-directed experimentation. Baseline adaptations, model configuration, and resource settings appear in Appendix G.

Scoring. We measure improvement using gap closure. For initial, reference, and candidate objectives s, r, and c with $s > r$, the score is $g(s, r, c) = (s - c)/(s - r)$. Zero matches the initial algorithm, one matches the reference, and scores above one improve on it. Each task reports the mean and sample standard deviation across the three data groups; the overall score equally weights the four tasks. This normalization compares improvement across objectives with different scales. Appendix E gives the complete scoring rules.

5.2 BENCHMARK RESULTS

Table 1 compares five methods on four TaskCapsule tasks using evaluation scores. Appendix E details the scoring and aggregation.

Table 1: Evaluation gap-closure scores (↑). Task entries are mean ± sample standard deviation; Avg. equally weights the four task means. The highest mean in each column is bold. Aggregation details are given in Appendix E.

Method	OCM	DFVS	Cluster Editing	Treedepth	Avg.
AdaEvolve	0.8528 ± 0.2549	0.4460 ± 0.2252	0.9192 ± 0.1143	0.9517 ± 0.0571	0.7924
OpenEvolve	$\mathbf{1.0000} \pm 0.0000$	$\mathbf{0.5147} \pm 0.2602$	0.9234 ± 0.1232	1.0348 ± 0.0158	0.8682
AI Scientist V1	0.9746 ± 0.0250	0.4486 ± 0.2677	0.8843 ± 0.1746	0.6527 ± 0.5668	0.7401
AI Scientist V2	0.9980 ± 0.0035	0.3803 ± 0.3194	0.9218 ± 0.1168	1.0405 ± 0.0329	0.8351
EVIDRA (ours)	0.9978 ± 0.0019	0.4630 ± 0.2581	$\mathbf{0.9823} \pm 0.0159$	$\mathbf{1.0515} \pm 0.0187$	**0.8736**

EVIDRA achieves the highest equal-task mean, 0.8736, compared with 0.8682 for OpenEvolve, and leads on Cluster Editing and Treedepth. Cluster Editing provides the clearest task-level advantage: EVIDRA scores 0.9823 against 0.9234 for the next-best method, a gap of approximately 0.0589. Its sample standard deviation of 0.0159 is also the smallest in this column, pairing a stronger mean with less variation across the three data groups. On Treedepth, EVIDRA exceeds the strongest competing mean, 1.0405 from AI Scientist V2, by approximately 0.0110.

OCM shows a different pattern: several methods approach the reference level, and EVIDRA's 0.9978 is within 0.0022 of the leading mean. OpenEvolve leads on OCM and DFVS, with EVIDRA

ranking second on the latter. These task-wise comparisons locate EVIDRA's aggregate advantage in improvements across two distinct optimization objectives, alongside competitive performance on the remaining tasks.

5.3 Component Ablations

We compare full EVIDRA with two variants, *w/o Process Adaptation* and *w/o Propose*, under the same experimental protocol. Table 2 reports their evaluation scores. Appendix F gives the variant settings and an additional comparison with a configuration-matched control on three tasks.

Table 2: EVIDRA component ablations: evaluation gap-closure scores (↑), mean ± sample SD. Avg. equally weights tasks. The benchmark reference reproduces Table 1; see Appendix E for aggregation.

Policy	OCM	DFVS	Cluster Editing	Treedepth	Avg.
EVIDRA (Table 1)	0.9978 ± 0.0019	0.4630 ± 0.2581	0.9823 ± 0.0159	1.0515 ± 0.0187	0.8736
w/o Process Adaptation	1.0000 ± 0.0000	0.3297 ± 0.2331	0.9313 ± 0.1191	1.0515 ± 0.0326	0.8281
w/o Propose	0.9114 ± 0.1234	0.2947 ± 0.3048	0.7094 ± 0.5033	0.8912 ± 0.0962	0.7017

w/o Propose. This variant's four-task mean is 0.7017 versus 0.8736 for EVIDRA, a 19.68% reduction in normalized score. All task means are lower, especially on Cluster Editing. The EVIDRA control with matching settings also exceeds this ablation in all three data groups (Appendix F).

w/o Process Adaptation. This variant's four-task mean is 0.8281, 5.21% below the benchmark reference, mainly through DFVS and Cluster Editing. OCM improves slightly and Treedepth is unchanged. On the configuration-matched three-task subset, w/o Process Adaptation averages 0.9942 and the EVIDRA control averages 0.9784; this subset excludes DFVS (Appendix F). Appendix H analyzes search trajectories, failed proposals, and an improvement found from an alternative parent.

6 Case Studies in Mathematical Discovery

We study three search objects: local speed neighborhoods, finite-prime certificate parameters, and rational packing weights. Each case combines hypothesis-driven search with exact verification to produce a checkable mathematical result.

6.1 Lonely Runner: A Counterexample to Translation Monotonicity

Can increasing every moving runner's speed reduce how far a stationary runner can become separated from the others? For runners starting together on a unit circle with positive integer speeds V, this separation is

$$d_V(t) = \min_{v \in V} \|tv\|, \qquad \kappa(V) = \max_{t \in [0,1/2]} d_V(t), \tag{4}$$

where $\|x\|$ is the distance from x to the nearest integer. The Lonely Runner Conjecture predicts $\kappa(V) \geq 1/(|V|+1)$. Perarnau & Serra (2025, Problem 4) ask a related question: does $\kappa(V+a) \geq \kappa(V)$ always hold for a positive integer a, where $V + a = \{v + a : v \in V\}$? Here the reference runner remains stationary; only the other speeds are shifted.

We search for counterexamples by maximizing the drop $\Delta(V) = \kappa(V) - \kappa(V+1)$ over triples of distinct speeds in $[1, 30]$, requiring both $\gcd(V) = 1$ and $\gcd(V+1) = 1$. The agent proposes local neighborhoods to explore, inspects the returned scores, and revises its next proposal. An exact-arithmetic tool expands each neighborhood and computes both separation maxima. Appendix B gives the search and verification details.

The search finds successively stronger counterexamples in 25 model calls (Figure 2B). Its last two accepted triples are $(1, 18, 20)$ and $(1, 20, 22)$, increasing the drop from $113/760$ to $82/525$. For the final triple $V = (1, 20, 22)$ and its shift $V + 1 = (2, 21, 23)$, exact evaluation gives

$$\kappa(V) = \frac{10}{21} \quad \text{at} \quad t = \frac{10}{21}, \qquad \kappa(V+1) = \frac{8}{25} \quad \text{at} \quad t = \frac{4}{25}. \tag{5}$$

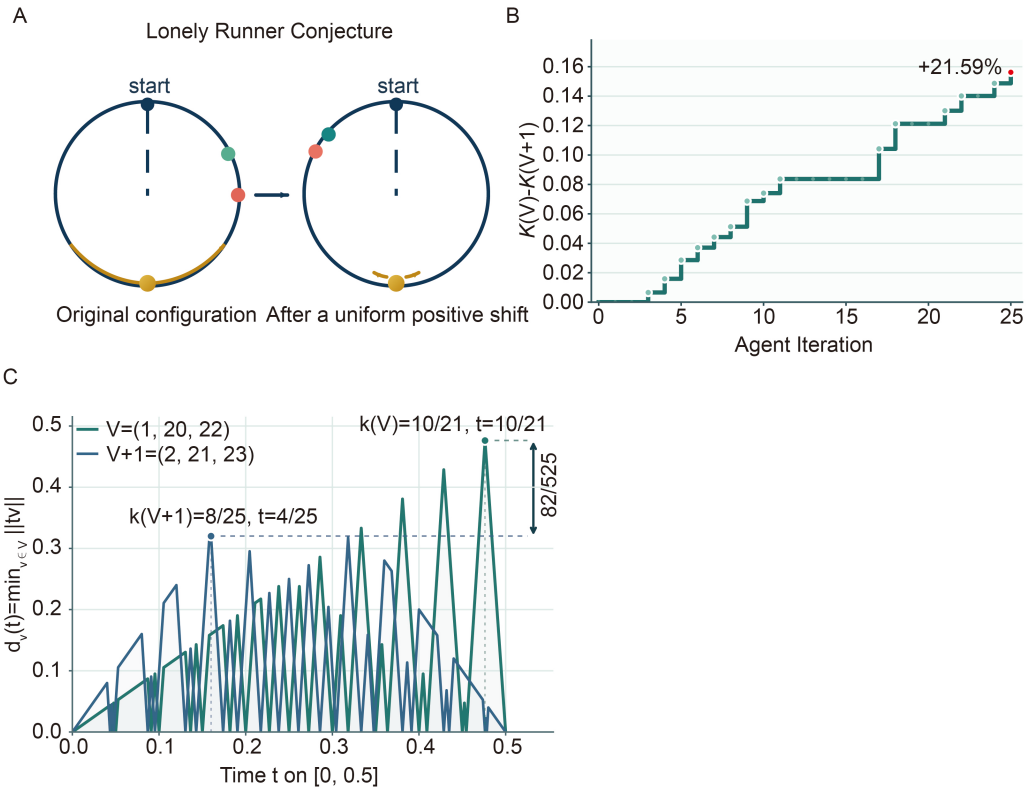

Figure 2: **Searching for a decrease in Lonely Runner separation. A:** Uniform speed translation. **B:** Last accepted drop over 25 calls and 15 score bands; 21.59% compares the final result with the reference triple $(7, 18, 26)$. **C:** Exact distance envelopes for $(1, 20, 22)$ and $(2, 21, 23)$; their maxima differ by $82/525$.

and therefore

$$\Delta(V) = \frac{10}{21} - \frac{8}{25} = \frac{82}{525} \approx 0.156190. \tag{6}$$

Since $82/525 > 0$, this is an explicit counterexample to the monotonicity inequality posed by Perarnau & Serra (2025), even when both speed sets are primitive. Both speed sets also satisfy the Lonely Runner threshold $1/4$, locating the counterexample specifically in translation monotonicity.

The drop is 21.59% larger than that of the reference triple $V = (7, 18, 26)$, which achieves $65/506$. For comparison with the best attainable value in this search domain, exhaustive enumeration of its 2,906 feasible triples gives $133/837$, attained by $(1, 26, 28)$. The final accepted witness reaches 98.29% of this finite-domain optimum.

6.2 Erdős–Straus Conjecture: An Improved Sieve-Weight Lower Bound

The Erdős–Straus conjecture asks whether every integer $n \geq 2$ admits positive integers x, y, z such that $4/n = 1/x + 1/y + 1/z$. One approach studies how many integers remain without a representation after searching a restricted family of witnesses. In the two-parameter sieve of Dahan (2026), the strength of the resulting estimate depends on the weight

$$W(J) = \sum_{\substack{1 \leq u,a \leq J \\ \gcd(u,a)=1}} \frac{1}{\varphi(4ua)}, \tag{7}$$

where φ is Euler's totient function. Lemma 4.22 of that work gives $W(J) \geq (3/(2\pi^2) + o(1))(\log J)^2$, with $3/(2\pi^2) \approx 0.15198$, and Section 7 asks for a lower-bound coefficient of at least 0.45.

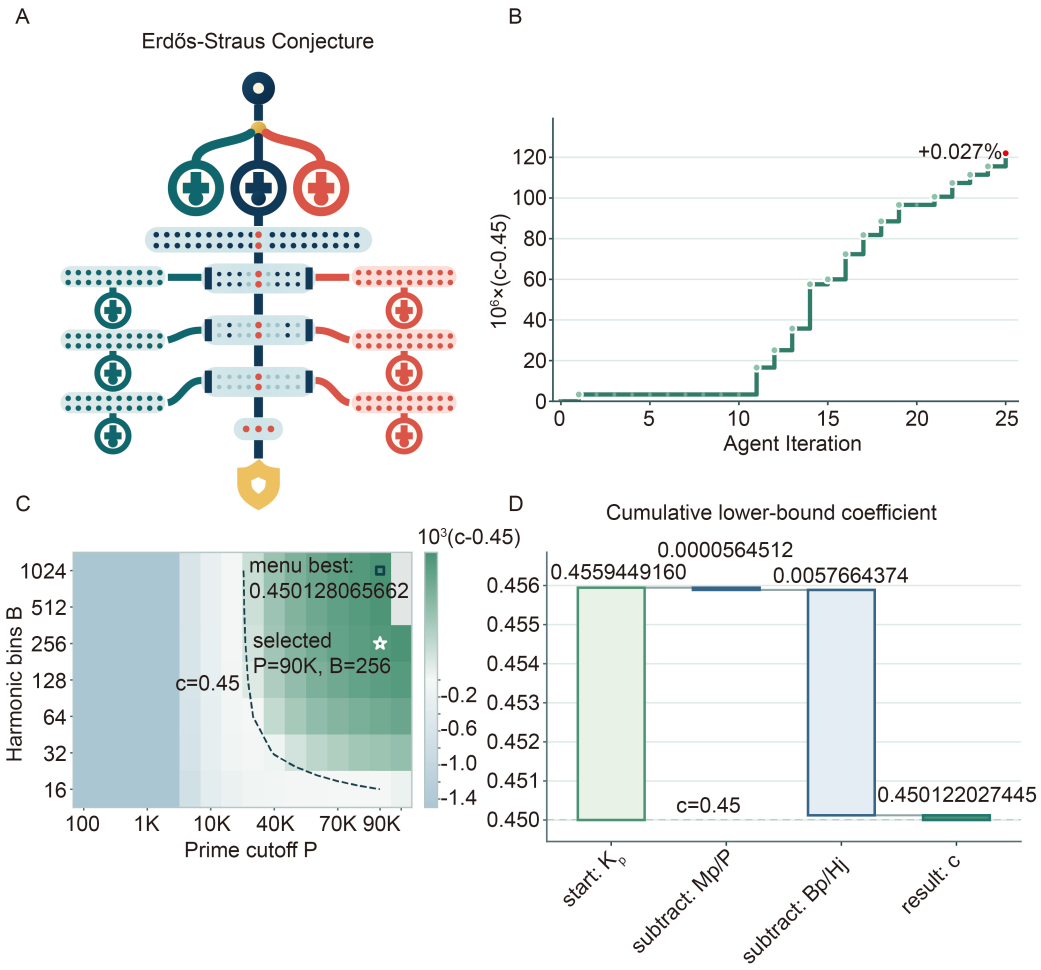

Figure 3: **An explicit lower bound for the Erdős–Straus sieve weight. A:** Problem schematic. **B:** Accepted gains over 25 calls, relative to 0.45. **C:** Parameter menu, with the selected point (star) and optimum (square). **D:** Main term minus prime-tail and harmonic losses gives 0.450122027445; the harmonic deduction uses $L_B \le H_J$.

We approach this question through a finite-prime certificate. Writing $H_J = \sum_{k=1}^{J} 1/k$, the argument separates a main contribution K_P from the loss caused by truncating the primes at P and the error in a finite harmonic approximation:

$$\frac{W(J)}{H_J^2} \ge K_P - \frac{M_P}{P} - \frac{B_P}{L_B}, \qquad J \ge 2^{1246}. \tag{8}$$

Here $L_B \le H_J$ is a computable lower bound obtained by partitioning dyadic intervals into B bins. Starting from this reduction, the agent adjusts P and B, uses exact certificate values as feedback, and refines its parameter choices, balancing prime-tail truncation against harmonic-approximation accuracy. Appendix C derives the bound and gives the arithmetic certificate.

Across 25 model calls, the accepted coefficient rises above 0.45 and reaches 0.450122027445 (Figure 3B). The final choice is $P = 90000$ and $B = 256$; Figure 3D shows the certificate's main term and two deductions. Exact arithmetic, with all rounding directed conservatively, establishes

$$J \ge 2^{1246} \quad \Longrightarrow \quad W(J) \ge 0.450122027445\,(\log J)^2. \tag{9}$$

This is an absolute increase of 0.000122027445 over 0.45, or 0.0271%, matching the rounded improvement shown in the figure.

This strengthens Dahan's sieve-weight coefficient and supplies an explicit range for the 0.45 target (Dahan, 2026).

6.3 Beat-the-Average: A Sharper Certified Probability Bound

How often can twice one random draw exceed the sum of three independent draws from the same distribution? We consider the four-variable problem studied by Bellec & Fritz (2026),

$$C = \sup_{\mu \in \mathcal{P}([0,\infty))} \Pr_{X_1,\ldots,X_4 \overset{\text{iid}}{\sim} \mu} [X_1 + X_2 + X_3 < 2X_4]. \tag{10}$$

The optimization ranges over all distributions on the nonnegative reals. Bellec and Fritz establish $0.400695 \le C \le 673/1615$ and conjecture that their lower bound is sharp. We target the upper bound: a successful search must rule out a larger probability for every admissible distribution.

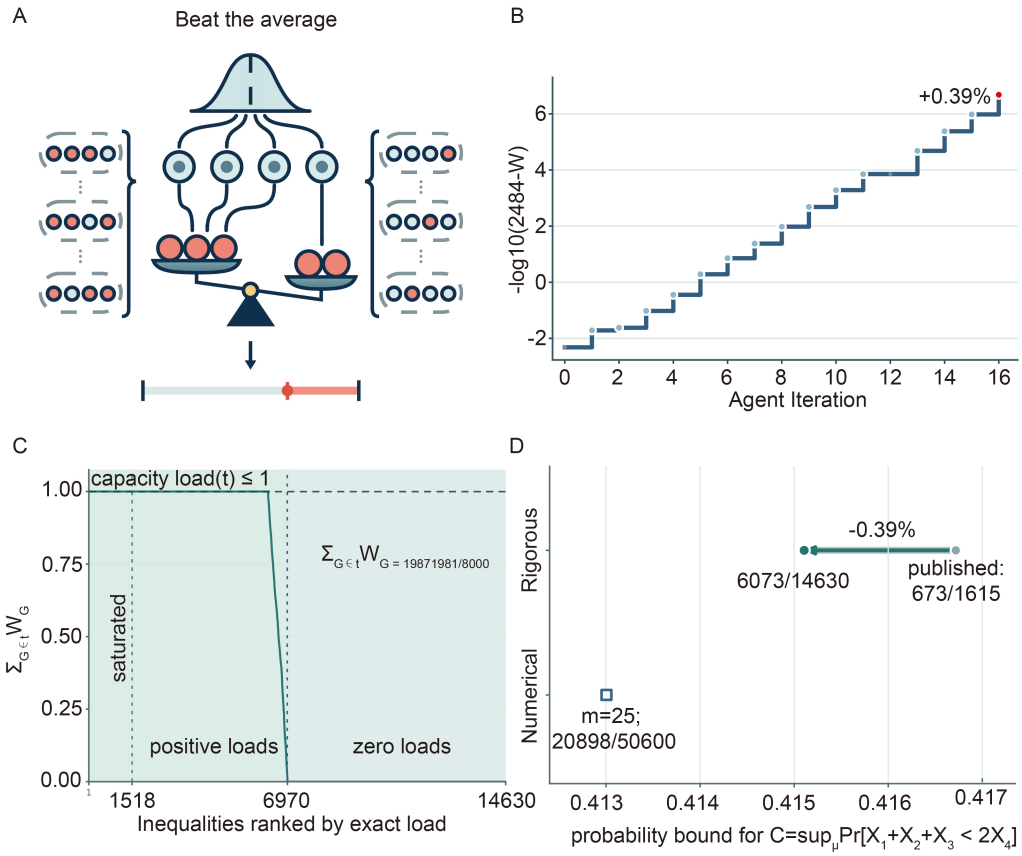

Figure 4: **A sharper certified bound for the four-variable Beat-the-Average problem. A:** Problem schematic. **B:** Weight refinement over 16 iterations, shown as $-\log_{10}(2484 - W)$. **C:** Exact loads, each at most one. **D:** Rigorous and unproved numerical bounds (Bellec & Fritz, 2026); 0.39% is the relative improvement over the previous rigorous bound.

Our search builds on their reduction to a finite system of inequalities. For $m = 22$ sorted nonnegative values, 14630 potentially satisfiable inequalities remain after symmetry and ordering are taken into account. Any group of inequalities that cannot hold simultaneously forces at least one failure. The key change is to allow such groups to overlap: instead of selecting only disjoint groups, we assign them nonnegative rational weights, subject to a total load of at most one on each inequality. The sum of the weights then lower-bounds the number of failures. The search lifts smaller certificates into larger systems, adds contradictory groups, and reoptimizes their weights. In the subsequent

certificate-refinement stage, the agent proposes denominator choices for rationalizing the weights and receives exact feasibility and weight feedback.

The resulting certificate contains 5438 weighted contradictory groups with total weight $W = 19871981/8000 = 2483.997625$. Exact arithmetic verifies that every load is at most one (Figure 4C). Since the number of failures is an integer, at least $\lceil W \rceil = 2484$ of the 14630 inequalities must fail. The finite reduction therefore gives

$$C \leq \frac{14630 - 2484}{2 \cdot 14630} = \frac{6073}{14630} \approx 0.4151059467. \tag{11}$$

Two independently implemented exact-arithmetic verifiers agree on this certificate; Appendix D gives the proof. The refinement in Figure 4B brings W to $2484 - 13/62500000$, tightening the certificate while retaining the same integer-strengthened probability bound.

Equation 11 improves the rigorous bound $673/1615 \approx 0.4167182663$ of Bellec & Fritz (2026) by $401/248710 \approx 0.00161232$, a relative reduction of 0.3869%. Their smaller numerical value $20898/50600 \approx 0.413004$ remains unproved and is shown separately in Figure 4D. Our certificate establishes the improved rigorous upper bound uniformly over the full distribution class.

Research decisions across the three cases. The cases turn verification feedback into distinct research decisions. In Lonely Runner, separation scores guide the choice of speed neighborhoods, leading to a primitive witness with a positive translation drop. In the sieve-weight study, the certificate exposes two sources of loss, making the prime cutoff and harmonic resolution concrete choices for improving the lower bound. In Beat-the-Average, feasibility and total-weight feedback guide rational certificate refinement; the integer failure count connects that certificate to a distribution-uniform probability bound. The shared experimental structure links a proposed change to a measurable outcome and retains the resulting evidence for the next decision. Each study culminates in an explicit mathematical object that establishes its reported result.

7 Limitations

Our evaluation covers four generated graph-optimization tasks and three mathematical problems under fixed API-call budgets. Larger instances, additional model backbones, and longer research campaigns would further characterize scaling and robustness. The three data groups capture variation across instance seeds and model outputs; additional repetitions would improve precision. For mathematical discovery, progress also depends on the available problem reductions and exact verification tools.

8 Conclusion

EVIDRA unifies evidence-guided intervention selection, scoped execution, and verification in an adaptive research process. By linking changes to prompts, workflows, context, memory, tools, and sampling with hypotheses and recoverable baselines, it makes experimental evidence actionable throughout research. The resulting records make successful interventions and failed trials available for subsequent decisions, linking the agent's current choices to accumulated experimental evidence.

Across four graph-optimization tasks, EVIDRA achieves the highest aggregate score of 0.8736, leading on Cluster Editing and Treedepth. Three mathematical studies produce a translation-monotonicity counterexample, an improved sieve-weight certificate, and a tighter certified probability bound. The graph tasks measure the quality of the resulting algorithms, while the mathematical cases connect search progress to explicit witnesses and certificates. Together, these outcomes show how a common experimental loop can organize different forms of research, with each proposed change tied to a hypothesis, a recoverable starting point, and an independently assessable result.

This design also provides a reusable interface for new research tasks: task-specific objectives, editable components, and verification procedures fit within the same controller. It offers a concrete foundation for developing and comparing adaptive research strategies.

AI USE STATEMENT

Generative AI assisted with implementing and debugging research code, discussing experimental designs, and drafting and polishing the manuscript. AI tools also assisted with editing schematic figures. As part of the reported solver benchmarks, DeepSeek API models proposed candidate programs. AI assistance also supported mathematical exploration; model suggestions were not treated as verification. Candidate solutions were checked by the task-specific evaluators, and the mathematical witnesses and certificates were assessed using the exact-arithmetic procedures described in the appendices. Reported tables were checked against stored experiment records. The authors are responsible for the final text, claims, and artifacts, including material produced with AI assistance.

ETHICS STATEMENT

The tasks use generated graphs and abstract mathematical objects, without human-subject data. Candidate programs execute with resource limits and network isolation; trusted code independently verifies solutions and scores. These safeguards apply to the evaluated setting. Released logs, code, and data require credential removal and anonymization. Mathematical claims are restricted to statements supported by their witnesses and certificates.

REPRODUCIBILITY STATEMENT

Section 5.1 defines the tasks, splits, interface, and score. Appendix A.1 specifies controller defaults and executable search steps; Appendices E and F give score aggregation, individual ablation scores, and comparison scope. Appendix G records versions, data seeds, execution limits, and experimental procedures; Appendix H defines the search-trajectory analysis. Appendices B, C, and D provide the mathematical reductions and verification procedures.

REFERENCES

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RQm2KQTM5r.

Pierre C. Bellec and Tobias Fritz. On maximizing the probability of a linear inequality between iid random variables. *The American Mathematical Monthly*, 2026. doi: 10.1080/00029890.2026.2702871. URL https://arxiv.org/abs/2412.15179v5. Published online August 7, 2026. Preprint: arXiv:2412.15179v5.

Mert Cemri, Shubham Agrawal, Akshat Gupta, Shu Liu, Audrey Cheng, Qiuyang Mang, Ashwin Naren, Lutfi Eren Erdogan, Koushik Sen, Matei Zaharia, Alex Dimakis, and Ion Stoica. AdaEvolve: Adaptive LLM driven zeroth-order optimization. *arXiv preprint arXiv:2602.20133*, 2026. URL https://arxiv.org/abs/2602.20133.

Benjamin Dahan. Sieve dimension and search depth for the Erdős–Straus conjecture, $n \equiv 1 \pmod{24}$, 2026. URL https://arxiv.org/abs/2608.24035v1. arXiv:2608.24035v1.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. Generalization in adaptive data analysis and holdout reuse. *arXiv preprint arXiv:1506.02629*, 2015. URL https://arxiv.org/abs/1506.02629.

Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis, and Pushmeet Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610:47–53, 2022. doi: 10.1038/s41586-022-05172-4. URL https://www.nature.com/articles/s41586-022-05172-4.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=t9U3LW7JVX.

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 state-of-the-art pipelines. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=sY5N0zY5Od.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. ShinkaEvolve: Towards open-ended and sample-efficient program evolution. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=lKEdGCoDNC.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024. URL https://arxiv.org/abs/2408.06292.

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. Towards end-to-end automation of AI research. *Nature*, 651:914–919, 2026. doi: 10.1038/s41586-026-10265-5. URL https://www.nature.com/articles/s41586-026-10265-5.

Daniel J. Mankowitz, Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, Edouard Leurent, Shariq Iqbal, Jean-Baptiste Lespiau, Alex Ahern, Thomas Köppe, Kevin Millikin, Stephen Gaffney, Sophie Elster, Jackson Broshear, Chris Gamble, Kieran Milan, Robert Tung, Minjae Hwang, Taylan Cemgil, Mohammadamin Barekatain, Yujia Li, Amol Mandhane, Thomas Hubert, Julian Schrittwieser, Demis Hassabis, Pushmeet Kohli, Martin Riedmiller, Oriol Vinyals, and David Silver. Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, 618:257–263, 2023. doi: 10.1038/s41586-023-06004-9. URL https://www.nature.com/articles/s41586-023-06004-9.

Rui Meng, Bhavana Dalvi Mishra, Jiefeng Chen, Chun-Liang Li, Palash Goyal, Mihir Parmar, Yiwen Song, Yale Song, Rajarishi Sinha, Parthasarathy Ranganathan, Burak Gokturk, Jinsung Yoon, and Tomas Pfister. ScientistOne: Towards human-level autonomous research via chain-of-evidence. *arXiv preprint arXiv:2605.26340*, 2026. URL https://arxiv.org/abs/2605.26340.

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. URL https://arxiv.org/abs/2506.13131.

Guillem Perarnau and Oriol Serra. The Lonely Runner Conjecture turns 60. *Computer Science Review*, 58:100798, 2025. doi: 10.1016/j.cosrev.2025.100798. URL https://doi.org/10.1016/j.cosrev.2025.100798.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475, 2024. doi: 10.1038/s41586-023-06924-6. URL https://www.nature.com/articles/s41586-023-06924-6.

Asankhaya Sharma. OpenEvolve: An open-source evolutionary coding agent. Open-source software, 2025. URL https://github.com/algorithmicsuperintelligence/openevolve. Repository documentation accessed September 25, 2026.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

Armando Solar-Lezama, Liviu Tancau, Rastislav Bodik, Vijay Saraswat, and Sanjit Seshia. Combinatorial sketching for finite programs. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 404–415, 2006. URL https://people.eecs.berkeley.edu/~sseshia/pubdir/asplos06-final.pdf.

XYZ Agentic Team. AI4AI at scale: A full-pipeline system for enhancing LLM agentic capabilities. Technical report, July 2026. URL https://xyz-lab.ai/blogs/ai4ai-at-scale/. Technical report dated July 24, 2026.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025. URL https://arxiv.org/abs/2504.08066.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639:609–616, 2025. doi: 10.1038/s41586-025-08661-4. URL https://www.nature.com/articles/s41586-025-08661-4.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel machine: Open-ended evolution of self-improving agents. In *International Conference on Learning Representations*, 2026a. URL https://arxiv.org/abs/2505.22954.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui 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. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=z5uVAKwmjf.

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. In *International Conference on Learning Representations*, 2026b. URL https://arxiv.org/abs/2510.04618. Author's camera-ready version, arXiv:2510.04618v3.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

A Implementation Details

A.1 Proposal Selection and Experimental Settings

Target registry and proposals. The six process families in Equation 1 have distinct roles: Prompt specifies instructions and examples; Workflow specifies execution and retry structure; Context determines the current step's information; Memory governs persistent-experience selection and use; Tools specifies editable interfaces and wrappers; Sampling specifies candidate-generation settings. The evidence ledger and frozen judge remain read-only throughout these interventions. A proposal names one registered target, its immutable baseline, and declared edit scope, together with observations expected to support or disfavor its hypothesis. It also estimates improvement, success likelihood, information value, cost, and regression risk. A process target selects a component indexed by $\mathcal{O}$; an Artifact target selects the output a. The registered target specifies the editable content and paths for the scoped change $\Delta(q)$. The controller removes duplicate proposals within the current batch, rejects unknown targets or unauthorized edits, and retains execution authority. Repetition of a previously attempted proposal is handled separately by the history penalty $D_t(q)$ in Equation 3.

Empirical ranking updates. The target-aware ranking in Equation 3 uses the following empirical adjustments. Let p_q be the proposal's success estimate, and let n_α, v_α count attempts and valid results for its action type α. If $n_\alpha > 0$, the first empirical adjustment is

$$p^{(1)} = (1 - w_\alpha)p_q + w_\alpha \frac{v_\alpha + 1}{n_\alpha + 2}, \qquad w_\alpha = \min(0.75, n_\alpha/10). \tag{12}$$

Otherwise $p^{(1)} = p_q$. When compatible prior action statistics $(n_\alpha^{\text{prior}}, v_\alpha^{\text{prior}})$ are available, the second adjustment is

$$p^{(2)} = (1 - w_{\text{prior}})p^{(1)} + w_{\text{prior}} \frac{v_\alpha^{\text{prior}} + 1}{n_\alpha^{\text{prior}} + 2}, \qquad w_{\text{prior}} = \min(0.25, n_\alpha^{\text{prior}}/40). \tag{13}$$

With no prior observations, $p^{(2)} = p^{(1)}$. For target o, let N_o count completed attempts excluding infrastructure failures and K_o count improvements over the development incumbent. If $N_o > 0$,

$$\widetilde{p}_q = (1 - w_o)p^{(2)} + w_o \frac{K_o + 1}{N_o + 2}, \qquad w_o = \min(0.35, N_o/12); \tag{14}$$

otherwise $\widetilde{p}_q = p^{(2)}$. The blend is a ranking heuristic combining action validity and target-level improvement. Information gain enters as the proposal's estimated value of the evidence an intervention would produce.

The implementation defaults are $\lambda_g = \lambda_I = \lambda_r = 1$, $\kappa = 0.15$, $\lambda_{\text{rep}} = 10$, and $\varepsilon = 10^{-9}$. Proposals with equal utility are ordered by their recorded identifiers. Duplicate proposals are identified from their action, payload, hypothesis, baseline lineage, target, intervention scope, fidelity, evaluation suite, experimental role, attribution record, and seed-set identity.

Execution, attribution, and evidence memory. The controller recomputes the actual content diff and checks registered targets, declared paths, and operation or source-line limits. Empty and out-of-scope edits are rejected as intervention evidence. Improved development artifacts enter the archive with a separate acceptance status. Attribution starts as *pending*; paired rollback restores baseline content and reuses the recorded seed-set identity. Confirmation requires the advantage to disappear under the domain comparison rule. Incomplete checks retain distinct development, attribution, and acceptance statuses. The ledger records assessment outcomes separately from rollback evidence.

Retrieval requires a matching domain-family fingerprint and ranks records by contextual overlap and reliability. Valid non-improvements and failed interventions remain stored with hypotheses, applicability conditions, outcomes, counterexamples, and verification status. Infrastructure failures are recorded separately and excluded from scientific success rates. Rollback updates attribution labels while preserving observations. Research hypotheses, target statistics, and long-term memory use development evidence. Independent validation has an explicit access quota and stores diagnostics separately. Final sealed testing, when configured, evaluates a committed artifact after search is complete.

Executable solver-search loop. For the graph tasks, EVIDRA executes the following search loop.

1. Freeze the task contract, development and evaluation suites, initial solver, scoring rule, verifier, and resource limits. Start without experience from previous runs and evaluate the initial solver.
2. Select the mutation parent under the configured search policy; if an attribution check is due, instead request an exact rollback to its recorded parent. Construct context using development evidence only.
3. Obtain a structured model proposal. Count the API call whether it succeeds or fails. Reject malformed output, invalid targets, parent mismatches, and unauthorized or empty edits.
4. Apply Equation 3 to the eligible proposal. The graph-task implementation uses a proposal batch size of one.
5. Execute the candidate in the sandbox. Outside the candidate process, verify returned solutions and recompute development scores.
6. Record the proposal, source digest, parent, observations, and failure or attribution status. Update the best verified development artifact and the configured branch state. A rollback updates attribution separately from the stored best score.
7. Repeat until the API-call limit or a method-specific stopping condition. Fix the valid solver with the highest development score and record its source hash.
8. Evaluate that frozen solver on evaluation instances, record its evaluation results and final score, and do not feed evaluation diagnostics back into the completed search.

Parent selection and restarts. An episode spans six completed responses, including rejected model outputs and rollback responses; transport failures do not advance the completed-response counter. The first three episodes start independently from the initial solver. Subsequent episodes alternate exploration from other promising programs with periodic independent restarts. Parents are drawn from at most three AST-distinct valid programs within $0.15\max(1, |s_{\text{best}}|)$ of the incumbent development score. A branch can continue improving below the global incumbent; two consecutive rejected responses abandon its active continuation. Up to six recent candidate records and 16,000 bytes of source code from failed candidates can be included for diagnosis. The w/o Process Adaptation and w/o Propose ablations alter the feedback and parent-selection rules specified in Appendix F, while retaining the task verifier.

B Lonely Runner Search and Exact Verification

The search used `deepseek-flash` to propose one to three integer anchor triples and a coordinate radius in $\{1, \dots, 6\}$. The tool expanded their neighborhoods within $[1, 30]$, retained distinct increasing triples, and ordered them by distance to an anchor, with deterministic tie-breaking. Triples failing either primitive-set condition were rejected; the remaining scores were computed as exact fractions. The model received the current target band, previous acceptances, and nearby unsuccessful candidates as feedback. Its initial instructions included the heuristic of trying a small first speed and varying the gaps between the other speeds.

To test repeated improvement, 15 successively higher target bands were fixed before the first model call. They were $(b_{j-1}, b_j]$, where $b_0 = 0$ and

$$\begin{aligned}(b_1, \dots, b_{15}) = (&0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08,\\ &0.095, 0.11, 0.125, 0.137, 0.145, 0.152, 0.16).\end{aligned} \tag{15}$$

These endpoints were represented as exact fractions. A pre-run enumeration checked that each band contained feasible triples before the protocol was frozen. The model was not supplied the enumerated maximizing triple. For each call, the tool stopped at the first previously unseen feasible triple in the active band. Previously evaluated candidates could not be accepted again. Candidates outside that band were logged even if their scores exceeded its upper endpoint; consequently, the curve in Figure 2B shows the last accepted score, not the maximum over every candidate examined by the tool. Across the 25 completed calls, the log contains 4,070 candidate visits and 1,753 distinct triples, including those rejected by the primitive-set checks.

Exact verification uses the piecewise-linear form of d_V. A maximum occurs at an intersection of increasing and decreasing distance segments, or at a peak of one segment. It therefore suffices to evaluate

$$T(V) = \left\{ \frac{m}{u+v} : u, v \in V,\ u \le v,\ m \in \mathbb{Z},\ 0 \le m \le \left\lfloor \frac{u+v}{2} \right\rfloor \right\}. \tag{16}$$

Allowing $u = v$ includes the individual peaks. This is the finite-time reduction discussed by Perarnau & Serra (2025, Section 2). Evaluating $d_V(t)$ at every $t \in T(V)$ with rational arithmetic checks global maximality on $[0, 1/2]$, rather than only evaluating a proposed time. The same evaluator supplies the earlier-witness score and the exhaustive finite-domain reference used in the main text.

C Finite Certificate for the Erdős–Straus Sieve Weight

For integer $J \ge 1$, let U, A be independent with $\Pr(U = k) = \Pr(A = k) = 1/(kH_J)$ for $1 \le k \le J$. With $g(u, a) = \mathbf{1}_{\gcd(u,a)=1} ua/\varphi(4ua)$, direct substitution gives $W(J)/H_J^2 = \mathbb{E}g(U, A)$. Set $x_p = \mathbf{1}_{p|u}$ and $y_p = \mathbf{1}_{p|a}$. Retaining only prime factors up to P gives

$$g_P(u, a) = \frac{1}{2}(1 - x_2 y_2) \prod_{\substack{3 \le p \le P \\ p \text{ prime}}} \left(1 + \frac{x_p + y_p}{p-1} - \frac{p+1}{p-1} x_p y_p\right). \tag{17}$$

Each factor is nonnegative, and $g_P \le M_P := \frac{1}{2}\prod_{3 \le p \le P} p/(p-1)$. Unless u, a share a prime greater than P, the omitted factors are at least one and $g \ge g_P$. Since $\Pr(p \mid U) = H_{\lfloor J/p \rfloor}/(pH_J) \le 1/p$,

independence and the union bound imply

$$\mathbb{E}g \geq \mathbb{E}g_P - M_P \sum_{p>P} \frac{1}{p^2} \geq \mathbb{E}g_P - \frac{M_P}{P}. \tag{18}$$

All products and prime sums below have the same odd-prime range $3 \leq p \leq P$ unless specified otherwise.

Expand $g_P = \sum_{r,s} \alpha_P(r,s)\mathbf{1}_{r|u}\mathbf{1}_{s|a}$. The harmonic integral bounds give, for every integer $r \geq 1$,

$$0 \leq \frac{1}{r} - \frac{H_{\lfloor J/r \rfloor}}{rH_J} \leq \frac{1+\log r}{rH_J}, \qquad H_0 = 0. \tag{19}$$

Replacing the divisibility probabilities by $1/r$ and $1/s$ therefore incurs a loss of at most $\mathcal{B}_P/H_J$, where

$$K_P = \sum_{r,s} \frac{\alpha_P(r,s)}{rs} = \frac{3}{8}\prod_p (1+p^{-2}), \tag{20}$$

$$\mathcal{B}_P = \sum_{r,s} \frac{|\alpha_P(r,s)|}{rs}(2+\log r+\log s) = A_P(2+D_P), \tag{21}$$

$$A_P = \frac{5}{8}\prod_p a_p, \qquad D_P = \frac{2}{5}\log 2 + \sum_p \frac{d_p}{a_p}\log p, \tag{22}$$

$$a_p = 1 + \frac{2}{p(p-1)} + \frac{p+1}{p^2(p-1)}, \qquad d_p = \frac{2}{p(p-1)} + \frac{2(p+1)}{p^2(p-1)}. \tag{23}$$

These formulas follow by multiplying the finite local polynomials, including the absolute values of their coefficients. They give $\mathbb{E}g \geq K_P - M_P/P - \mathcal{B}_P/H_J$ without assuming independence of divisibility by different primes under harmonic sampling.

For computation, a rational upper bound $B_P \geq \mathcal{B}_P$ is obtained by bounding each logarithm. Write $n = 2^k x$ with $1 \leq x < 2$ and $z = (x-1)/(x+1) \leq 1/3$. The first ten terms of $\log x = 2\sum_{j\geq 0} z^{2j+1}/(2j+1)$ have remainder at most $2z^{21}/(21(1-z^2))$. The same formula at $z = 1/3$ bounds $\log 2$. The resulting rational upper bounds for $\log n$ are rounded upward to multiples of 10^{-12} before evaluating B_P.

For a power of two $B \leq 2^{1246}$, divide each dyadic interval $(n, 2n]$, $B \mid n$, into B equal bins. Its harmonic sum is at least $H_{2B} - H_B$. Summing from B to 2^{1246} yields, for $J \geq 2^{1246}$,

$$H_J \geq L_B := H_B + (1246 - \log_2 B)(H_{2B} - H_B). \tag{24}$$

Substitution establishes Equation 8. The search respects the resource constraint $\pi(P) + B \leq 10000$, where $\pi(P)$ counts primes up to P. The selected $P = 90000$ and $B = 256$ use $8713 + 256 = 8969$ resource units. In Figure 3D, the main term is approximately 0.455944916040, and the prime-tail and harmonic deductions are 0.000056451183 and 0.005766437412, respectively. For $P = 90000$ and $B = 256$, finite arithmetic gives the conservative bounds

$$\begin{aligned} K_P &\geq 0.455944916039870, \\ M_P/P &\leq 0.000056451182976, \\ B_P/L_B &\leq 0.005766437411564. \end{aligned} \tag{25}$$

The first number minus the latter two is exactly $0.450122027445330 > 0.450122027445$. Together with $H_J \geq \log J \geq 0$, this proves Equation 9. The displayed terminating decimals are rational bounds, not floating-point equality claims. Their directions were also checked using integer interval arithmetic on a 10^{-60} grid, with outward rounding at every multiplication and division.

D An Exact Certificate for the Beat-the-Average Bound

Fix $m = 22$ and $0 \leq x_1 \leq \cdots \leq x_m$. Define

$$\mathcal{T} = \{(i,j,k,\ell) : 1 \leq i < j < k \leq m,\ j < \ell \leq m,\ \ell \neq k\}. \tag{26}$$

Each $t = (i, j, k, \ell)$ represents $x_i + x_j + x_k < 2x_\ell$. For each four-element index set, the distinguished index ℓ can occupy either of its two largest positions, so $|\mathcal{T}| = 2\binom{22}{4} = 14630$. The other two choices cannot satisfy the strict inequality: two terms on the left are at least x_ℓ and the third is nonnegative.

For $q \in \{1, \dots, m\}$, put

$$F(q, t) = \mathbf{1}_{q \le i} + \mathbf{1}_{q \le j} + \mathbf{1}_{q \le k} - 2\mathbf{1}_{q \le \ell}. \tag{27}$$

A nonempty group $G \subseteq \mathcal{T}$ is contradictory whenever $\sum_{t \in G} F(q, t) \ge 0$ for every q. Indeed, write $d_1 = x_1$ and $d_q = x_q - x_{q-1}$ for $q > 1$. These gaps are nonnegative and

$$\sum_{(i,j,k,\ell) \in G} (x_i + x_j + x_k - 2x_\ell) = \sum_{q=1}^{m} d_q \sum_{t \in G} F(q, t) \ge 0. \tag{28}$$

If every strict inequality in G held, the left-hand side would instead be negative, a contradiction. This test is the summation criterion used in the certificate construction of Bellec & Fritz (2026).

Now assign a rational weight $w_G > 0$ to each certified group and require

$$L_t := \sum_{G \ni t} w_G \le 1 \quad (t \in \mathcal{T}), \qquad W := \sum_{G} w_G. \tag{29}$$

For any fixed sorted vector, let z_t equal one if inequality t fails and zero otherwise. Every group contains a failure, hence

$$W \le \sum_{G} w_G \sum_{t \in G} z_t = \sum_{t \in \mathcal{T}} z_t L_t \le \sum_{t \in \mathcal{T}} z_t. \tag{30}$$

Thus at least $\lceil W \rceil$ inequalities fail. The computed rational certificate has 5438 groups and 15802 group–inequality incidences. Direct evaluation gives $W = 19871981/8000$, $\max_t L_t = 1$, 1518 loads equal to one, 5452 strictly between zero and one, and 7660 zero loads, for 6970 positive loads in total (Figure 4C). In particular, at most $14630 - \lceil 19871981/8000 \rceil = 12146$ inequalities in $\mathcal{T}$ can hold.

To transfer this deterministic statement to probability, take 22 independent draws from an arbitrary probability measure μ on $[0, \infty)$. For any realization, sort the draws, breaking ties arbitrarily. There are $22 \cdot 21 \cdot 20 \cdot 19 = 175560$ ordered choices of four distinct sample indices. Each satisfiable inequality in $\mathcal{T}$ accounts for six choices, corresponding to permutations of the three terms on the left; all other choices fail by the ordering argument above. Therefore the sum of the 175560 event indicators is at most $6 \cdot 12146 = 72876$ for every realization. Every ordered choice has the same event probability because the draws are iid. Taking expectations proves

$$175560 \Pr_{\mu^{\otimes 4}}[X_1 + X_2 + X_3 < 2X_4] \le 72876, \tag{31}$$

which yields Equation 11 for every μ. The argument permits atoms and ties and imposes no moment or bounded-support assumption.

We checked the finite certificate with two independent exact-arithmetic implementations. The first evaluates $\sum_{t \in G} F(q, t)$ directly for every group G and coordinate q. The second constructs each group's coefficient vector in the variables $x_1, \dots, x_{22}$ and checks that all suffix sums are nonnegative. Both verify positive rational weights, distinct group members belonging to $\mathcal{T}$, the capacity constraints in Equation 29, and the total weight $W = 19871981/8000$. These checks use rational arithmetic throughout and do not rely on floating-point optimizer tolerances. The complete group and rational-weight list, together with both standalone verifiers, is provided in the anonymous supplement *Exact Probability Certificate*.

E Benchmark Scoring and Aggregation

Gap-closure score. For initial, reference, and candidate objectives s, r, and c, with $r \le s$, the complete score is

$$g(s, r, c) = \begin{cases} (s - c)/(s - r), & s > r, \\ (s - c)/\max(|s|, 1), & s = r. \end{cases} \tag{32}$$

The second branch handles an initial algorithm that already matches the reference, using the initial objective's scale instead of a zero gap.

Evaluation protocol. All methods use the task instances, initial solver, reference solver, trusted checker, and execution constraints described in Section 5.1. A candidate returns a solution; the checker verifies feasibility and recomputes the objective. The evaluation score averages Equation 32 over the instances in a task/block. Any invalid solution or execution failure assigns that task/block a score of -1.

Summary statistics. Each task entry in Table 1 reports the arithmetic mean and sample standard deviation of three evaluation scores, using denominator $3 - 1$ for the variance. The overall score equally weights the four task means. Calculations use unrounded values; the table displays four decimal places and marks the largest mean in each column in bold. Before rounding, EVIDRA's overall mean exceeds OpenEvolve's by 0.005421. Scores are not clipped: a score above one improves on the reference solver, which need not be globally optimal.

Experimental records. Each evaluation record identifies the submitted solver, search history, and stored model interactions. These distinguish model proposals and explanations from execution feedback and verification outcomes. Appendix G gives the instance seeds, baseline versions, execution settings, and verification procedures.

F Component Ablation Details

F.1 Protocol and Intervention Scope

The study evaluates two ablations across four tasks and three prespecified data groups, giving 24 search runs. The groups use different instance-generation and search seeds and are identified as 02, 04, and 05. Each run starts independently from the initial solver with fresh per-run experience. We report all three blocks, including identical scores. The development/evaluation counts are 8/4 for OCM and DFVS, 10/5 for Cluster Editing, and 6/3 for Treedepth; these are the generated TaskCapsule suites described in Section 5.1, using small and medium instances.

Each ablation run on a task and data group has a cap of 100 API calls to `deepseek-flash`, with temperature 0.7, a prompt ceiling of 65,000 tokens, and an output ceiling of 6,000 tokens. Failed requests and model calls used for rollback count toward this cap. The proposal batch size is one. Candidate execution uses remote Linux CPUs with a per-instance CPU limit of 10 seconds and memory limit of 512 MiB, network isolation, a cleared environment, and a read-only job directory. The valid solver with the highest development score is fixed by its source hash before final evaluation. In total, 2,400 requests were attempted: 2,396 completed and four encountered HTTP 429 transport failures. All 24 final submissions were valid and passed checks linking scores to the submitted source code.

w/o Process Adaptation. The proposer sees evaluator validity and the aggregate score, but not dynamic family-level scores, error categories, or detailed diagnostics. It retains its hypotheses, solver code, past aggregate scores, parent selection, verification, and rollback checks. The intervention changes the evaluator information exposed to the proposer.

w/o Propose. Except for rollback checks, each new candidate starts from the best program found so far. Branches from other parents and independent restarts are disabled together, while complete solver rewrites remain permitted. Detailed feedback, search history, edit-scope checks, verification, and final solver selection are retained.

Both arms retain rollback, the verifier, sandboxing, and final evaluation. Rollback updates attribution status separately from the stored development best. The ablations change feedback and parent selection while keeping these verification mechanisms fixed.

F.2 Final Evaluation Results

Table 3 contains both ablation arms. The benchmark reference is copied unchanged from Table 1; its aggregation is specified in Appendix E. Comparisons to this reference are descriptive; a comparison with controls using matching settings appears below.

Table 3: Final evaluation scores for the two ablations. Task entries are mean ± sample SD; Avg. equally weights the four task means. CE denotes Cluster Editing. All ablation arms use blocks 02, 04, and 05; the reference row retains the aggregation of Table 1.

Policy	OCM	DFVS	CE	Treedepth	Avg.
EVIDRA (Table 1)	0.9978 ± 0.0019	0.4630 ± 0.2581	0.9823 ± 0.0159	1.0515 ± 0.0187	0.8736
w/o Process Adaptation	1.0000 ± 0.0000	0.3297 ± 0.2331	0.9313 ± 0.1191	1.0515 ± 0.0326	0.8281
w/o Propose	0.9114 ± 0.1234	0.2947 ± 0.3048	0.7094 ± 0.5033	0.8912 ± 0.0962	0.7017

Aggregation and variation. Scores follow Equation 32 and are not clipped; any invalid evaluation instance makes its entire task/block score -1. Sample SD uses denominator $n - 1 = 2$. For an ablation's overall SD, we first average the four tasks within each block, then compute SD over the three block averages. The resulting means and SDs are 0.828100 ± 0.075042 for w/o Process Adaptation and 0.701671 ± 0.035152 for w/o Propose. The Table 1 reference supplies the benchmark mean. Blocks use different frozen data seeds, so variation reflects both data-block and model-generation differences. The repeat unit is the block ($n = 3$).

Other taskwise differences. The w/o Process Adaptation aggregate is 0.045536 below the benchmark reference (5.21%), primarily through DFVS and Cluster Editing. Its Treedepth mean equals the reference, but its three scores differ and its SD is larger. w/o Propose is 0.171964 below the reference (19.68%), with lower means on all four tasks. Both percentages express relative changes in normalized gap-closure score.

F.3 Individual Scores and Matched Controls

Table 4: Individual evaluation scores for the ablations and EVIDRA control, including all three blocks and rounded to six decimals. The control rows are separate from the EVIDRA summary in Table 1.

Policy	Task	Block 02	Block 04	Block 05
w/o Process Adaptation	OCM	1.000000	1.000000	1.000000
	DFVS	0.143016	0.590909	0.255155
	Cluster Editing	0.793750	1.000000	1.000000
	Treedepth	1.068254	1.013889	1.072222
w/o Propose	OCM	0.968707	0.995648	0.769765
	DFVS	0.624148	0.022727	0.237242
	Cluster Editing	0.128205	1.000000	1.000000
	Treedepth	0.955556	0.937500	0.780556
EVIDRA control	OCM	0.994218	0.949710	0.996711
	Cluster Editing	0.792515	1.000000	0.969231
	Treedepth	1.046032	1.020833	1.036111

To match the agent configuration as well as the data, we compare the ablations with separate EVIDRA control runs on OCM, Cluster Editing, and Treedepth. DFVS lacks a control run with the same configuration and is excluded. These controls precede the ablation runs, so execution order was not randomized across methods. The Cluster Editing control in data group 02 includes a correction for an API response exceeding the output limit. The control scores are separate from the benchmark aggregate. The EVIDRA control task means and SDs are 0.980213 ± 0.026446 (OCM), 0.920582 ± 0.111971 (Cluster Editing), and 1.034325 ± 0.012694 (Treedepth).

For each block, we average the three matched tasks before comparing policies. Let Δ_{ctrl} denote the EVIDRA control minus the ablation; positive values favor the EVIDRA control. Table 5 reports the three-task aggregate and the mean and SD of these paired block differences. This separate comparison uses the same agent configuration and task/group set.

Table 5: Comparison with EVIDRA under matching settings on OCM, Cluster Editing, and Treedepth. All SDs use three block-level observations. The last column counts blocks where the EVIDRA control has the higher three-task aggregate.

Policy	Three-task mean ± SD	Δ_{ctrl} mean ± SD	Control higher
EVIDRA control	0.978373 ± 0.030010	—	—
w/o Process Adaptation	0.994235 ± 0.036174	-0.015862 ± 0.006931	0/3
w/o Propose	0.837326 ± 0.147197	0.141047 ± 0.124092	3/3

In block order 02/04/05, the paired differences for w/o Process Adaptation are -0.009746, -0.014449, and -0.023390. For w/o Propose they are 0.260099, 0.012465, and 0.150577. The EVIDRA control exceeds w/o Propose in all three block aggregates. The Cluster Editing advantage is concentrated in block 02; OCM favors w/o Propose in one block. w/o Process Adaptation exceeds the EVIDRA control in every three-task block aggregate.

F.4 Comparison Scope and Reporting

The two interventions isolate feedback exposure and the combined branching-and-restart policy, respectively. Evaluator information changes in the first; candidate-parent choices change in the second. Both retain the verifier, execution limits, and final evaluation procedure.

Tables 3–5 report the aggregate results, individual data-group scores, and matched-control comparisons. Scores follow the aggregation rule in Appendix E; all calculations use unrounded values, with rounding applied only for display.

G Reproduction Details

G.1 Data Generation and Random Seeds

TaskCapsule generates each instance deterministically from a task, structural family, scale, and seed. The development suite uses one small and one medium instance per family; evaluation uses one medium instance per family with a separate seed. The development/evaluation counts per data group are 8/4 for OCM, 8/4 for DFVS, 10/5 for Cluster Editing, and 6/3 for Treedepth. The family lists, in generator order, are:

- OCM: `banded`, `blocks`, `interval`, `bimodal`. Small/medium bipartite sizes are 22/14 and 72/46. The certificate is a permutation of the free side; the trusted checker recomputes its crossing count.
- DFVS: `planted_scc`, `cycle_cover`, `sparse`, `dense`. Small instances have 16–24 vertices and medium instances 56–84. The certificate is a deletion set; the checker verifies acyclicity and recomputes deleted weight.
- Cluster Editing: `balanced_iid`, `imbalanced_iid`, `ring_bridges`, `merge_split_traps`, `hub_corruption`. Small/medium sizes are 28/72 vertices. The certificate is a partition; its edge-edit cost is recomputed externally.
- Treedepth: `path_power`, `closure_tree`, `random_connected`. Small instances have 10–14 vertices and medium instances 28–42. The checker validates an elimination forest and edge-ancestor constraints, then measures depth.

For development, instance seed equals its base plus $100i + j$, where $i = 0, 1$ denotes small/medium and j is the zero-based family index. Evaluation enumerates medium as its sole scale, with seed equal to its evaluation base plus j. The task-level search seed is the search base plus the task index in the order OCM, DFVS, Cluster Editing, Treedepth. Each block specifies its own frozen data and search seed, so variation includes both instance and search differences.

Table 6: Frozen seed bases for the three data blocks. Each data group uses separate seeds for search, development instances, and evaluation instances.

Block	Search	Development	Evaluation
02	202622190	2026100100	2026100900
04	202623390	2026112000	2026113000
05	202623490	2026114000	2026115000

G.2 Baseline Implementations and Configuration

All candidate programs implement `solve(instance)`. An external checker verifies the returned solution and recomputes its objective. Every method uses the same checker, initial and reference algorithms, and execution limits. Each task/group permits at most 100 API calls, including failed requests and model calls used for rollback. Methods retain their own stopping rules within this common cap.

We use the following baseline code revisions:

- AdaEvolve, `https://github.com/skydiscover-ai/skydiscover`: `700711ec4c37d0be677456f66d38461310c0bcf8`.
- OpenEvolve, `https://github.com/algorithmicsuperintelligence/openevolve`: `411fb59c886c18704caaffb611e17cf9e7d824d2`.
- AI Scientist V1, `https://github.com/SakanaAI/AI-Scientist`: `1de1dbc1f4ee2c5f61e9c94348d55eb51d7fa2eb`.
- AI Scientist V2, `https://github.com/SakanaAI/AI-Scientist-v2`: `96bd51617cfdbb494a9fc283af00fe090edfae48`.

AdaEvolve and OpenEvolve request up to 99 evolution iterations with a 100-request cap. V1 runs its `perform_experiments` stage through Aider, with five experiment runs and up to 21 candidate attempts in the recorded configuration. V2 runs 19 initial-stage tree-search steps and its final candidate-selection rule. The evaluated V2 workflow ends after candidate selection. API meters account for search steps that use multiple model calls.

All methods use the recorded `deepseek-flash` model identifier and the common 65,000/6,000 prompt/output ceilings. EVIDRA, AdaEvolve, and OpenEvolve use temperature 0.7; AI Scientist V1/V2 use their respective stage-specific decoding settings. The execution environment is Linux x86-64 with Python 3.11.15. Candidate execution uses bubblewrap isolation, a cleared environment, a read-only job directory, a 10-second CPU limit per instance, and 512 MiB address-space limit. The budget meter also enforces the API-call limit and reserved cost budget. A run can end early when the next prompt exceeds its input limit or the method reaches its stopping condition.

Failed, discarded, and rollback calls count toward the common limit for each task/group; methods may stop before reaching it.

G.3 Experimental Reproduction Procedure

Generate the development and evaluation instances using the families, sizes, and seed rules listed above. Run each method with the common execution limits and its specified configuration. For EVIDRA, follow the controller and search procedure in Appendix A.1; apply the interventions in Appendix F for the two ablation variants.

After search, fix each method's selected candidate and evaluate it on the corresponding evaluation suite. Verify each returned solution, recompute its objective, and aggregate the unrounded instance scores using Appendix E. Search-progress curves and candidate-outcome statistics use the definitions in Appendix H.

G.4 Mathematical Result Index

The three studies have different comparison objects and certificates:

- Lonely Runner translation monotonicity: the primitive triple $(1, 20, 22)$ gives an exact drop $82/525$ under a uniform $+1$ shift. Compared with the reference triple $(7, 18, 26)$, whose drop is $65/506$, this is a 21.59% improvement and reaches 98.29% of the exhaustive finite-domain maximum $133/837$ on the specified triples. Appendix B gives the exact breakpoint enumeration and curriculum protocol for this translation-monotonicity result.
- Erdős–Straus sieve weight: the certificate establishes $W(J) \geq 0.450122027445(\log J)^2$ for $J \geq 2^{1246}$, compared with the target coefficient 0.45. Appendix C supplies the finite-prime and harmonic-block derivation and conservative arithmetic for the improved sieve-weight certificate.
- Four-variable iid probability: the certified upper bound $6073/14630 \approx 0.4151059467$ improves the rigorous reference $673/1615 \approx 0.4167182663$ by $401/248710$, or approximately 0.3869%. The smaller published numerical value $20898/50600$ remains unproved. Appendix D derives the rational packing certificate and its integer-strengthened bound.

These results rely on exact-arithmetic witnesses and certificates.

H Search Progress and Candidate Outcomes

H.1 Search Progress

We analyze all 24 ablation runs across both variants and three data groups, with 100 API calls per run. Logs link solver code to scores, excluding initialization and preliminary checks outside the call budget.

For task t, block b, and API call k, let z_{tbk} be the best verified development score after that call. A failed request or rejected output without evaluation retains the preceding best score; the initial solver has score zero. Define the normalized discrete area

$$A_{tb} = \frac{1}{100} \sum_{k=1}^{100} z_{tbk}. \tag{33}$$

We first average A_{tb} over the four tasks within each block, then report the mean and sample SD over three blocks. This area summarizes progress on the development suites as the API-call budget accumulates.

The four-task mean areas are 0.734266 ± 0.082834 for w/o Process Adaptation and 0.606532 ± 0.053800 for w/o Propose. The latter has a lower mean area on every task, consistent with its weaker final scores. These curves compare the two reported variants under the same request cap.

H.2 Valid Programs and Rejections

A mutation is a newly evaluated candidate, excluding rollback checks and the initial solver. It is valid when every development instance passes. We count distinct valid programs by abstract syntax tree (AST), ignoring source positions and summing the count within each run. Repeated programs can therefore yield more valid mutations than distinct programs. AST counts distinguish syntactically different candidate implementations.

Table 7: Candidate outcomes across 12 runs per ablation (four tasks $\times$ three blocks), with 1,200 requests each. "Distinct" counts valid programs with different ASTs within each run, summed over runs. Invalid outputs did not produce an evaluated candidate.

Arm	Evaluated	Valid	Distinct	Invalid program	Invalid output
w/o Process Adaptation	843	597	580	246	309
w/o Propose	992	762	754	230	160

w/o Process Adaptation uses 47 rollback calls and has one transport failure; w/o Propose uses 45 and has three failures. They improve the best score 47 and 45 times, respectively. Together with Table 7, these counts cover all 1,200 calls per ablation. w/o Propose yields more valid programs but lower development area and final evaluation scores.

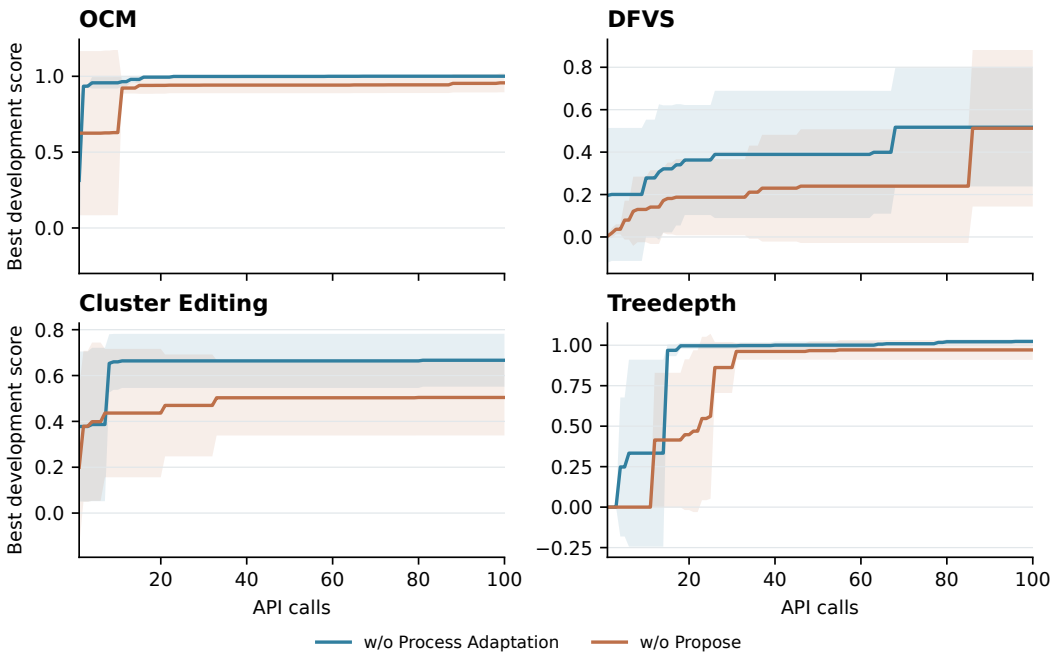

Figure 5: Best development score against API calls for w/o Process Adaptation and w/o Propose. Lines average the three data blocks and shading shows their sample SD; both instances and generated programs vary across blocks.

H.3 An Improvement from an Alternative Parent

For the w/o Process Adaptation ablation, we examine the first verified gain from a parent below the incumbent, ordering runs by task (OCM, DFVS, Cluster Editing, Treedepth), data group (02, 04, 05), and API call.

On OCM, group 02, call 7, the initial barycenter solver (score 0) is the parent, while the incumbent scores 0.942510638. Deterministic initial orders, local search, and perturbations together yield 0.942742764. Rollback restores score 0, verifying the combined edit's contribution.