

# The Incumbent Did the Work: What the Simple-Baseline Results in LLM-Guided Program Search Settle, Why the Dispute Over Archives, Diversity and Test-Time Training Is Fought on Benchmarks That Cannot Rank It, and the Comparisons That Would

Pranay Mahendrakar

SONYTECH · ORCID 0009-0003-7224-029X

pranaymahendrakar@sonytech.in · mahendrakarpranay@gmail.com · +91 6361723454

**Abstract.** Systems that let a language model write programs, score them with an exact verifier and feed the best ones back have improved published bounds on open mathematical problems, and each new system has added machinery: program archives, islands, novelty filters, model ensembles, and since 2025, weight updates at test time. A run of 2026 papers argues that most of this machinery is unnecessary. The sharpest, Hill Sampling, keeps only the best verified program, samples hundreds of edits of it per round from a frozen open-weight model, and reports a circle-packing score above every published method, an Erdos minimum-overlap bound below AlphaEvolve's, and evolution strategies on the weights losing to plain sampling. In this analysis the author reviews and synthesises published results to show that this literature answers two different questions under one heading. The first, whether carrying the best program forward beats sampling afresh from the start, is close to settled in favour of carrying it forward, and the procedure that wins was already a named baseline in 2024. The second, whether anything beyond that incumbent pays, is not settled. The evidence on each side was gathered under different settings of six variables: the unit and size of the budget, the benchmark's headroom relative to its verifier's tolerance, the information placed in the prompt and seed, the base model, the statistic and number of runs, and how many accepted improvements a record requires. On the most cited benchmark, at least five results now agree to six decimals on a sum of radii of 2.635983, the three reported more finely lie within about  $2.5e-7$  of one another, and one 2026 record was set partly by enlarging radii inside the verifier's tolerance. Twenty-seven published results are gathered in a single table, a seven-step procedure for crediting a reported gain to a harness component is given, and the paper sets out the comparisons that would reveal what the machinery adds: problems with measured headroom, and budgets reported in several units at once.

**Keywords:** LLM-guided program search; evolutionary search; test-time compute; AlphaEvolve; hill climbing; test-time training; evaluation methodology; mathematical discovery

SONYTECH ORCID: 0009-0003-7224-029X

## 1. Introduction

A language model can be made into a search operator. Give it a program, ask it to change the program, run the result through an exact checker, and keep what scores well. Lehman et al. (2022) proposed code-trained models as mutation operators for genetic programming and paired them with MAP-Elites to generate working programs in a domain the model had never seen. FunSearch showed that this loop could find constructions that improved on known results in extremal combinatorics (Romera-Paredes et al., 2024). AlphaEvolve extended it from a single function to whole files, ran it with an ensemble of Gemini models, and reported improved bounds on a list of open problems, among them packing 26 circles in a unit square (from 2.634 to 2.635 in sum of radii) and Erdos's minimum-overlap problem (from 0.380927 to 0.380924) (Novikov et al., 2025). A companion study applied the same system to 67 problems and found it matched the best known constructions on most and improved several (Georgiev et al., 2025). Open-source systems followed,

each with more parts. ShinkaEvolve added fitness-weighted parent selection over island subpopulations, rejection of near-duplicate code by embedding similarity, and a bandit over a pool of models (Lange et al., 2025). CodeEvolve combined island search, a MAP-Elites archive, crossover by inspiration and meta-prompting (Assumpcao et al., 2025). ThetaEvolve and TTT-Discover went further and updated the model's weights by reinforcement learning during the search (Wang et al., 2025; Yuksekgonul et al., 2026).

Against this growth, 2026 brought papers whose message is subtraction. Gideoni et al. (2026) compared two simple baselines with ShinkaEvolve on nine of AlphaEvolve's problems, giving every method models from the same Gemini 2.5 family, the same minimal prompts and a 20-dollar budget, and found that the sequential baseline matched or beat the complex pipeline on six of the nine. Gupta et al. (2026) rebuilt OpenEvolve-style and TTT-Discover-style harnesses from a greedy root one component at a time, evaluated 30 harnesses over 12 model-problem pairs with more than 3.1 million rollouts, and found that no fixed harness beat the greedy root significantly once multiple comparisons

were corrected for. Most recently Beck et al. (2026) introduced Hill Sampling: keep the best verified program, draw many edits of it from a frozen model each round, and replace it whenever an edit scores at least as well. With three open-weight models of 20 to 32 billion parameters and eight H100 GPUs, they report a circle-packing construction scoring 2.635983084917604, which they place above every published method under a zero-slack verifier, an Erdos bound of 0.38089767, below AlphaEvolve's, and a series of failures for added machinery: evolution strategies on the weights, archives, diversity selection, execution feedback, entropy control. Their abstract's recommendation is to sample edits of the best verified solution before adding any further complexity.

Read together, these papers seem to say that the field built machinery it did not need. This paper accepts nearly every number in them and disputes that reading. Their findings divide into answers to two questions that are usually asked as one.

The first question is whether a search that carries forward its best program does better than one that samples afresh from the starting program each time. On this the literature agrees, across five independent groups and with a few exceptions that can be located: carrying the incumbent forward wins. AlphaEvolve's own ablation is the reverse image of Hill Sampling, because its "no evolution" arm is exactly repeated sampling from the initial program (Novikov et al., 2025). The 2024 PPSN benchmark of Zhang et al. (2024) introduced a (1+1) hill-climbing baseline under the name (1+1)-EPS and showed it beating a standalone model given an order of magnitude more queries. The greedy root of Gupta et al. (2026), which they call Sequential Best-of-N, is Hill Sampling under another name. Hill Sampling's contribution is therefore not the procedure but a demonstration that the procedure, run with a large number of samples per round, reaches published-best scores with open models in hours.

The second question is whether anything beyond the incumbent pays: populations, archives, islands, diversity filters, model ensembles, weight updates. On this question the literature does not agree, and the disagreement is not noise. The results that favour machinery and the results that favour its removal were obtained under different settings of a small number of variables that no study varies jointly. The budget is counted in evaluations by one study, in completions by another, in dollars by a third, and in wall-clock hours by a fourth, and the number of programs sampled per round ranges from one to 512. The flagship benchmark, 26 circles in a square, has stopped discriminating: at least five independent results now agree to six decimal places, the three reported more finely lie within about  $2.5 \times 10^{-7}$  of one another, verifier tolerances in use range from  $1 \times 10^{-10}$  to  $1 \times 10^{-6}$ , and one 2026 paper states that its record was obtained partly by enlarging radii within its evaluator's tolerance (Chen et al., 2026). The prompt and seed carry domain knowledge whose effect can be as large as the

harness's: the same complex pipeline found the best-known packing in three of three runs with its default prompt and one of three with a minimal one (Gideon et al., 2026). Where an effect has been measured on more than one base model, its size has moved with the model, and that includes the benefit of test-time training. The number of runs per arm is three, five or one, and the statistic is a maximum over runs, a mean over runs, or a mean over the top 50 programs. Finally, the strongest evidence for machinery comes from problems whose records are built up over many steps, while the problems behind the simplification results are harder to place: on the anchor's own problems, configured run length and the round in which the record appeared rank them in opposite orders.

The thesis of this paper is this. The simple-baseline literature has established that the incumbent is the load-bearing component of LLM-guided program search, and it has not established that the remaining components are unnecessary. It has shown that their value is not visible on the benchmarks, budgets and statistics now in use. Those are different findings with different consequences. The first licenses a default, which Hill Sampling's abstract states well: start with the incumbent. The second licenses only a measurement programme, because a component whose benefit cannot be seen on a saturated benchmark with three seeds may still be worth having on an unsaturated one with thirty.

The paper is an analysis. In this analysis the author reviews and synthesises published results to show that the verdicts on search machinery rest on unmatched settings, and to specify the comparisons that would decide between them. Each number quoted below names the study that reported it, and any arithmetic the paper performs on reported values is marked as the author's. The paper's own contributions are the separation of the two questions (Section 3), the six-variable account of why the second is unsettled (Section 5), the scoping of the test-time training dispute (Section 6), the consolidated evidence in Table 1 and the design comparison in Table 2, the attribution procedure in Algorithm 1, the headroom display in Figure 1, and the matched comparisons in Section 10.

Section 2 sets out what Hill Sampling did and what its design holds fixed. Section 3 separates the two questions and traces the incumbent baseline's lineage. Section 4 collects the evidence on the first question. Section 5 takes the six variables on which the second question turns. Section 6 examines test-time weight learning, and Section 7 the narrower contrast between noise in weights and noise in tokens. Section 8 sets the dispute beside earlier ones in neural architecture search, reinforcement learning and generative modelling. Section 9 consolidates the evidence, Section 10 states the comparisons that would settle the open part, and Sections 11 to 13 put the best objection to the argument, list open unknowns, and note limits of the method.

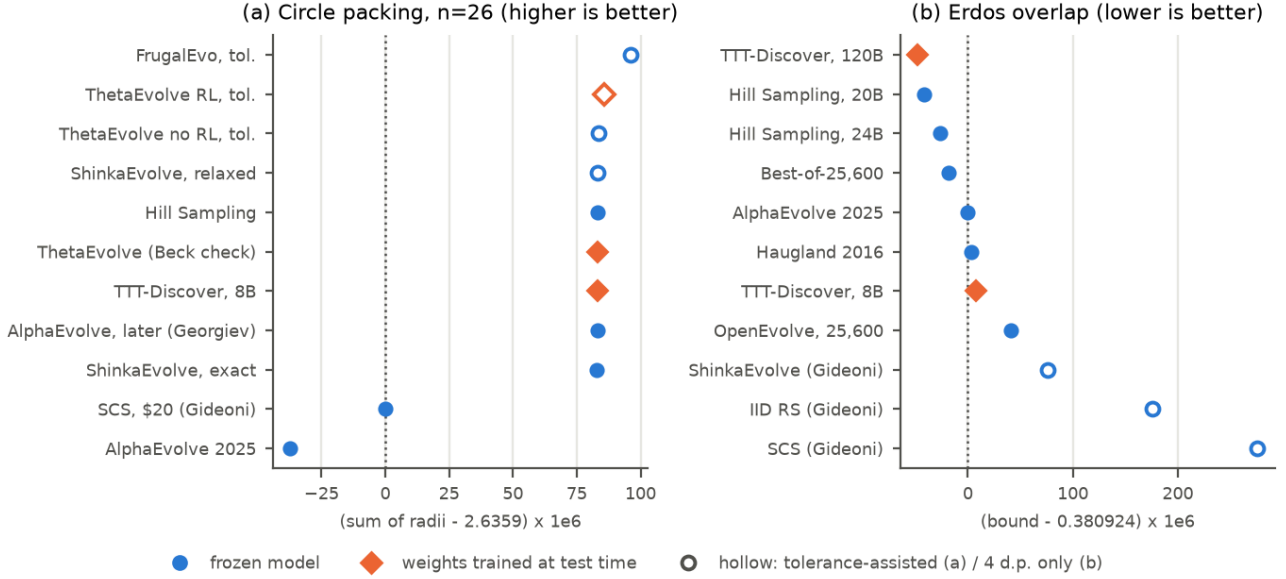


**Figure 1.** Reported results on the two benchmarks used most in the machinery dispute. (a) Circle packing with 26 circles in a unit square: sum of radii minus 2.6359, times 1e6, as reported by Novikov et al. (2025, value as given by Beck et al., 2026), Georgiev et al. (2025, value as given by Yuksekgonul et al., 2026), Gideon et al. (2026), Lange et al. (2025), Yuksekgonul et al. (2026), Beck et al. (2026), Wang et al. (2025) and Chen et al. (2026); hollow markers were obtained under a verifier tolerance or slack. (b) Erdos minimum-overlap upper bound minus AlphaEvolve's 0.380924, times 1e6, as reported by Haugland (2016), Novikov et al. (2025), Yuksekgonul et al. (2026), Beck et al. (2026) and Gideon et al. (2026); hollow markers are reported to four decimals only, so their position is uncertain by up to 50 units. Orange diamonds mark runs that updated model weights at test time.

## 2. What Hill Sampling did and what its design holds fixed

Hill Sampling is defined in a few lines (Beck et al., 2026, Section 3). A round draws  $N$  edits of the current incumbent program from a frozen model at a fixed temperature, runs each once, and replaces the incumbent with the best edit if that edit scores at least as well as the incumbent's stored score. The incumbent is never re-run, so the stored best can only rise. There is no archive, no crossover, no diversity objective and no parameter update. Two settings are reported,  $N = 64$  and  $N = 512$ , with the same total number of completions, so the second runs one eighth as many rounds. Inference runs on eight vLLM instances, one per H100.

The evaluation covers three problems, each paired with one model to keep compute manageable: circle packing with gpt-oss-20b, sums and differences of finite sets with OLMo-3.1-32B-Instruct, and Erdos's minimum-overlap problem with Mistral-Small-3.1-24B-Instruct. Each method is run with three seeds and a manually tuned temperature, and the authors state that compute is matched by the number of model completions. The verifiers come from OpenEvolve's open implementation of AlphaEvolve's tasks (Sharma's repository, as the anchor cites it), hardened with value and type checks after reward hacking was observed on the sets problem. The circle-packing verifier has a tolerance of  $1e-10$  in its containment and overlap checks, and the headline packing is reported without slack.

The best results appear in the anchor's Table 1. On circles the best run scored 2.635983084917604 in 4 hours 33 minutes, found in round 14. The anchor lists

ThetaEvolve's 2.6359830774 as the best previously published zero-slack value and AlphaEvolve's as 2.6358627564136983. On Erdos the best run scored 0.38089767 in 12 hours, found in round 70, against AlphaEvolve's 0.38092303510845016 and TTT-Discover's 0.380876, which remains better. On sets the best run reached 1.109543 against AlphaEvolve's 1.158417281556896, in 1 hour 13 minutes, found in round 2. The caption gives completions as 512 times the round number, which on the author's arithmetic is 7,168 completions for the circle result, 35,840 for Erdos and 1,024 for sets. A footnote adds that switching Erdos to gpt-oss-20b improved the bound to 0.38088232, and a separate sets run with no initial code, a longer timeout and a prompt note that the best known solution is sparse reached 99.02 percent of AlphaEvolve's score; that run is excluded from the table because of the added domain knowledge.

The machinery comparisons are of three kinds. Against repeated sampling, defined as drawing edits of the original program every round without carrying improvements forward, Hill Sampling is the strongest method on circles and on Erdos, and on sets no method differs significantly from any other (anchor, Figure 1). Against weight-space methods, evolution strategies with a nonzero learning rate raise the final mean return on circles but leave the maximum return best at learning rate zero, which reduces the method to evaluating random Gaussian perturbations of the weights; ordinary token sampling from the unperturbed model beats those perturbations in turn (anchor, Section 5.1). Against search machinery, on Erdos only, the anchor tests top-K parents, top-K with random or embedding-diverse

additions, multiple programs in context, 64 subpopulations, a four-step in-context RL history, execution feedback, adaptive temperature, long-horizon ES and hill climbing in weight space. It reports that none conveys a significant advantage and that all except ES and adaptive temperature added on top of Hill Sampling reduce performance (anchor, Section 5.3).

Six features of this design matter for what the results can show.

First, the budget is counted in completions, and only completions. A completion that runs to 8,000 tokens and one that stops at 800 count the same, and the anchor states plainly that its goal is wall-clock practicality, not sample efficiency in evaluations or completions. That is a legitimate choice. It also means that a method which spends more tokens per completion, such as one that places several programs in context, is charged nothing for them, while a method that needs fewer completions to reach a score earns no credit for it.

Second, each problem is paired with a single model, so no result separates a property of the method from a property of the model. The sets result, where every method reached about 95 percent of AlphaEvolve's score, is one the anchor itself reads as "suggestive of a model or domain knowledge failure" (Beck et al., 2026), and the 99.02 percent run with a different model and a sparsity hint supports that reading.

Third, each arm has three seeds, the figures compare means with standard errors, and the headline table reports the best seed. Section 5.5 returns to what three seeds can resolve.

Fourth, the machinery tests were run on Erdos because, in the anchor's words, "Erdos requires the fewest rounds of improvement" (Beck et al., 2026, Section 5.3). The choice was made for cost. The phrase fits the configured run lengths, 80 rounds at  $N = 64$  for Erdos against 200 for circles and 600 for sets, but not the rounds in which the records were found: round 70 for Erdos, 14 for circles and 2 for sets (anchor, Table 1). The two measures point opposite ways, so it cannot be said whether the broadest evidence against archives and diversity comes from a problem on which a population has little time to matter or from the one, of the three, whose record took the longest to build.

Fifth, the circle-packing comparison sits at the edge of numerical resolution. The anchor's own footnote reports that two web-reported results with undocumented methods fail its verifier, that after removing slack one scores lower and one higher than its own, and that at this precision validity can turn on floating-point details such as the order of operations. Section 5.2 shows that this is not a footnote-sized problem.

Sixth, the evolution-strategies arm is evolution strategies. Its update is the scalable antithetic estimator of Salimans et al. (2017), with rewards standardised across the population, applied to all parameters at a learning rate of  $1e-7$  with a noise scale of  $3e-4$ . It is not policy-gradient reinforcement learning, and none of its variants

uses an entropic policy-gradient objective of the kind TTT-Discover credits; its max@8 and softmax variants are cruder max-seeking forms (Section 6). Section 6 takes this up.

None of this makes the anchor's results wrong. It locates them. The next section separates what they establish from what they are read as establishing.

### 3. Two questions under one name

#### 3.1 State versus machinery

Every LLM-guided program search can be described by three choices: what state it keeps between model calls, how it picks the program or programs that condition the next call, and whether the model itself changes. Repeated sampling keeps no state that conditions generation; each call sees the same starting program, and the best output is selected at the end (Brown et al., 2024). Hill Sampling keeps one program, the incumbent, and conditions every call on it. Everything else in the field keeps more: a top-K set, a quality-diversity map, island subpopulations, a tree of programs with visit counts, a history of attempts in context, or a set of updated weights.

The first question is whether the incumbent beats no state. Call it the state question. The second is whether anything beyond the incumbent beats the incumbent. Call it the machinery question. These are distinct because the first comparison changes what the model sees on every call, while the second changes only which of several good programs it sees, or how the model was adjusted. A result on one says nothing about the other. A paper that shows its full system beating repeated sampling has answered the state question and is silent on the machinery question, and a paper that shows the incumbent beating repeated sampling and also matching the full system has answered both, but only for its budget, model and problems.

The distinction is old in evolutionary computation. Hill Sampling, with  $N$  edits per round and elitist replacement, is what the evolution-strategies literature writes as a  $(1 + \lambda)$  strategy: one parent,  $\lambda$  offspring, the best of parent and offspring survives (Beyer and Schwefel, 2002). How large  $\lambda$  should be is a classical question. Jansen et al. (2005) analysed it for a simplified evolutionary algorithm and concluded that the offspring population size can change the efficiency of search a great deal, and suggested a simple way of adapting it during the run when necessary. The opposite position, that pure fitness-following is the problem, also has a classical statement. Lehman and Stanley (2011) argued that objective functions can deceive a search into dead ends and reported that novelty search, which ignores the objective, outperformed objective-based search on deceptive maze and biped tasks. MAP-Elites, which keeps the best solution in each cell of a user-chosen map of behaviours, was introduced with the claim that exploring more of the space also tends to find a better single solution (Mouret

and Clune, 2015). The machinery question is the LLM-era version of an argument that predates LLMs, and the classical answer was that it depends on the landscape, which is what the no-free-lunch theorems would lead one to expect (Wolpert and Macready, 1997).

### 3.2 The incumbent baseline has a history

Hill Sampling is described by its authors as new, and as a procedure it is not. OPRO conditioned each call on previously generated solutions and their scores in the prompt (Yang et al., 2024). Zhang et al. (2024) proposed (1+1)-EPS, a single-parent, single-offspring evolutionary program search modelled on the (1+1) evolution strategy, explicitly as the simplest adequate baseline for LLM-based heuristic design. Its role was to stand for the lower bound of what program search could do. AlphaEvolve's "no evolution" ablation removed the evolving state and fed the model the same initial program each time (Novikov et al., 2025). ShinkaEvolve's parent-selection ablation included a hill-climbing arm that always mutated the best program (Lange et al., 2025). ThetaEvolve simplified AlphaEvolve to a single model and, optionally, only the parent program in the prompt (Wang et al., 2025). Gupta et al. (2026) took greedy Sequential Best-of-N, which repeatedly mutates the best program found so far and generates N children per iteration, as the root from which every other harness was derived.

This lineage matters for interpretation in two ways. It means that the comparison between the incumbent and the full machinery had been run several times before 2026, with mixed results that Section 5 examines. And it means that Hill Sampling's characterisation of its nearest neighbours needs care. The anchor says of Gupta et al. (2026) that they advocate a complex adaptive allocation across harnesses and never match AlphaEvolve's scores. The first is true and the second is true, but Gupta et al.'s baseline is the anchor's own procedure, their central finding is that no fixed harness beats it significantly, and their budgets were 160 to 1,600 rollouts per run, below the anchor's 7,168 completions for its circle result. Their failure to reach AlphaEvolve's scores is at least as plausibly explained by their budgets and models as by their harnesses. The anchor's summary of Gideoni et al. (2026) is also strict: it says those baselines do not consistently surpass both AlphaEvolve and ShinkaEvolve on any domain. The source's own table has its sequential baseline matching or exceeding ShinkaEvolve on six of nine problems and AlphaEvolve on four of nine, under a budget its authors judge to be likely much lower than AlphaEvolve's.

## 4. The state question is close to settled

Table 1 in Section 9 collects the results; the account here is of their direction.

AlphaEvolve's ablations, on tensor decompositions for matrix multiplication and on kissing-number lower bounds, report that removing evolution, removing problem context from prompts, removing meta-prompt

evolution, evolving only a loss function instead of the full file, and using a single small model each reduce the results, and the authors call each component responsible for a significant improvement (Novikov et al., 2025, Section 4). The "no evolution" arm is the state question. The other arms are mixtures of machinery and information, a point Section 5.3 returns to. In its discussion the AlphaEvolve paper describes its evolutionary procedure as significantly enhancing the base model's capability compared with, for example, repeated sampling.

Zhang et al. (2024) ran four LLM-based program-search methods, FunSearch, EoH (Liu et al., 2024), ReEvo (Ye et al., 2024) and their own baseline, over four heuristic-design problems, nine models and five independent runs. They found that their (1+1)-EPS baseline produced better mean performance with much lower variance than a standalone model given an order of magnitude more queries, and that a small code model with (1+1)-EPS beat standalone sampling from far larger models. That is a direct answer to the state question, and on three of their four problems it is yes.

ShinkaEvolve's ablation on circle packing ranks its Best-of-N arm, which always uses the initial program as parent, as the poorest of its parent-selection strategies (Lange et al., 2025, Section 5). Hill Sampling's main figure has repeated sampling below Hill Sampling on circles and Erdos (Beck et al., 2026, Figure 1). TTT-Discover's kernel-engineering ablation has Best-of-N, with no reuse and no training, at 5,352.36 microseconds against 2,060.70 for the same search with reuse but no training (Yuksekgonul et al., 2026, Table 8).

There are four exceptions, and each can be located. In Zhang et al.'s own appendix, on the OBP (OR) bin-packing problem, "the LLM-based EPS methods do not demonstrate a significant advantage" over the standalone model, and the same holds for the best of five runs (Zhang et al., 2024). On the sets problem, Hill Sampling and repeated sampling do not differ significantly, but every method stalled near 95 percent of AlphaEvolve's score, which the anchor reads as a model or domain-knowledge failure: when the search space excludes the answer, keeping state cannot help. In Gideoni et al. (2026), the IID random-sampling baseline matched or beat ShinkaEvolve on four of nine problems and the sequential baseline on six. The sequential baseline keeps state of a weaker kind than an incumbent: it conditions on a random subset of the previous generation's programs that ran, with no selection by score. The advantage of keeping state is visible in that comparison but not uniform, and on ALE-Bench LITE both baselines beat ShinkaEvolve on eight of ten tasks. In TTT-Discover's mathematics table, Best-of-25,600 sampling from gpt-oss-120b reached an Erdos bound of 0.380906, better than AlphaEvolve's 0.380924 and better than OpenEvolve's 0.380965 at the same budget (Yuksekgonul et al., 2026, Table 2). That last result needs its caveat: the authors report that many OpenEvolve rollouts were truncated because OpenEvolve's prompts outgrew the

32,768-token context window, and they kept OpenEvolve's prompts unchanged. The arm that kept state was handicapped by the state it kept.

Taken together, the evidence supports a scoped conclusion. When the answer lies inside the search space the verifier exposes, and when the model's context can hold the state, conditioning on the best program found so far beats sampling afresh from the start, often by a wide margin. The Large Language Monkeys result that coverage grows roughly log-linearly with repeated samples (Brown et al., 2024) does not contradict this; it describes how far resampling alone goes, and the program-search studies show that keeping state goes further at the same budget. Stroebel et al. (2024) add the boundary condition: when the verifier admits false positives, resampling has a ceiling no budget removes, and the program-search papers that report reward hacking or leaky verifiers (Beck et al., 2026; Georgiev et al., 2025) describe verifiers with exactly the false positives under which that ceiling applies.

## 5. The machinery question turns on six variables

The results on the machinery question point both ways. ShinkaEvolve's ablations favour weighted parent sampling over hill climbing, bandit model ensembling over a fixed ensemble, and novelty rejection over none (Lange et al., 2025). ThetaEvolve reports that a larger program database improves final performance as test-time compute grows (Wang et al., 2025). CodeEvolve reports that under matched conditions it beats OpenEvolve and ShinkaEvolve on six of nine AlphaEvolve problems, and that its ablations attribute the gains to interactions between components rather than to any single operator (Assumpcao et al., 2025). EvoX, which evolves its own search strategy, reports outperforming AlphaEvolve, OpenEvolve, GEPA and ShinkaEvolve on most of nearly 200 tasks (Liu et al., 2026). PACEvolve identifies context pollution and mode collapse as failure modes of agents anchored to their current hypotheses and adds hierarchical context management and backtracking to counter them (Yan et al., 2026a). Against these stand Hill Sampling's Erdos battery (Beck et al., 2026), Gideoni et al.'s nine-problem comparison, and Gupta et al.'s finding that no fixed harness beats the greedy root significantly after Holm correction (Gupta et al., 2026).

This paper argues that the two sets of results do not contradict each other as much as they appear to, because they were obtained under different settings of six variables. This section takes them in turn.

### 5.1 The budget, its unit, and how it is split

A budget for program search can be counted in at least five units: verifier evaluations, model completions, tokens, dollars and wall-clock hours. The studies in this dispute each pick one, and the pick favours some designs over others.

ShinkaEvolve's headline is sample efficiency in evaluations: a new best circle packing within 150 program evaluations (Lange et al., 2025). Its hill-climbing ablation, which it reports as strong at first but plateauing quickly, was run in a design that, as Gideoni et al. (2026) describe it, evaluates one program per generation. Hill Sampling counts completions and spends 64 or 512 per round, and its circle record came after 7,168 completions on the author's arithmetic from the anchor's caption. Gupta et al. (2026) count rollouts, at 1,600 per run for 3- and 4-billion-parameter Qwen models, 320 for GPT-OSS-20B and 160 for GPT-OSS-120B. Gideoni et al. (2026) count dollars, 20 per problem per method, and note that ShinkaEvolve ran for 500 to 800 generations within that sum. TTT-Discover counts rollouts, 25,600 per run, and estimates a run at about 500 dollars (Yuksekgonul et al., 2026). FrugalEvo argues that the right objective is gain per unit cost and introduces an area-under-the-curve measure over cumulative model spend, reporting circle-packing results for 1.68 and 0.55 dollars (Chen et al., 2026). Gideoni et al. also report wall-clock: their baselines needed one to three hours per problem against about ten for ShinkaEvolve, because independent samples parallelise across processors and sequential generations do not.

These units are not interchangeable, and the conversion between them is design-dependent. A method that places several programs in context spends more tokens per completion than one that places a single incumbent; under a completion budget the extra context is free, under a dollar budget it is not. A method that generates many children per round converts budget into wall-clock speed through parallel inference, which a sequential evaluator-limited design cannot do. ShinkaEvolve rejects near-duplicate programs before they are evaluated so as not to waste computation on them (Lange et al., 2025); under a completion count that saving is invisible. Gideoni et al. give a candidate reason for ShinkaEvolve's weak showing on ALE-Bench in exactly these terms: diff-based edits may be inefficient under a finite sample budget.

The split of a fixed budget into breadth and depth is a separate variable, and it moves results. Gupta et al. (2026) report that shifting a fixed rollout budget from breadth to depth, as OpenEvolve does, was among the changes that did not help and sometimes reversed gains from simpler variants. Hill Sampling's two settings,  $N = 64$  and  $N = 512$  at equal completions, are both broad by the standard of ShinkaEvolve's one child per generation. The author conjectures a reading consistent with all three studies: a greedy incumbent needs breadth. With many edits per round it may sample the incumbent's neighbourhood widely enough to substitute for a population, and with one edit per round it plateaus, as ShinkaEvolve's ablation reports. If that reading is right, ShinkaEvolve's verdict against hill climbing and Hill Sampling's verdict for it are both correct at their own breadth, and neither transfers to the other's.

Snell et al. (2024) reached a related conclusion for reasoning tasks before any of these papers: how well a given way of spending test-time compute works varied with the difficulty of the prompt, and allocating compute adaptively per prompt was more than four times as efficient as a best-of-N baseline. Gupta et al.'s adaptive allocation, which starts several harnesses, prunes weak runs at checkpoints and gives the remaining budget to survivors, raised the average normalised score from 84.35 percent for five greedy runs to 85.75 percent (Gupta et al., 2026, Table 1). That is the same lesson at the level of harnesses.

## 5.2 Headroom against the verifier's tolerance

A benchmark can rank search methods only if the methods' results differ by more than the resolution of the measurement. For 26 circles in a unit square this condition has stopped holding.

Figure 1 shows the reported values. Panel (a) plots each reported sum of radii minus 2.6359, in units of  $1e-6$ . AlphaEvolve's 2025 construction, as given to 16 digits by Beck et al. (2026), sits at about  $-37$ . At least five later results agree to the six decimals they share, at 2.635983. Three are reported to more digits: ShinkaEvolve after its relaxed solution was made exact (2.6359828390), ThetaEvolve (2.6359830774 in the anchor's zero-slack check) and Hill Sampling (2.6359830849), and these lie within about 0.25 of a unit, that is within about  $2.5e-7$ , of one another (the author's arithmetic on the reported values). The other two, TTT-Discover with Qwen3-8B and AlphaEvolve's later round reported by Georgiev et al. (2025), are given as 2.635983 in Yuksekgonul et al.'s table, so their position inside that band cannot be read; FrugalEvo's table adds best runs of 2.635983 for OpenEvolve and EvoX with GPT-5.6 Terra, not plotted (Chen et al., 2026, Table 1). Above them sit values obtained with tolerance: ShinkaEvolve's relaxed solution, 2.635983099, made exact by shrinking each radius by  $1e-8$  (Lange et al., 2025); ThetaEvolve's best runs, 2.6359857 with RL and 2.6359834 without, under an evaluator with a  $1e-6$  tolerance (Wang et al., 2025, Table 2); and FrugalEvo's 2.635996, which its authors describe as found partly by applying small radius increases within the evaluator's tolerance (Chen et al., 2026).

The tolerances in use span four orders of magnitude:  $1e-10$  in Hill Sampling's hardened verifier, a shrink of  $1e-8$  per circle in ShinkaEvolve's exact conversion,  $1e-6$  in the OpenEvolve-derived evaluator ThetaEvolve used. The spread among the zero-slack results, about  $2.5e-7$ , is smaller than the largest of these tolerances and larger than the smallest. A claim that one method beats another on this benchmark is therefore a claim about the verifier as much as about the method. Hill Sampling's own footnote acknowledges the problem: of two web-reported results it re-verified, one scored above its record once slack was removed, and the anchor notes that at this precision validity can depend on floating-point details. Its claim of a new state of the art rests on a margin of about  $7.5e-9$  over ThetaEvolve (the author's arithmetic on the anchor's Table 1), well below the  $1e-6$  tolerance

of the evaluator ThetaEvolve's own runs used.

The benchmark also appears no longer to reward the capability the machinery was built to supply. The evolved programs that reach the cluster are numerical optimisers. Hill Sampling's best program seeds a core-and-shell layout and refines it with simulated annealing and SLSQP (Beck et al., 2026, Appendix E). TTT-Discover's initialises a staggered or hexagonal grid and refines with sequential least squares programming (Yuksekgonul et al., 2026). FrugalEvo's combines SLSQP with a linear program over radii (Chen et al., 2026). ThetaEvolve reports that its program reaches the best solution in three seconds of runtime (Wang et al., 2025). Once a model reliably writes a good local optimiser with a reasonable initialisation, the search over programs reduces to a search over initialisations and solver settings, which a greedy incumbent with many samples per round handles well. A benchmark on which the winning program is a call to a standard solver measures whether a harness can get a model to write that call, not whether it can find structure no solver finds.

Unsaturated records exist elsewhere, though how much of their headroom is open to search has not been measured. Sander (2026) reports improving the Packomania records for variable-radius circle packing at ten values of  $n$  between 101 and 114, by 2.4 to 5.4 percent, within 15 iterations and for 27.72 dollars in total. All ten records fell at iteration 0, to the hand-written seed, "a simple multi-start penalty L-BFGS-B solver with LP-optimal radii, used as the initial champion", before any model-proposed change; the 14 iterations of evolution that followed raised the total over all twelve targets from 59.39 to 59.98 (Sander, 2026, Tables 1 and 2). The percent-size gaps therefore measure how far the old records sat below a standard solver, not headroom that LLM search opened, and the system itself keeps a single incumbent ("we maintain only one champion"). Gaps of percent size, not parts per ten million, are where a harness comparison could be read, but only once the gap remaining above a standard solver has been measured. Erdos's problem, panel (b) of Figure 1, still has room: TTT-Discover's 0.380876 sits about  $48e-6$  below AlphaEvolve's bound and Hill Sampling's best runs at about  $26e-6$  and  $42e-6$  below it. But the four-decimal values in Gideoni et al.'s table (0.3810, 0.3811, 0.3812 for ShinkaEvolve and their two baselines, against 0.3809 for AlphaEvolve) cannot be placed on this scale at all.

The sets problem shows the opposite failure. There the cap is not tolerance but representation. AlphaEvolve's bound of 1.1584 used a set of 54,265 integers, and Gerbicz (2025) improved it to 1.173050 with an explicit construction containing more than  $10^{43546}$  elements, which no program can hold in memory as a list; Zheng (2025) improved it again to 1.173077 with a sequence of sets whose limit gives the bound. A search whose verifier expects an array of integers is capped by that representation, as Gideoni et al. (2026) argue in general: a problem's search space sets its ceiling, and on their

reformulated uncertainty inequality every method, simple or complex, moved from 0.3521 to 0.3482 for the same 20 dollars. Restricting the space has the symmetric effect: with circle radii capped at 0.1 every method found 2.514, and with Erdos step functions limited to ten steps every method found 0.3824. When the space caps the score, every harness that reaches the cap ties, and ties carry no information about machinery.

### 5.3 Information placed outside the search

The prompt, the initial program and the problem formulation carry domain knowledge, and the size of their effect can rival the size of the harness effects being measured.

Gideoni et al. (2026) ran ShinkaEvolve three times on circle packing with its default prompt, which tells the model the best known score, recommends variable sizes, hybrid layouts, attention to corners and edges and the use of scipy's SLSQP, and three times with a minimal prompt. With the default prompt it found the best-known packing in three of three runs; with the minimal prompt, in one of three. The default prompt then made the simple baselines worse: given ShinkaEvolve's prompt, IID sampling and sequential conditioned sampling reached 2.626 and 2.621, below what they reached with no domain knowledge at all. The authors' reading is that a prompt tuned against one pipeline's characteristic errors can hurt another pipeline. Hill Sampling's own circle prompt is the second phase of OpenEvolve's phased schedule, which names the AlphaEvolve target of 2.635 and lists seven of the same hints (Beck et al., 2026, Appendix B). Its no-information variant on circles reached the same best score, which suggests that by 2026 the models may carry the relevant knowledge themselves. On Erdos, removing the initial information substantially hurt, and on sets the initial program was, in the anchor's words, misleading, searching integers below 250 when good constructions are sparse and their largest element may run into the millions.

The AlphaEvolve team reports the same thing from the other side. Georgiev et al. (2025) found the advice in the prompt to have a significant impact on the final construction, found the system performing much better for a subject expert than for a non-expert, and stated that in general the best results came from human expertise combined with the system's computation. They also report that choosing the verifier was critical and that the system found loopholes in leaky verifiers. AlphaEvolve's own ablation list mixes machinery and information: "no context in the prompt" and "small base LLM only" are not harness components in the sense the machinery question asks about (Novikov et al., 2025).

The implication for attribution is direct. A comparison between harnesses that use different prompts or seeds measures the sum of a harness effect and an information effect, and the second can be larger. Gideoni et al.'s recommendation, to give every method the same minimal prompt and initial program, is the minimum. A comparison that wants to know whether machinery

helps a well-informed search would also need the matched informed condition, because the prompt that helps one design may hurt another.

### 5.4 The base model

The effects that have been measured on more than one model move with it. Zhang et al. (2024) found significant variance in performance attributable to the choice of model for every method they tested, "with the notable exception of the OBP (OR) problem where this variance is marginal", and singled out (1+1)-EPS as varying considerably across models because of its greedy nature, while EoH was stable on the travelling salesman problem across models. Gupta et al. (2026) found that exploration by occasional sampling from the full history gave nominally significant gains for Qwen2.5-3B on circle packing and for both GPT-OSS models on the Heilbronn triangle problem and scored lower on other pairs, and concluded that harness choice should be treated as a model- and problem-dependent hyperparameter. FrugalEvo's table reverses the order of two baselines between models: with GPT-5.6 Terra at 2 dollars, OpenEvolve's best circle run reached 2.635983 and ShinkaEvolve's 2.631285, while with GLM-5.3 at 1 dollar ShinkaEvolve's best reached 2.615012 and OpenEvolve's 2.598002 (Chen et al., 2026, Table 1).

ThetaEvolve shows the dependence most sharply for test-time training. On circle packing with ProRL-1.5B, RL raised the best of three seeds from 2.2491 without RL (at 600 steps) to 2.5225 (at 200 steps). With DeepSeek-R1-0528-Qwen3-8B, the best of three seeds was 2.6359857 with RL and 2.6359834 without, a gap of about  $2.3e-6$  measured under an evaluator with a  $1e-6$  tolerance (Wang et al., 2025, Table 2; the gap is the author's arithmetic). The same intervention is worth about a tenth of the score for a weak model and is at the resolution limit for a stronger one on this benchmark. On their third autocorrelation inequality, the 8B model's best fell from 1.5084 to 1.4930 with RL, a clearer effect on a benchmark with more room.

Hill Sampling's models, at 20 to 32 billion parameters, sit above ThetaEvolve's and below TTT-Discover's 120 billion. Each problem was paired with one model. Nothing in the anchor's design can separate "machinery does not help" from "machinery does not help this model on this problem".

### 5.5 The statistic and the number of runs

Discovery is judged by the best result a run finds. Comparisons of methods need a statistic over runs, and the choice and the number of runs decide how much a comparison can resolve.

Hill Sampling uses three seeds per arm. Its headline table reports the best seed, and its figures compare means with standard errors. Gideoni et al. (2026) used one ShinkaEvolve run per problem, because running all three methods cost more than 70 dollars per problem, and estimated their baselines' probability of matching that run by resampling their own runs. ThetaEvolve reports

the mean and best of three seeds (Wang et al., 2025). TTT-Discover reports the best kernel per run in its ablation (Yuksekgonul et al., 2026). FrugalEvo reports the mean, standard deviation and best of three runs (Chen et al., 2026). EvoTune's headline metric is the mean optimality gap of the top 50 programs over ten seeds, and its single-best-program table shows it ahead of FunSearch in most cases rather than all (Surina et al., 2025).

Gupta et al. (2026) is the only study in the set built around the statistical question. It ran every harness five times, built reference pools of 100 greedy runs for each Qwen pair and 30 for each GPT-OSS pair, and tested each harness's best-of-five against the distribution of the greedy baseline's best-of-five. Its strongest fixed harness, greedy with a 20 percent chance of picking a parent from the full history, won a majority of the 12 pairs with an estimated probability of 0.914 and a cross-pair p-value of 0.023, which became 0.678 after Holm correction. Its weakest, the full OpenEvolve configuration, had a majority-win probability of 0.033. These numbers show what a comparison with fewer runs and no correction cannot see. A component that helps on some model-problem pairs and hurts on others will appear, in a three-seed study on three pairs, as whatever sign those three pairs happen to give.

The history of other fields is a guide here. Henderson et al. (2018) showed that in deep reinforcement learning, variance across seeds and implementations made many claimed improvements hard to interpret without significance testing. Agarwal et al. (2021) showed that conclusions drawn from point estimates over a few runs diverged substantially from those drawn with interval estimates, and proposed robust aggregates. Gideoni et al. (2026) borrow Agarwal et al.'s probability of improvement for scaffold search and report, after re-evaluating scaffolds ten times, a probability of 0.49 that ShinkaEvolve beats IID sampling, which is to say no difference.

The choice between maximum and mean also interacts with what is being optimised. Hill Sampling's central weight-space finding is that ES raised mean return and lowered maximum return (Beck et al., 2026, Figure 2). EvoTune's headline gains are on a top-50 mean (Surina et al., 2025). A method can win on one statistic and lose on the other, and the discovery objective is the maximum.

### 5.6 How many accepted steps a record needs

The last variable is a property of the problem. Some records are reached in a few accepted improvements, others are built up over many. Hill Sampling's own table shows the spread: its best sets result appeared in round 2, its circle result in round 14, its Erdos result in round 70. The anchor ran its broadest machinery battery on Erdos on the stated ground that Erdos "requires the fewest rounds of improvement" (Beck et al., 2026, Section 5.3). That matches its configured runs, 80 rounds at  $N = 64$  for Erdos against 200 for circles and 600 for

sets, but not its rounds-to-record, on which Erdos took the most of the three. The two measures point opposite ways, so the Erdos battery cannot be placed at either end of this variable. Apart from Erdos, the problems in the 2026 simple-baseline studies appear to be mostly ones on which a capable model can write a strong optimiser in a handful of accepted steps.

The strongest evidence that machinery matters comes from problems of a different shape. AlphaEvolve's ablations were run on tensor decomposition and kissing numbers (Novikov et al., 2025). On the 11-dimensional kissing number, Gideoni et al.'s own table has AlphaEvolve at 593 against 402 for ShinkaEvolve and 438 for both simple baselines, a large gap, and their discussion of the sequential baseline was motivated by exactly this problem, where solutions must extend known constructions rather than be written afresh. AlphaEvolve's budget there is unknown and probably much larger, so the gap is not a harness comparison; it is a reminder that the 2026 baselines were not shown to close it. TTT-Discover's ablation, which shows reuse and training both mattering, is on a GPU kernel task where each improvement builds on the last: removing reuse while keeping training moved the best kernel from 1,203.10 to 5,274.03 microseconds, and an epsilon-greedy reuse rule reached 1,328.89 (Yuksekgonul et al., 2026, Table 8).

The classical literature would lead one to expect this. Lehman and Stanley (2011) argue that following the objective can lead a search into dead ends when the landscape is deceptive, and the number of offspring per generation can change the efficiency of search a great deal (Jansen et al., 2005). A problem whose record is one good optimiser away from the start offers little deception; a problem whose record is a long chain of constructions may offer much. The 2026 simple-baseline results draw mostly on the first kind; where they include the second, as with the kissing number, the baselines did not close the gap.

## 6. Test-time learning: evolution strategies are not reinforcement learning, and a mean is not a maximum

The most quotable line in Hill Sampling's abstract is that learning the weights by evolution strategies did worse than running evolution strategies with the learning rate set to zero. The result is reported carefully, and its scope is narrower than the quotation suggests.

On the other side of this dispute stand three lines of work. ThetaEvolve reports that RL during the search consistently outperformed inference-only runs across two open models and four problems, and that a checkpoint trained this way, used later for pure inference, made faster progress on the trained task and on unseen ones (Wang et al., 2025). TTT-Discover trains gpt-oss-120b with LoRA during the search, reports new best results in almost every problem it attempted, among them Erdos's problem, an autocorrelation inequality and a GPU kernel competition, and in its kernel ablation finds the best

kernel at 1,203.10 microseconds with training and 2,060.70 without, holding its reuse rule fixed (Yuksekgonul et al., 2026). EvoTune interleaves RL fine-tuning, by DPO, with FunSearch-style evolution and reports consistently lower optimality gaps than FunSearch across models, tasks and budgets of 9,600 to 22,400 sampled programs (Surina et al., 2025). PACEvolve++ trains an advisor model during evolutionary search and reports outperforming, with faster convergence, the state-of-the-art evolutionary search framework it compares with (Yan et al., 2026b).

These systems and Hill Sampling’s ES arm differ in at least four ways, each of which could carry the difference in verdict.

The update rule differs. ThetaEvolve trains with GRPO and TTT-Discover with a policy-gradient objective of its own design; EvoTune updates the model with an off-policy form of DPO on preference pairs drawn from the search. Hill Sampling uses ES over all parameters with antithetic Gaussian perturbations. Qiu et al. (2025) argue that ES at billion-parameter scale can match or outperform RL on several axes for fine-tuning, so ES is not a weak method in general, but its success there was on fine-tuning tasks, not on discovery.

The objective differs. TTT-Discover designs its objective to prize the single best outcome: an entropic utility whose parameter beta is set adaptively per starting state and which tends to the maximum as beta grows, and a reuse rule whose value term is the best child rather than the mean (Yuksekgonul et al., 2026). Its ablation shows the objective mattering: replacing it with expected reward moved the best kernel from 1,203.10 to 1,985.67 microseconds, and a constant beta to 1,483.83. Walder and Karkhanis (2025) made the general argument that optimising expected reward favours individually strong samples at the expense of the joint value of a set, and derived estimators for optimising pass@k directly. Hill Sampling’s standard ES optimises a mean, and its Figure 2 shows exactly the predicted mismatch: mean return rises, maximum return falls. Its ES (max@8) and softmax variants move towards a maximum objective, and they too failed to beat Hill Sampling, which is evidence that ES with a crude max objective does not help in this setting. It is not evidence about an entropic policy-gradient objective, which the anchor does not test. The anchor also notes that the softmax weighting it borrowed follows Yuksekgonul et al. (2026), so that variant is the closest its ES arm comes to TTT-Discover’s objective.

The model differs. ThetaEvolve’s large gain was for a 1.5-billion-parameter model; for the 8-billion model on circle packing the gain was at the evaluator’s resolution (Section 5.4). TTT-Discover’s kernel gains are for a 120-billion model trained through LoRA of rank 32. Hill Sampling’s ES runs are on 20 to 32 billion parameters, full-parameter.

The statistic and problem differ. EvoTune’s headline is a top-50 mean. TTT-Discover’s kernel task is one where improvements compound. Hill Sampling’s ES

comparisons are on circles, Erdos and sets, problems on which, by Section 5.6, the evidence on how much accumulation matters is mixed: the circle and sets records came in rounds 14 and 2, but the Erdos record in round 70.

The conclusion this licenses is scoped. Hill Sampling shows that full-parameter ES at its scale and objectives did not improve the best program found on its three problems, and that adding it to Hill Sampling did not help. The anchor’s conclusion that test-time parameter learning is an unnecessary complication holds for that method on those problems. Whether max-seeking policy-gradient training at test time adds to a strong frozen incumbent search on problems with headroom has not been tested by anyone in a matched design. TTT-Discover’s own ablation does not test it either, because its no-training arm keeps PUCT reuse, not a greedy incumbent with large breadth, and Gupta et al. (2026) ablated TTT-Discover’s search harness but explicitly not its training component.

## 7. Weight noise and token noise

A narrower result in Hill Sampling deserves its own note because it is easy to over-read. Gan and Isola (2026) report that in large, well-pretrained models, task-improving variants become dense in the neighbourhood of the pretrained weights, and that sampling random perturbations, keeping the best and ensembling them by majority vote is competitive with PPO, GRPO and ES as a post-training method. Cendra et al. (2026) report that Bayesian optimisation in a random low-dimensional embedding of weight space matches or beats that random search with five times fewer candidate evaluations, on models of 0.5 to 3 billion parameters. Hill Sampling’s finding is that, for program discovery with an exact verifier, token-level sampling from the unperturbed model gave more useful diversity than fixed Gaussian weight perturbations at any of the noise scales tried (Beck et al., 2026, Section 5.1).

These results are about different objectives and do not conflict. Neural thickets concern generalisation to held-out inputs of a task, where an ensemble of perturbed experts is the product. Program discovery needs one exceptional output, verified exactly, and the anchor’s comparison asks only which source of randomness generates it more often. The anchor’s own framing says as much. Its model-noise comparison also uses one seed per noise scale, which the anchor flags, so its ordering among noise scales is weak. The finding that a perturbed model is a worse sampler than a temperature-1 model, for this purpose, stands at the level the anchor claims.

Bahlous-Boldi et al. (2026) add a qualification from the other direction. Training a model to produce diverse solutions across the components of a vector-valued reward matched or beat the strongest scalar RL baselines on best-of-k and pass@k, with the gap widening as the search budget grew, and in evolutionary search it let models solve problems that GRPO-trained models could not. If diversity of the sampler is a lever, then the

sampler's training, not only the harness around it, is a variable in every comparison of harnesses, and models post-trained in different ways may disagree about which harness is best.

## 8. This has happened before

The pattern in this literature, elaborate methods losing to simple baselines once comparisons are matched, has occurred in several neighbouring fields, and how those episodes resolved is instructive.

In neural architecture search, Li and Talwalkar (2019) showed random search with early stopping performing at least as well as ENAS on two standard benchmarks and documented how hard published results were to reproduce. Yu et al. (2019) found state-of-the-art search policies performing, on average, similarly to random selection of architectures. Yang et al. (2020) benchmarked eight methods on five datasets, found many unable to beat the average randomly sampled architecture significantly, and traced much of the reported performance to evaluation tricks and to the hand-designed search space. In reinforcement learning, Mania et al. (2018) matched state-of-the-art sample efficiency on MuJoCo locomotion with random search over static linear policies and, by running hundreds of seeds, showed how variable the benchmark results were. In generative modelling, Lucic et al. (2018) found that most GAN variants reached similar scores given enough hyperparameter search and restarts, and found no evidence that any tested variant consistently beat the original non-saturating GAN. In language modelling, Melis et al. (2017) found that properly regularised standard LSTMs, with large-scale black-box tuning, outperformed more recent architectures.

The lesson these papers drew was not that the elaborate methods were worthless but that the comparisons had to change. Li and Talwalkar released code, seeds and documentation and reported results over multiple runs. Yang et al. proposed measuring each method against the average randomly sampled architecture and set out best practices. Henderson et al. (2018) asked for significance metrics and tighter standards of reporting in RL, and Agarwal et al. (2021) for interval estimates and robust aggregates. Lucic et al. called for more systematic and objective evaluation. In each case the finding that the baseline matched the method was a finding about the comparison, and the remedy proposed was a better comparison. Yang et al.'s observation that the search space contributed more than the search strategy is the same point Gideoni et al. (2026) make about problem formulation. The LLM program-search literature appears to be at roughly the stage those fields were in around 2019: simple baselines have been shown competitive, and the protocol that would show where elaborate methods earn their cost does not yet exist.

## 9. Consolidated evidence and an attribution procedure

Each row of Table 1 is one published result on which the argument depends. Its final column records which question or variable the result informs, in the author's reading under Sections 3 to 6, and none of the sources uses these codes. Numbers in the result column are as each source states them.

Table 1 makes two patterns visible. The rows bearing on S point one way. The rows bearing on M point both ways, and when they are sorted by the other codes, the ones favouring machinery are disproportionately those with evaluation-counted budgets, longer accumulation or weaker models, while those against are disproportionately those with completion- or dollar-counted budgets, saturated benchmarks or a single model per problem. The sorting is the author's and is not a statistical test; it is the pattern Algorithm 1 is built to check.

Table 2 sets the designs side by side. Even studies that share a nominal unit do not share a budget (rollouts of different lengths and models; dollars at different prices), and only two of the seven run more than three seeds per arm.

Algorithm 1 states the decision structure this paper applies to a reported gain for a search component. It is meant to be run on a methods section before a headline is repeated.

## 10. The comparisons that would settle the open part

The state question is answered well enough to act on. The machinery question needs comparisons that no published study has run. Five would settle most of it.

**A benchmark with measured headroom.** Choose problems on which the gap between the best known result and what a strong greedy incumbent reaches in a fixed budget is large relative to the verifier's tolerance and to seed variance, and report that gap before comparing harnesses. Variable-radius packings at larger  $n$ , where ten records fell by percents to a standard seed solver before any model-proposed change, so that the records are unsaturated but the headroom left for search beyond such a solver has not been measured (Sander, 2026), and problems like the 11-dimensional kissing number, where the gap between AlphaEvolve and the 2026 baselines is large (Gideoni et al., 2026), are candidates. Circle packing at  $n = 26$  should be retired as a ranking benchmark, or kept only as a smoke test with a declared tolerance.

**Budgets reported in every unit.** Each arm should report evaluations, completions, input and output tokens, dollars at stated prices, and wall-clock on stated hardware, so that a reader can re-rank the methods under the unit that matters to them. FrugalEvo's cost-indexed area under the best-so-far curve (Chen et al., 2026) and Gupta et al.'s per-step score pools (Gupta et

**Table 1.** Reported findings on LLM-guided program search, with the question or variable each bears on (S state; M machinery; T test-time training; B budget; H headroom; I information; Mo model; St statistic; P problem structure), as read by the author.

| Source                  | Setting                     | Comparison                                    | Budget              | Reported result                                              | Bears on |
|-------------------------|-----------------------------|-----------------------------------------------|---------------------|--------------------------------------------------------------|----------|
| Novikov et al. 2025     | tensor decomp., kissing     | full vs ablations                             | not stated          | each removal reduces results                                 | S, I     |
| Zhang et al. 2024       | 4 heuristic tasks, 9 LLMs   | (1 + 1)-EPS vs LLM alone                      | queries             | baseline better with 10x fewer queries                       | S        |
| Zhang et al. 2024       | same, 5 runs                | (1 + 1)-EPS vs EoH, ReEvo, FunSearch          | queries             | no method consistently best                                  | M, Mo    |
| Lange et al. 2025       | circles                     | weighted vs hill climb vs BoN                 | evaluations         | weighted best; hill climb plateaus                           | M, B     |
| Lange et al. 2025       | circles                     | novelty rejection vs none                     | evaluations         | rejection strongly better                                    | M        |
| Beck et al. 2026        | circles, 20B, 3 seeds       | Hill Sampling vs RS, ES                       | 7,168 compl.        | 2.635983085; ES worst                                        | S, T, H  |
| Beck et al. 2026        | Erdos, 24B                  | Hill Sampling vs 10 add-ons                   | completions         | none significantly better                                    | M, P     |
| Beck et al. 2026        | sets, 32B                   | all methods                                   | completions         | no significant difference, ~95% of AE                        | H        |
| Beck et al. 2026        | circles                     | ES learning rate > 0 vs 0                     | completions         | mean up, max best at 0                                       | T, St    |
| Gideoni et al. 2026     | 9 AE problems, 1 Shinka run | SCS, IID RS vs Shinka                         | \$20                | SCS $\geq$ Shinka on 6/9, IID on 4/9                         | M, St    |
| Gideoni et al. 2026     | circles                     | Shinka default vs minimal prompt              | \$20                | best-known found 3/3 vs 1/3                                  | I        |
| Gideoni et al. 2026     | circles                     | baselines given Shinka prompt                 | \$20                | 2.626, 2.621, below no hints                                 | I        |
| Gideoni et al. 2026     | uncertainty ineq.           | reformulated space, all methods               | \$20                | 0.3521 to 0.3482 for all                                     | H        |
| Gideoni et al. 2026     | kissing, 11D                | AE vs Shinka vs baselines                     | AE unknown          | 593 vs 402 vs 438                                            | P        |
| Gideoni et al. 2026     | ALE-Bench LITE              | baselines vs Shinka                           | 50 programs         | baselines ahead on 8/10                                      | M        |
| Gupta et al. 2026       | 12 model-problem pairs      | 30 harnesses vs greedy, 5 runs                | 160-1,600 rollouts  | best p = 0.023, 0.678 after Holm                             | M, St    |
| Gupta et al. 2026       | same                        | full OpenEvolve-style                         | same                | majority-win prob. 0.033                                     | M        |
| Gupta et al. 2026       | same                        | adaptive allocation vs greedy                 | 5 run-equiv.        | 85.75% vs 84.35%                                             | B        |
| Wang et al. 2025        | circles, 1.5B, 3 seeds      | RL vs no RL                                   | steps               | best 2.5225 vs 2.2491                                        | T, Mo    |
| Wang et al. 2025        | circles, 8B, 3 seeds        | RL vs no RL                                   | steps               | best 2.6359857 vs 2.6359834                                  | T, H     |
| Yuksekgonul et al. 2026 | Erdos, 120B                 | TTT vs BoN vs OpenEvolve                      | 25,600 rollouts     | 0.380876, 0.380906, 0.380965                                 | S, T     |
| Yuksekgonul et al. 2026 | TriMul kernel               | train, reuse ablations                        | 25,600 rollouts     | 1,203 full; 2,061 no TTT; 5,352 BoN                          | T, P     |
| Surina et al. 2025      | combinatorial opt.          | EvoTune vs FunSearch                          | 9.6k-22.4k programs | better top-50 mean, 10 seeds                                 | T, St    |
| Assumpcao et al. 2025   | 9 AE problems               | CodeEvolve vs OpenEvolve, Shinka              | matched             | ahead on 6/9                                                 | M        |
| Chen et al. 2026        | circles, 2 model sets       | OpenEvolve vs Shinka                          | \$2, \$1            | order reverses across models                                 | Mo, B    |
| Chen et al. 2026        | circles                     | FrugalEvo                                     | \$1.68              | 2.635996, radii raised within tolerance                      | H        |
| Sander 2026             | Packomania, n = 101-114     | seed solver, then evolved solvers, vs records | \$27.72             | records improved by 2.4-5.4%, all by the seed at iteration 0 | H        |

**Algorithm 1.** Attributing a reported gain of harness X over harness Y.

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>Input: a study reporting <math>X &gt; Y</math> on a verified-score benchmark</p> <p>Step 1: QUESTION. Does Y keep the best program and condition on it?</p> <p>no -&gt; the comparison bears on the state question only; stop here for machinery.</p> <p>yes -&gt; continue: the comparison bears on machinery.</p> <p>Step 2: HEADROOM. Is <math> \text{score}(X) - \text{score}(Y) </math> larger than the verifier tolerance and larger than the spread of Y over its own seeds?</p> <p>no -&gt; record a tie; the benchmark cannot rank X and Y.</p> <p>Step 3: INFORMATION. Do X and Y share prompt, seed program and formulation?</p> <p>no -&gt; gain = harness effect + information effect; flag as unattributed.</p> <p>Step 4: BUDGET. Is the budget matched in the unit the deployment pays for (evaluations, completions, tokens, dollars or hours), with the breadth/depth split reported for both arms?</p> <p>no -&gt; restate the gain per unit actually matched; do not transfer it.</p> <p>Step 5: MODEL. Was the comparison repeated on more than one base model?</p> <p>no -&gt; scope the claim to that model.</p> <p>Step 6: STATISTIC. How many runs per arm, which statistic (max, mean, top-k mean), and was the test corrected for the number of comparisons?</p> <p>fewer than 5 runs or uncorrected -&gt; report as suggestive.</p> <p>Step 7: STRUCTURE. How many accepted improvements did the best run need?</p> <p>few -&gt; the problem cannot show the value of populations; scope accordingly.</p> <p>Output: a verdict scoped to (problem, model, budget unit, information, statistic).</p> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Table 2.** Design of the studies most cited in the machinery dispute, from each source's methods section.

| Study                   | Budget unit    | Models                  | Runs per arm                    | Statistic reported       | Problems                            |
|-------------------------|----------------|-------------------------|---------------------------------|--------------------------|-------------------------------------|
| Beck et al. 2026        | completions    | one per problem, 20-32B | 3                               | best seed; mean and SE   | circles, sets, Erdos                |
| Gideoni et al. 2026     | dollars        | Gemini 2.5 family       | 1 Shinka; resampled baselines   | best; prob. of match     | 9 AE math problems, 3 other domains |
| Gupta et al. 2026       | rollouts       | 4, 3B-120B              | 5, against 30-100 baseline runs | best-of-5 vs null, Holm  | circles, Heilbronn, 2nd autocorr.   |
| Wang et al. 2025        | training steps | 1.5B, 8B                | 3                               | mean and best            | 4 problems                          |
| Yuksekgonul et al. 2026 | rollouts       | 8B, 120B                | not stated                      | best kernel or bound     | math, kernels, algorithms, biology  |
| Chen et al. 2026        | dollars        | 2 model families        | 3                               | mean, sd, best, cost AUC | 20 tasks                            |
| Zhang et al. 2024       | queries        | 9 LLMs                  | 5                               | mean gap to optimum      | 4 heuristic tasks                   |

al., 2026) are the right kinds of object; the two should be combined.

**A breadth sweep for the greedy incumbent.** Hill Sampling at several values of  $N$ , from one child per round to hundreds, at fixed total completions, on the same problems as ShinkaEvolve's ablation. If greedy search plateaus at small  $N$  and wins at large  $N$ , the two verdicts on hill climbing are reconciled and the size of  $N$  becomes the quantity a harness must beat.

**A two-by-two on information.** Every harness comparison run twice, once with a minimal shared prompt and seed and once with an expert prompt written for neither harness, so that the harness effect and the information effect can be separated and their interaction measured (Gideoni et al., 2026; Georgiev et al., 2025).

**A matched test of test-time training.** The strongest frozen baseline available, a greedy incumbent with tuned breadth or a lightweight tree search of the kind Gupta et al. found strongest, against the same search with max-seeking policy-gradient training at test time, on at least two model sizes and on one saturated and one unsaturated problem, with at least five runs per arm and the statistic fixed in advance as the maximum. This is the comparison neither Hill Sampling nor TTT-Discover ran.

In all five, the statistical protocol of Gupta et al. (2026), repeated runs against a baseline null distribution with correction across pairs, should be the floor, and results should be reported for each model-problem pair separately before any aggregate.

## 11. The strongest case against this paper

The strongest objection is that the paper explains away a clean result. Hill Sampling, Gideoni et al. and Gupta et al. are three independent groups with three designs reaching the same conclusion, and when complex machinery cannot beat a greedy baseline across three designs, the burden should be on the machinery. The six variables are a list of ways any result could fail to transfer; listing them does not show that they explain this one.

The objection is partly right. The burden is on the machinery to show its value, and nothing in this paper shows that it has any. The paper's claim is narrower: the evidence that machinery has no value was gathered where its value, if any, would be hardest to see. That claim is testable and could be wrong. If a breadth sweep, a headroom benchmark and a matched information design all show the greedy incumbent tying the complex harnesses, the paper's reading fails and the 2026 conclusion stands in its strong form.

A second objection is that the headroom argument cuts against AlphaEvolve as much as against Hill Sampling: the  $n = 26$  record that AlphaEvolve set in 2025 was a small difference too, and the paper should not treat that as meaningful while treating later differences as noise. The reply is quantitative. AlphaEvolve's 2.6358628 exceeded the previous best of 2.634 that its paper states by about  $1.9e-3$ , and the later move from AlphaEvolve's value to the 2.635983 cluster was about  $1.2e-4$  (the author's arithmetic on the reported values). Both are at least two orders of magnitude above the largest tolerance in use. The differences inside the cluster, about  $2.5e-7$  and below, are not. The argument is about resolution, not about which side benefits.

A third objection is that the paper is itself an instance of the attribution problem it describes, since it assigns codes to results and reads patterns into a table without a statistical test. That is correct, and the paper says so at Table 1. Algorithm 1 exists so that the reading can be checked result by result rather than taken on trust.

## 12. What is not known

Whether any search component beyond a well-tuned greedy incumbent improves results on problems with headroom, at matched budgets in any unit, is not known.

How the optimal number of samples per round for a greedy incumbent depends on the problem and the model is not known, and no published study has swept it.

Whether max-seeking policy-gradient training at test time adds to a strong frozen incumbent search, as opposed to adding to a tree-search harness or to a weak

model, is not known.

How much of the reported progress on AlphaEvolve's problems since 2025 comes from stronger models, better prompts and formulations, and larger budgets rather than from search design is not known, because no study varies these jointly.

How large the variance of a single discovery run is on most of these problems is not known; only Gupta et al. (2026) report reference pools large enough to estimate it, and only for three problems.

### 13. Limitations

The paper reads published results and is limited by what those results report. Several numbers the argument uses come from figures whose values are not tabulated, and the paper has used only values stated in text or tables. AlphaEvolve's budgets are not public, so no comparison against AlphaEvolve in this paper or in the sources is budget-matched. Several of the sources are 2026 preprints without peer review, including the anchor; their numbers are reported as their authors state them. The codes in Table 1 and the pattern read from them are the author's judgement. The account of what problems need many accepted steps rests on a small number of reported rounds-to-record and on the classical theory, and is offered as a hypothesis to test, not a finding. The literature search covered arXiv and the venues cited by the sources through early October 2026, and work outside that net may change parts of the account.

### 14. Conclusion

The 2026 simple-baseline papers have done the field a service. They have shown that the component of LLM-guided program search that does most of the work is the simplest one: keep the best verified program and keep sampling changes to it. That was already the lower-bound baseline of 2024 and the root of every harness Gupta et al. decomposed, and Hill Sampling has shown that run broadly, with open models and modest hardware, it reaches published-best scores. That is a default worth adopting.

What they have not shown is that the rest of the machinery is unnecessary. They have shown that its value cannot be seen on a benchmark whose leading results differ by less than its verifiers' tolerances, under budgets counted in a unit that hides what some components save, with prompts that move results as much as harnesses do, on one model per problem and three runs per arm. Those are reasons to measure better, and Section 10 states what better would be. Until those measurements exist, the honest summary is the one the sources' own data support: the incumbent did most of the work, and the question of what else pays is open.

### References

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A., & Bellemare, M. G. (2021). Deep Reinforcement Learning at the Edge of the Statistical Precipice. *NeurIPS 2021*. arXiv:2108.13264. doi:10.48550/arXiv.2108.13264.

Assumpcao, H., Ferreira, D., Campos, L., & Murai, F. (2025). CodeEvolve: an open source evolutionary coding agent for algorithmic discovery and optimization. arXiv:2510.14150. doi:10.48550/arXiv.2510.14150.

Bahlous-Boldi, R., Puri, I., Shenfeld, I., Kumar, A., et al. (2026). Vector Policy Optimization: Training for Diversity Improves Test-Time Search. arXiv:2605.22817. doi:10.48550/arXiv.2605.22817.

Beck, J., Ogren, P. V., & Kobren, A. (2026). Hill Sampling for Test-Time Scaling: A Simple and Better Alternative to Repeated Sampling, Evolution, and Training. arXiv:2609.25510. doi:10.48550/arXiv.2609.25510.

Beyer, H.-G., & Schwefel, H.-P. (2002). Evolution strategies - A comprehensive introduction. *Natural Computing*, 1, 3-52. doi:10.1023/A:1015059928466.

Brown, B., Juravsky, J., Ehrlich, R., Clark, R., Le, Q. V., Re, C., & Mirhoseini, A. (2024). Large Language Monkeys: Scaling Inference Compute with Repeated Sampling. arXiv:2407.21787. doi:10.48550/arXiv.2407.21787.

Cendra, N. B., Ezzer, A., Cendra, F. J., Knoblauch, J., et al. (2026). Can Bayesian Optimization Efficiently Find a Strong Single Expert in Neural Thickets? arXiv:2608.10867. doi:10.48550/arXiv.2608.10867.

Chen, H., Qi, X., Zhao, J. X., Feng, Z., et al. (2026). FrugalEvo: Towards Cost-Aware LLM-Guided Program Evolution. arXiv:2610.03675. doi:10.48550/arXiv.2610.03675.

Gan, Y., & Isola, P. (2026). Neural Thickets: Diverse Task Experts Are Dense Around Pretrained Weights. arXiv:2603.12228. doi:10.48550/arXiv.2603.12228.

Georgiev, B., Gomez-Serrano, J., Tao, T., & Wagner, A. Z. (2025). Mathematical exploration and discovery at scale. arXiv:2511.02864. doi:10.48550/arXiv.2511.02864.

Gerbic, R. (2025). Sums and differences of sets (improvement over AlphaEvolve). arXiv:2505.16105. doi:10.48550/arXiv.2505.16105.

Gideoni, Y., Risi, S., & Gal, Y. (2026). Simple Baselines are Competitive with Code Evolution. arXiv:2602.16805. doi:10.48550/arXiv.2602.16805.

Gupta, A., Lei, J., Lu, A., Anumanchipalli, G., & Choshen, L. (2026). Automated Discovery Has No Universally Superior Harness. arXiv:2607.18235. doi:10.48550/arXiv.2607.18235.

Haugland, J. K. (2016). The minimum overlap problem revisited. arXiv:1609.08000. doi:10.48550/arXiv.1609.08000.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018). Deep Reinforcement Learning that Matters. *AAAI 2018*. arXiv:1709.06560. doi:10.48550/arXiv.1709.06560.

Jansen, T., De Jong, K. A., & Wegener, I. (2005). On the Choice of the Offspring Population Size in Evolutionary Algorithms. *Evolutionary Computation*, 13(4), 413-440. doi:10.1162/106365605774666921.

Lange, R. T., Imajuku, Y., & Cetin, E. (2025). ShinkaEvolve: Towards Open-Ended And Sample-Efficient Program Evolution. arXiv:2509.19349. doi:10.48550/arXiv.2509.19349.

Lehman, J., Gordon, J., Jain, S., Ndousse, K., Yeh, C., & Stanley, K. O. (2022). Evolution through Large Models. arXiv:2206.08896. doi:10.48550/arXiv.2206.08896.

Lehman, J., & Stanley, K. O. (2011). Abandoning Objectives: Evolution Through the Search for Novelty Alone. *Evolutionary Computation*, 19(2), 189-223. doi:10.1162/EVCO\_a\_00025.

Li, L., & Talwalkar, A. (2019). Random Search and Reproducibility for Neural Architecture Search. *UAI 2019*. arXiv:1902.07638. doi:10.48550/arXiv.1902.07638.

Liu, F., Tong, X., Yuan, M., Lin, X., et al. (2024). Evolution of Heuristics: Towards Efficient Automatic Algorithm Design Using Large Language Model. arXiv:2401.02051. doi:10.48550/arXiv.2401.02051.

Liu, S., Agarwal, S., Maheswaran, M., Cemri, M., et al. (2026). EvoX: Meta-Evolution for Automated Discovery. arXiv:2602.23413. doi:10.48550/arXiv.2602.23413.

- Lucic, M., Kurach, K., Michalski, M., Gelly, S., & Bousquet, O. (2018). Are GANs Created Equal? A Large-Scale Study. *NeurIPS 2018*. arXiv:1711.10337. doi:10.48550/arXiv.1711.10337.
- Mania, H., Guy, A., & Recht, B. (2018). Simple random search provides a competitive approach to reinforcement learning. arXiv:1803.07055. doi:10.48550/arXiv.1803.07055.
- Melis, G., Dyer, C., & Blunsom, P. (2017). On the State of the Art of Evaluation in Neural Language Models. arXiv:1707.05589. doi:10.48550/arXiv.1707.05589.
- Mouret, J.-B., & Clune, J. (2015). Illuminating search spaces by mapping elites. arXiv:1504.04909. doi:10.48550/arXiv.1504.04909.
- Novikov, A., Vu, N., Eisenberger, M., Dupont, E., et al. (2025). AlphaEvolve: A coding agent for scientific and algorithmic discovery. arXiv:2506.13131. doi:10.48550/arXiv.2506.13131.
- Qiu, X., Gan, Y., Hayes, C. F., Liang, Q., et al. (2025). Evolution Strategies at Scale: LLM Fine-Tuning Beyond Reinforcement Learning. *ICML 2026*. arXiv:2509.24372. doi:10.48550/arXiv.2509.24372.
- Romera-Paredes, B., Barekatin, M., Novikov, A., Balog, M., Kumar, M. P., et al. (2024). Mathematical discoveries from program search with large language models. *Nature*, 625, 468-475. doi:10.1038/s41586-023-06924-6.
- Salimans, T., Ho, J., Chen, X., Sidor, S., & Sutskever, I. (2017). Evolution Strategies as a Scalable Alternative to Reinforcement Learning. arXiv:1703.03864. doi:10.48550/arXiv.1703.03864.
- Sander, W. (2026). LLM-Guided Program Evolution for Circle Packing: Breaking 10 Packomania Records for \$28. arXiv:2609.05093. doi:10.48550/arXiv.2609.05093.
- Snell, C., Lee, J., Xu, K., & Kumar, A. (2024). Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters. arXiv:2408.03314. doi:10.48550/arXiv.2408.03314.
- Stroebl, B., Kapoor, S., & Narayanan, A. (2024). The Limits of Inference Scaling Through Resampling. arXiv:2411.17501. doi:10.48550/arXiv.2411.17501.
- Surina, A., Mansouri, A., Quaendvlieg, L., Seddas, A., et al. (2025). Algorithm Discovery With LLMs: Evolutionary Search Meets Reinforcement Learning. arXiv:2504.05108. doi:10.48550/arXiv.2504.05108.
- Walder, C., & Karkhanis, D. (2025). Pass@K Policy Optimization: Solving Harder Reinforcement Learning Problems. arXiv:2505.15201. doi:10.48550/arXiv.2505.15201.
- Wang, Y., Su, S.-R., Zeng, Z., Xu, E., et al. (2025). ThetaEvolve: Test-time Learning on Open Problems. arXiv:2511.23473. doi:10.48550/arXiv.2511.23473.
- Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67-82. doi:10.1109/4235.585893.
- Yan, M., Peng, B., Coleman, B., Chen, Z., et al. (2026a). PACEvolve: Enabling Progress-Aware Consistent Evolution. arXiv:2601.10657. doi:10.48550/arXiv.2601.10657.
- Yan, M., Peng, B., Coleman, B., Chen, Z., et al. (2026b). PACEvolve + +: Improving Test-time Learning for Evolutionary Search Agents. arXiv:2605.07039. doi:10.48550/arXiv.2605.07039.
- Yang, A., Esperanca, P. M., & Carlucci, F. M. (2020). NAS evaluation is frustratingly hard. *ICLR 2020*. arXiv:1912.12522. doi:10.48550/arXiv.1912.12522.
- Yang, C., Wang, X., Lu, Y., Liu, H., Le, Q. V., Zhou, D., & Chen, X. (2024). Large Language Models as Optimizers. *ICLR 2024*. arXiv:2309.03409. doi:10.48550/arXiv.2309.03409.
- Ye, H., Wang, J., Cao, Z., Berto, F., et al. (2024). ReEvo: Large Language Models as Hyper-Heuristics with Reflective Evolution. *NeurIPS 2024*. arXiv:2402.01145. doi:10.48550/arXiv.2402.01145.
- Yu, K., Sciuto, C., Jaggi, M., Musat, C., & Salzmann, M. (2019). Evaluating the Search Phase of Neural Architecture Search. arXiv:1902.08142. doi:10.48550/arXiv.1902.08142.
- Yuksekgonul, M., Kocaja, D., Li, X., Bianchi, F., et al. (2026). Learning to Discover at Test Time. arXiv:2601.16175. doi:10.48550/arXiv.2601.16175.
- Zhang, R., Liu, F., Lin, X., Wang, Z., Lu, Z., & Zhang, Q. (2024). Understanding the Importance of Evolutionary Search in Automated Heuristic Design with Large Language Models. *PPSN 2024*. arXiv:2407.10873. doi:10.48550/arXiv.2407.10873.
- Zheng, F. (2025). Sums and differences of sets: a further improvement over AlphaEvolve. arXiv:2506.01896. doi:10.48550/arXiv.2506.01896.