EVODUET: BILEVEL CO-EVOLUTION OF WEB SEARCHING AND TASK SOLVING FOR SCIENTIFIC DISCOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

Evolutionary search with large language models (LLMs) can stall when progress requires external knowledge the model lacks. Supplying relevant documents helps, but simply adding web search tool can keep returning the same pages as solutions change. We introduce **EVODUET**, a bi-level optimization method that co-evolves *solutions* and *search queries* with fixed model parameters. At each iteration, a retrieval gate lets the LLM assess its knowledge gap and choose to retrieve new documents, reuse stored ones, or proceed without them. An inner loop refines queries and ranks documents by the solution scores they are predicted to yield; an outer loop generates candidates in parallel from these documents and records the evaluated outcomes for later searches. Across 21 optimization tasks with one candidate per iteration, EVODUET raises OpenEvolve's normalized discovery gain from 74.1% to 78.0% with GPT-5.6-Luna and from 61.3% to 82.3% with Gemini-3.8-Flash, whereas Qwen3.5-9B does not benefit. Our best runs surpass the previously reported best scores on eight tasks, including Swap Reduction on Q20 and Rosetta, and match them on three more. EVODUET also improves with other scaffolds (e.g., Top-K, EvoX) on Sums/Diffs and Denoising, demonstrating its applicability across evolutionary search scaffolds.

1 INTRODUCTION

Evolutionary search scaffolds driven by large language models (LLMs) (Romera-Paredes et al., 2024; Novikov et al., 2025) have begun to discover new solutions to optimization tasks such as the Erdős minimum-overlap problem (Erdős, 1955), GPU kernel design (Ouyang et al., 2025), and single-cell RNA-seq denoising (Yuksekgonul et al., 2026). In these tasks, an evaluator can score any candidate solution, but the optimal solution cannot be computed directly (Kirkpatrick et al., 1983). Scaffolds therefore iterate: the LLM proposes candidates, the evaluator scores them, and high-scoring candidates seed the next iteration.

Most scaffolds (Sharma, 2025; Lange et al., 2025; Cemri et al., 2026) run only this *solution loop*. Others keep the solution loop from saturating by wrapping additional loops around it, a design we call *Loop Packing* (Figure 1). For example, EvoX (Liu et al., 2026a) adds a strategy loop that revises the search strategy when progress stagnates. However, every loop packed so far draws on the same two sources of knowledge: the run's evolutionary history and the LLM's parametric knowledge. The search is therefore **closed**, and it stalls when progress requires external knowledge that neither source contains (e.g., how to use a newer library version). Human scientists, by contrast, consult the literature and search again as new knowledge gaps emerge.

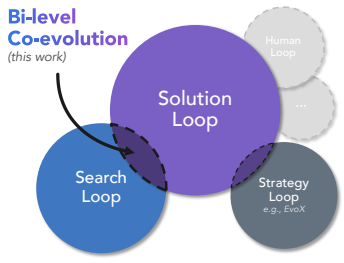


Figure 1: Packing loops of multiple optimization scaffolds

Our preliminary analysis (§ 2.3) shows that external knowledge helps LLM-driven search, but not automatically. Supplying task-relevant and helpful documents at every iteration (oracle documents) raises OpenEvolve's average Normalized Discovery Gain (NDG) on 31 tasks by 10.3% with Qwen3.5-9B and 4.0% with GPT-5.6-Luna. Two further observations show that turning such knowledge into progress takes more than a search tool: First, the LLM needs room to explore: on Sums/Diffs, oracle documents help only when the LLM generates eight candidates per iteration

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

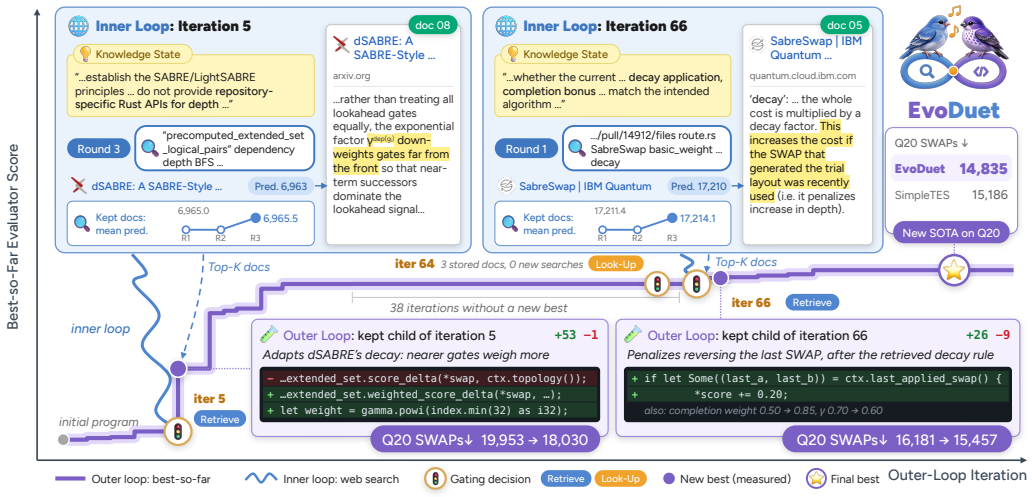


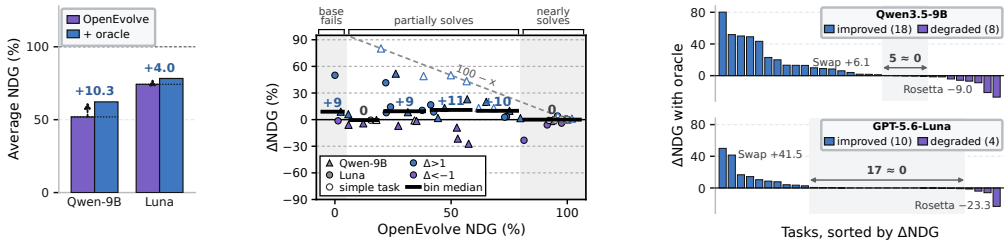
Figure 2: **EvoDuet co-evolves solutions and web queries.** The inner loop evolves queries; the outer loop uses retrieved documents to revise and evaluate solutions. The Swap Reduction example illustrates *method transfer* (the most common of the six behaviors analyzed in Section 5.1): the model applies retrieved routing principles to prioritize gates closer to execution (iteration 5) and penalize reversing the last SWAP (iteration 66). At iteration 64, the gate reuses stored documents without a new web search. The best router found in this run uses 14,835 Q20 SWAPs, 2.31% fewer than the released SimpleTES program (WILL Team, 2026) (15,186) under the same evaluator.

rather than one. Second, queries must evolve with the solution: on Denoising, an LLM that searches inside the solution loop mostly retrieves pages it has already seen (88.1% of returned URLs) and stops improving after 50 iterations. We therefore co-evolve web search queries with solutions (Figure 1), making the search **open**.

We introduce **EvoDuet**, a bi-level optimization method that co-evolves solutions in an outer loop and web search queries in an inner loop, with the LLM's parameters fixed. A knowledge-gap-based retrieval gate connects the loops: at each iteration, the LLM assesses what it still needs to know to improve the current solution (its knowledge gap) and decides whether to retrieve new documents, reuse stored ones, or proceed without documents. When it retrieves, the inner loop constructs queries targeting the gap, searches the web, and ranks documents by *hypothetical evidence scoring*, which predicts the evaluator score of a candidate built on each document. These predictions and the updated knowledge state steer the next round of queries without evaluating any candidate. Given documents, the outer loop generates candidates in parallel and records the documents with the evaluated outcome to guide later searches. Figure 2 illustrates this interplay on Swap Reduction (quantum compilation).

We evaluate EvoDuet with OpenEvolve on 21 optimization tasks. With one candidate per iteration, it raises overall NDG from 74.1% to 78.0% with GPT-5.6-Luna and from 61.3% to 82.3% with Gemini-3.8-Flash, and parallel generation improves it further. Qwen3.5-9B does not benefit, even though oracle documents help it: it often leaves retrieved methods unused or misimplemented (26% of sampled revisions), suggesting that co-evolution requires a model that can both find and apply evidence. The best runs of EvoDuet surpass the previously reported best scores on eight tasks and match them on three more; on Denoising, EvoDuet surpasses SimpleTES (WILL Team, 2026) at 6.9× lower estimated cost. It also improves other scaffolds, Top-K and EvoX, on Sums/Diffs and Denoising, showing that the search loop can be packed alongside existing loops.

Finally, we analyze how retrieved documents shape the search. Candidates improve on their parent in 67.1% of iterations that retrieve a document predicted to beat the parent, compared with 52.1% of iterations without documents. The most common use of documents is *method transfer*, in which the model applies a retrieved method to its solution (55 of 82 runs). Yet documents predicted to help can also yield much worse candidates (29 of 82 runs), so predicted gains must still be validated by the evaluator.



(a) Average NDG (↑) **(b)** Gain vs. Headroom **(c)** ΔNDG per task (↑)

Figure 3: **OpenEvolve with and without oracle documents** on 31 optimization tasks using Qwen3.5-9B and GPT-5.6-Luna. (a) Average NDG across tasks. (b) Per-task ΔNDG versus baseline NDG, with bin medians pooled across models. Shading marks baseline NDG below 5% or at least 80%; hollow markers denote simple tasks. The dashed line shows the gain needed to reach 100% NDG. (c) Per-task ΔNDG in descending order. Legends count tasks with NDG increases (improved) or decreases (degraded) of more than 1%; shading indicates changes within ±1%.

2 Analysis of Web Search in Scientific Discovery

2.1 Preliminaries: Evolutionary Search Scaffold

An optimization task $\tau = (I, x_0, \mathcal{E})$ consists of an instruction I, an initial solution x_0, and a deterministic, verifiable evaluator $\mathcal{E}(x) \in \mathbb{R}$. The goal of an evolutionary search scaffold is to find $x^\star = \arg\operatorname{opt}_{x \in \mathcal{X}} \mathcal{E}(x)$, where $\operatorname{opt} \in \{\max, \min\}$ and $\mathcal{X} = \{x_0, \ldots, x_T\}$ is the set of solutions explored within a search budget of T. The task determines the optimization direction (e.g., minimizing the c_5 bound in the Erdős minimum overlap problem). At each iteration, the evolutionary search scaffold (Romera-Paredes et al., 2024; Novikov et al., 2025; Sharma, 2025; Liu et al., 2026a) selects a parent solution x_t and its evolutionary history $\mathcal{H}_t$ from its solution database (*a.k.a.* population) $\mathcal{D}_p$ under a policy ϕ, forming the context $c_t = [\mathcal{H}_t; x_t]$. A frozen LLM $\mathcal{M}_\theta$, serving as the mutation operator, generates a candidate solution $x_{t+1} \sim \mathcal{M}_\theta(I, c_t)$, which the evaluator $\mathcal{E}$ then scores. Based on the selection policy ϕ, the candidate solution x_{t+1} is then either added to the population $\mathcal{D}_p$ or discarded. We call $x^\star$ a *discovery* if it improves upon the previous best-known solution x_{sota}, i.e., $\mathcal{E}(x^\star) > \mathcal{E}(x_{\text{sota}})$ for maximization tasks, with the inequality reversed for minimization tasks.

2.2 Experimental Setup

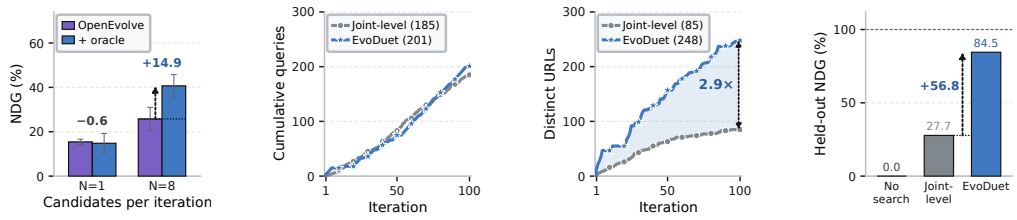
Tasks. We evaluate on 31 optimization tasks spanning simple scientific optimization (10), quantum compilation (1), astrodynamics (5), scientific algorithms (1), AI foundations (4), mathematics (8), and algorithm engineering (2). Detailed task descriptions are provided in Appendix C.1.

Baselines. We use Qwen3.5-9B (Team, 2026) and GPT-5.6-Luna (OpenAI, 2026) as the LLM $\mathcal{M}_\theta$ in OpenEvolve. For each model, we compare (1) OpenEvolve and (2) OpenEvolve with oracle documents supplied to the LLM at every iteration t. Appendix C.2 describes how we construct the oracle documents. For robustness, we run each baseline with four different random seeds.

Evaluation Metric. Since evaluator score ranges vary across tasks, we report *Normalized Discovery Gain* (NDG) to compare optimization progress on a common scale. NDG measures the percentage of the gap between the initial and SOTA scores closed by a run: $\text{NDG} = \frac{\mathcal{E}(x_{\text{best}}) - \mathcal{E}(x_0)}{\mathcal{E}(x_{\text{SOTA}}) - \mathcal{E}(x_0)} \times 100$, where x_{best} is the best-scoring solution in $\mathcal{X}$, and x_{SOTA} is the best program reported in prior work (WILL Team, 2026). Higher NDG indicates greater progress toward SOTA performance.

2.3 Does Web Search Help Scientific Discovery?

Oracle documents improve optimization on average, but their benefits vary across models and tasks. As shown in Figure 3a, oracle documents raise the average NDG of Qwen3.5-9B by 10.3% and that of GPT-5.6-Luna by 4.0%. The smaller gain for GPT-5.6-Luna likely reflects limited headroom: OpenEvolve alone already reaches at least 95% NDG on 16 of the 31 tasks. On these tasks, oracle documents change NDG by only 0.1% on average (Figure 3b). Although oracle documents lead to 100% NDG primarily on simple optimization tasks, their benefits also extend to challenging



(a) Parallel generation (↑) **(b)** Cumulative search queries **(c)** Distinct URLs returned (↑) **(d)** Held-out NDG (↑)

Figure 4: **Effects of parallel generation and query evolution with GPT-5.6-Luna.** (a) NDG of OpenEvolve on Sums/Diffs (mathematics) with and without oracle documents, using $N = 1$ or $N = 8$ candidates per iteration. (b) Cumulative number of search queries. (c) Number of distinct URLs returned. (d) Held-out NDG of the best program for each method.

tasks. For example, on Swap Reduction (quantum compilation), oracle documents increase NDG by 6.1% for Qwen3.5-9B and 41.5% for GPT-5.6-Luna (Figure 3c). However, these benefits are not universal: oracle documents lower NDG by more than 1% on 8 and 4 of the 31 tasks for Qwen3.5-9B and GPT-5.6-Luna, respectively, including decreases of 9.0% and 23.3% on Rosetta. Full results are provided in Appendix D.1.

LLMs may need more opportunities to benefit from oracle documents. The mixed effects of oracle documents raise a question: does the LLM lack the ability to use oracle documents, or does it lack sufficient opportunities to act on them? To examine this, we generate N candidates in parallel at each iteration t from the same context c_t. On Sums/Diffs, oracle documents offer no improvement at $N = 1$ (14.8% NDG with oracle documents versus 15.4% without), but increase NDG from 25.8% to 40.7% at $N = 8$ (Figure 4b). This suggests that the LLM can benefit from oracle documents when given more opportunities to explore strategies grounded in them. Additional results are provided in Appendix D.2.

Without query evolution, in-loop web search repeatedly retrieves the same pages and stalls. On Denoising with GPT-5.6-Luna, we equip OpenEvolve with Tavily [1] as an in-loop search tool (*joint-level*), allowing the LLM to construct its own queries during solution optimization. We compare this baseline with EvoDuet, which evolves queries based on the LLM's knowledge gap through bi-level co-evolution. Over 100 iterations, joint-level search uses the tool in 99 iterations and produces 185 queries, compared with 201 for EvoDuet (Figure 4b). However, it retrieves only 85 distinct URLs, compared with 248 for EvoDuet (Figure 4c); 88.1% of its returned URLs have appeared before, compared with 62.4% for EvoDuet. Moreover, joint-level search stops improving after iteration 50, with its best program achieving a held-out NDG of 27.7%, compared with 84.5% for EvoDuet (Figure 4d). These results suggest that evolving what to ask based on the LLM's knowledge gap helps it search for external knowledge more effectively, enabling it to find better solutions.

Takeaways: Web search benefits from broader exploration and query evolution

- Oracle documents increase average NDG across the 31 tasks, with a larger gain for the smaller model (+10.3% vs. +4.0%).
- More opportunities for exploration can help LLMs use oracle documents: on Sums/Diffs, oracle documents improve NDG by 14.9% with eight parallel candidates per iteration.
- Query evolution helps sustain progress: with comparable query counts on Denoising, joint-level search repeatedly retrieves the same pages and stalls, whereas bi-level search, which evolves queries based on the LLM's knowledge gap, retrieves about 2.9× as many distinct pages and increases NDG by 56.8%.

3 EvoDuet

3.1 An Overview of EvoDuet

We formulate discovery as a bi-level optimization problem with an outer loop that evolves solutions and an inner loop that evolves queries for web search. As shown in Figure 5, EvoDuet has three main components:

[1] https://www.tavily.com/

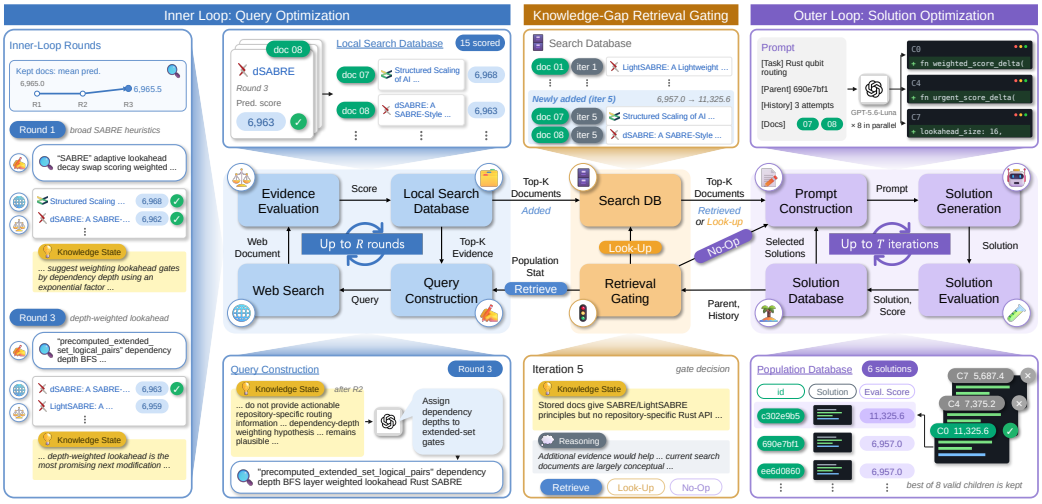


Figure 5: **Overview of EvoDuet.** The outer loop evolves solutions, while the inner loop evolves web search queries using predicted solution scores. A knowledge-gap-based retrieval gate links the two loops by deciding whether to retrieve new documents (RETRIEVE), reuse stored documents (LOOK-UP), or proceed without documents (NO-OP).

- **Solution Optimization Loop (outer, § 3.2):** The outer loop follows the scaffold's evolutionary loop (§ 2.1) to evolve solutions. At iteration t, the selection policy ϕ selects a parent solution x_t and its evolutionary history $\mathcal{H}_t$ from the population $\mathcal{D}_e$, and the LLM $\mathcal{M}_\theta$ generates a candidate solution $x_{t+1} \sim \mathcal{M}_\theta(I, c_t, \mathcal{S}_t)$, where $\mathcal{S}_t$ is either empty or a set of retrieved web documents.
- **Knowledge Gap-based Retrieval Gating (§ 3.3):** This component connects the outer and inner loops, enabling their co-evolution. Based on gaps in the LLM $\mathcal{M}_\theta$'s current knowledge status, it determines whether to invoke the inner loop for web search.
- **Query Optimization Loop (inner, § 3.4):** The inner loop evolves web search queries to retrieve documents $\mathcal{S}_t$ that provide the external knowledge needed to generate an improved candidate solution x_{t+1}.

Both loops share the task objective $\mathcal{E}$, with query quality defined by the performance of the candidate solution generated from the retrieved documents. Let $\mathcal{S}_t(q)$ denote the documents retained from query q and $x_{t+1}(q) \sim \mathcal{M}_\theta(I, c_t, \mathcal{S}_t(q))$ the resulting candidate. The ideal bi-level optimization is formulated as

$$\underbrace{x^\star = \arg\operatorname*{opt}_{x \in \mathcal{X}} \mathcal{E}(x)}_{\text{outer: solution optimization}} \quad \text{s.t.} \quad \underbrace{q_t^\star = \arg\operatorname*{opt}_{q \in \mathcal{Q}_t} \mathcal{E}\big(x_{t+1}(q)\big)}_{\text{inner: query optimization}}, \qquad x_{t+1} \sim \mathcal{M}_\theta\big(I, c_t, \mathcal{S}_t(q_t^\star)\big), \tag{1}$$

where $\mathcal{X} = \{x_0, \ldots, x_T\}$ is the set of solutions explored within budget T, and $\mathcal{Q}_t$ is the set of admissible queries at iteration t. However, the inner objective $\mathcal{E}(x_{t+1}(q))$ is observable only after the corresponding candidate has been generated and evaluated. Repeating this process for every query within the inner loop would be costly. We therefore approximate the inner optimization in Eq. 1 using $\mathcal{M}_\theta$ to **predict the candidate score** that the retrieved documents would yield. Denoting this score predictor by $\widehat{\mathcal{E}}_\theta$, the approximate inner optimization can then be reformulated as $\hat{q}_t = \arg\operatorname{opt}_{q \in \mathcal{Q}_t} \widehat{\mathcal{E}}_\theta\big(I, c_t, \mathcal{S}_t(q)\big)$. The detailed algorithm is presented in Algorithm 1.

3.2 Outer Loop: Solution Optimization

The outer loop follows the scaffold's typical evolutionary procedure (§ 2.1), but differs in the inputs provided to the LLM $\mathcal{M}_\theta$ during candidate generation. When web documents $\mathcal{S}_t$ are provided, the LLM $\mathcal{M}_\theta$ generates a candidate solution $x_{t+1} \sim \mathcal{M}_\theta(I, c_t, \mathcal{S}_t)$, which is then evaluated by $\mathcal{E}$ and retained or discarded according to ϕ. The web documents $\mathcal{S}_t$ are stored in the search database $\mathcal{D}_\text{s}$ together with the candidate's actual evaluator score $\mathcal{E}(x_{t+1})$. This stored information is used to decide whether to reuse existing documents in $\mathcal{D}_\text{s}$ through a search database $\mathcal{D}_s$ lookup, retrieve additional knowledge from the web, or skip the inner loop.

Gated parallel candidate solution generation. In Section 2, we observe that parallel scaling with rich oracle documents enables the LLM $\mathcal{M}_\theta$ to explore diverse approaches and find better solutions. Motivated by this observation, EVODUET generates N_t candidates in parallel from the same input prompt, with $N_t = N$ for LOOK-UP or RETRIEVE and $N_t = 1$ for NO-OP. Candidate generation and selection are formulated as

$$x_{t+1}^{(n)} \sim \mathcal{M}_\theta\big(I, c_t, \mathcal{S}_t\big), \quad n = 1, \ldots, N_t, \qquad x_{t+1} = \operatorname*{arg\,opt}_{x \in \mathcal{C}_t^{\text{valid}}} \mathcal{E}(x), \tag{2}$$

where $\mathcal{C}_t^{\text{valid}}$ is the set of valid candidates (i.e., those that encounter no errors, such as evaluator errors) among the N_t generated solutions. Each candidate is evaluated, and only the best valid solution is passed to ϕ as x_{t+1}.

3.3 KNOWLEDGE-GAP-BASED RETRIEVAL GATING

The gate allows $\mathcal{M}_\theta$ to determine *when* to search by assessing whether it needs new external knowledge to improve the current solution x_t, can reuse information stored in $\mathcal{D}_\text{s}$, or can evolve the solution using only its internal knowledge. At each iteration, given the context c_t and the documents most recently stored in $\mathcal{D}_\text{s}$, $\mathcal{M}_\theta$ outputs its current knowledge state K_t and a retrieval decision g_t:

$$(K_t, g_t) = \text{GATE}_{\mathcal{M}_\theta}\big(c_t, \mathcal{D}_\text{s}\big), \qquad g_t \in \{\text{NO-OP}, \text{LOOK-UP}, \text{RETRIEVE}\}. \tag{3}$$

The knowledge state K_t distinguishes the model's existing knowledge, findings from previous searches and experiments, and unresolved questions about x_t that define the current knowledge gap. The gate selects NO-OP when the model's knowledge suffices, LOOK-UP when stored documents provide the missing knowledge, and RETRIEVE when neither source is sufficient. Under LOOK-UP, the selected documents are included in the solution prompt without a new web search. This makes retrieval responsive to the current knowledge gap and enables the reuse of previously retrieved evidence. The results in Table 1b show that knowledge-gap-based gating is more effective than the heuristic gating based on stalled progress used in EvoX (Liu et al., 2026a).

3.4 INNER LOOP: QUERY OPTIMIZATION

When $g_t = \text{RETRIEVE}$, the inner loop approximates the query optimization in Eq. 1 over R inner rounds within the current outer iteration t, without generating or evaluating candidate solution.

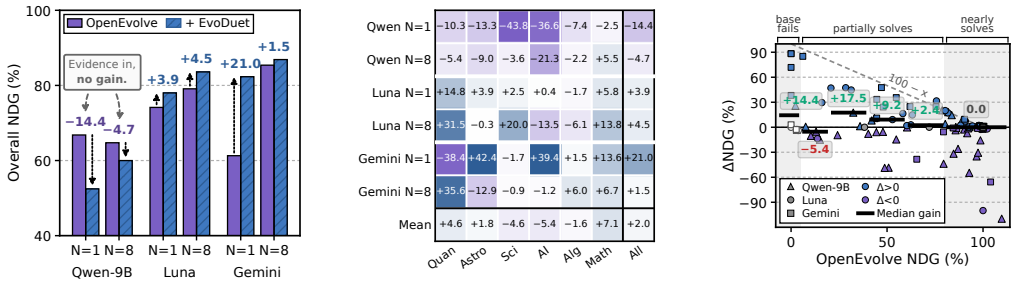
Population state descriptor. Following EvoX (Liu et al., 2026a), the inner loop first computes a population state descriptor, $\text{stats}(\mathcal{D}_p)$, comprising deterministic statistics of the retained population, its score distribution, recent parent-to-child outcomes, and parent-selection frequencies. $\mathcal{M}_\theta$ summarizes these statistics into factual observations $A_t = \mathcal{M}_\theta(\text{stats}(\mathcal{D}_p))$ without additional interpretation. These observations complete the initial inner-loop context $\tilde{c}_t^0 = (c_t, K_t, A_t)$, which the inner rounds condition on and update.

Iterative query optimization. Let $\mathcal{S}_t^r$ denote the documents retained after inner round r, and $\hat{s}_t(d)$ the predicted evaluator score of a candidate x_{t+1} obtained by improving x_t using document d alone. Starting with $\mathcal{S}_t^0 = \emptyset$ and $K_t^0 = K_t$, each round $r = 1, \ldots, R$ performs four operations:

(1) **Query Construction:** The LLM $\mathcal{M}_\theta$ constructs J queries targeting the remaining knowledge gaps. The query batch is $Q_r = (q_r^{(j)})_{j=1}^J$, with each query sampled as $q_r^{(j)} \sim \mathcal{M}_\theta(\tilde{c}_t^{r-1}, \mathcal{S}_t^{r-1}, \{\hat{s}_t(d)\}_{d \in \mathcal{S}_t^{r-1}})$. At $r = 1$, the document and score inputs are empty, so query construction uses only the initial context $\tilde{c}_t^0$. Later rounds also use the retained documents and their predicted scores.

(2) **Web Search:** Each distinct query in Q_r is executed once per round. The returned documents are combined with the previously retained documents to form the pool (i.e., local search database in Figure 5) $\mathcal{P}_r = \mathcal{S}_t^{r-1} \cup \text{SEARCH}(Q_r)$.

(3) **Hypothetical Evidence Scoring:** When unscored documents are available, a single call to $\mathcal{M}_\theta$ predicts $\hat{s}_t(d) = \mathcal{M}_\theta(\tilde{c}_t^{r-1}, \mathcal{E}(x_t), d)$ for each such $d \in \mathcal{P}_r$. These *hypothetical absolute scores* are expressed on the evaluator's scale and provide a surrogate signal for the inner objective in Eq. 1 (Figure 8a shows the model's ability to predict these scores). Each document is scored only once within the inner loop, and its score is reused in later rounds.



(a) Overall NDG (↑) **(b)** Group ΔNDG (↑) **(c)** Gain vs. Headroom

Figure 6: **Discovery performance of OpenEvolve with and without EvoDuet.** Results cover 21 paired tasks with $N = 1$ or 8 candidates per iteration. For each model, method, task, and N, we report the best NDG across four runs. **(a)** Mean best NDG across tasks and the gain from adding EvoDuet (ΔNDG). **(b)** Mean ΔNDG within each task group. **(c)** Task-level ΔNDG versus OpenEvolve's NDG; black bars and labels show the median gain within each NDG range.

(4) **Knowledge State Update:** The same call used for hypothetical evidence scoring also updates the knowledge state to K_t^r using the new documents, yielding the next round's context $\tilde{c}_t^r = (c_t, K_t^r, A_t)$. If no documents require scoring, the call is skipped and $K_t^r = K_t^{r-1}$. The loop then retains up to D documents with the highest predicted scores as $\mathcal{S}_t^r = \mathrm{TopD}_{\hat{s}_t}(\mathcal{P}_r)$.

After all R rounds, the retained Top-D documents $\mathcal{S}_t = \mathcal{S}_t^R$ are passed to the outer loop.

4 Experimental Results

EvoDuet improves discovery, but only for LLMs that can exploit retrieved evidence. At $N = 1$, Figure 6a shows that EvoDuet improves OpenEvolve's overall NDG from 74.1% to 78.0% (+3.9%) with GPT-5.6-Luna and from 61.3% to 82.3% (+21.0%) with Gemini-3.8-Flash. Qwen3.5-9B, however, shows a 14.4% decline at $N = 1$ and still loses 4.7% with parallel generation at $N = 8$, despite gaining 10.3% when given oracle documents (Figure 3a). Unlike the oracle setting, EvoDuet requires the model to decide when and what to search for based on its own knowledge gaps. This contrast suggests that these additional demands may limit the benefits of retrieval for weaker backbones such as Qwen3.5-9B. Inspection of 50 randomly sampled Qwen3.5-9B revisions with documents at $N = 1$ identifies unused methods in 6% and incorrect implementations in 20% (26% combined; Appendix E.3). These findings suggest that effective bi-level co-evolution depends on the model's ability to both retrieve and apply relevant evidence.

Parallel generation consistently improves discovery across models. Figure 6a shows that increasing the candidate budget from $N = 1$ to $N = 8$ improves EvoDuet's overall NDG for all three models. For Qwen3.5-9B, although EvoDuet remains below the corresponding OpenEvolve baseline at both budgets, its overall NDG improves from $N = 1$ to $N = 8$, indicating that parallel generation also benefits this model. We further compare $N \in \{1, 8, 16\}$ for GPT-5.6-Luna and Gemini-3.8-Flash on 8 tasks in Figure 7. We observe that the average NDG increases monotonically with N for both models, rising from 89.6% to 95.7% for GPT-5.6-Luna and from 84.5% to 89.6% for Gemini-3.8-Flash. These results suggest that broader exploration guided by web documents helps LLMs discover better solutions.

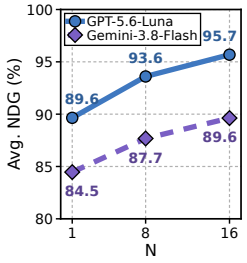
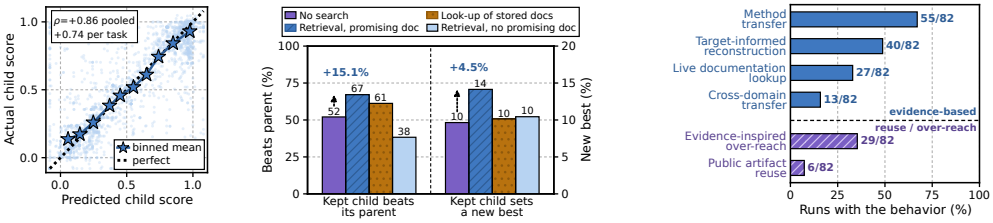


Figure 7: **Scaling effect of N.**

EvoDuet achieves its largest average gain in mathematics. As shown in Figure 6b, EvoDuet improves NDG over OpenEvolve on mathematics tasks in five of the six model/budget settings, with an average gain of 7.1% across all six. At $N = 8$, Qwen3.5-9B gains 5.5% despite its negative overall gain, and Gemini-3.8-Flash gains 6.7%.

EvoDuet discovers novel solutions across 11 different tasks. Table 1a shows that EvoDuet surpasses the previous SOTA program scores on eight tasks across five domains and matches them on three more. The mean cost across the eleven reported runs is $45.56. Interestingly, on Rosetta and Erdős, the model initially evolves solutions, then retrieves public constructions at a later iter-



(a) Predicted vs. evaluated scores **(b)** Iteration improvements **(c)** Six patterns of document use

Figure 8: **Program improvement with web documents and patterns of document use.** Results cover 8,200 iterations from 82 GPT-5.6-Luna runs across 21 tasks. **(a)** Predicted versus evaluated candidate scores, scaled to $[0, 1]$ using each task's 2nd and 98th percentiles. **(b)** Rates at which the selected candidate improves on its parent (left axis) or sets a new best score within the run (right axis), grouped by gate decision. Arrows compare NO-OP iterations with RETRIEVE iterations that retain at least one promising document. **(c)** Percentage of runs exhibiting each behavior. Each run is counted at most once per category and may appear in multiple categories.

ation and uses them as starting points for further optimization: refining a published trajectory and locally optimizing a published construction ($C_5 : 0.380859 \rightarrow 0.380859$), respectively. We call this behavior *public artifact reuse*, one of six behaviors reported in Appendix E.1. This behavior occurs in only 6 of 82 GPT-5.6-Luna runs (7.3%), compared with *method transfer* (55 of 82), in which the model applies principles or methods retrieved by searching relevant literature (Figure 8c). Only two of these 11 SOTA results reuse public artifacts in their best programs (in Appendix K).

The largest median gain occurs on partially solved tasks. Figure 6c compares OpenEvolve and EvoDuet across 126 combinations of tasks, models, and candidate counts. When OpenEvolve's NDG is 5–20%, 20–40%, 40–60%, and 60–80%, the median gains from EvoDuet are -5.4%, +17.5%, +9.2%, and +2.4%, respectively. When OpenEvolve already performs well (NDG $\geq$80%), the median gain is 0.0% across 75 comparisons. Of the 52 comparisons with OpenEvolve NDG $\geq$95%, six show declines greater than 5%. When OpenEvolve makes little progress (NDG below 5%), EvoDuet raises five of the ten cases above 5%, with a median gain of +71.8% among these five. In the remaining five cases, both methods stay near the initial program's performance. Overall, EvoDuet tends to provide smaller gains when OpenEvolve's NDG is higher (Pearson $r = -0.41$).

5 Analysis and Discussions

5.1 How Does Web Search Help?

Hypothetical evidence scores track actual program performance. Across 4,324 retrievals, hypothetical evidence scores correlate strongly with the selected candidates' evaluated scores (Spearman $\rho = +0.86$; Figure 8a). Correlations are positive on all 21 tasks, with a median of $+0.74$. This supports using these scores as a surrogate signal for inner-loop query optimization (§ 3.4).

Programs improve more often with promising documents. A document is *promising* when its hypothetical evidence score exceeds the parent's actual evaluator score, i.e., $\hat{s}_t(d) > \mathcal{E}(x_t)$. When RETRIEVE retains at least one such document, the selected candidate improves on its parent in 67.1% of iterations and sets a new run best in 14.1%, compared with 52.1% and 9.6% under NO-OP (Figure 8b). Under LOOK-UP, which reuses stored documents, these rates are 61.2% and 10.1%, respectively. On 17 of 21 tasks, parent improvement rates are higher with promising documents than under NO-OP (medians: 69% vs. 54%). However, 36.5% of retrievals yield no promising document; the parent improvement rate in these cases is only 38.3%. These results suggest that hypothetical evidence scoring helps identify documents useful for improving the current solution.

Web documents mainly provide methods and performance targets. Figure 8c summarizes six behavior categories across 82 GPT-5.6-Luna runs (8,200 iterations), with each run counted once in every applicable category. The most common behaviors are applying principles and methods from retrieved documents (55 runs) and using published best scores as reference targets (40 runs). These targets guide program development, while the task evaluator measures the resulting programs' performance. We also observe reuse of published solutions (6 runs) and unsuccessful revisions inspired by retrieved ideas (29 runs). Detailed explanations are presented in Appendix E.1.

Table 1: **Additional analyses of EvoDuet.** For (b)-(e), we use GPT-5.6-Luna.

(a) Best programs of EvoDuet.

Task	Prev. SOTA	EvoDuet	Cost (USD)
Swap Reduction (↓)	15,186	**14,835**	50.90
Rosetta (↓)	1.552968	**1.396424**	44.12
Voyager 2 (↓)	3.430214	**3.430206**	43.02
Denoising (↑)	0.722690	**0.722906**	38.45
Domain mix. (↑)	0.996922	**0.997062**	15.84
Parallel (↑)	0.999970	**0.999975**	11.93
Erdős (↓)	0.380868	**0.380859**	29.55
Hadamard (↑)	**0.935673**	**0.935673**	27.33
Sums/Diffs (↑)	1.144887	**1.144999**	161.33
CP (n=26) (↑)	**2.635983**	**2.635983**	36.33
CP (n=32) (↑)	**2.939573**	**2.939573**	42.32

(b) Retrieval gating.

Gating	Denoising	Erdős
Random	58.5%	93.9%
Heuristic	60.6%	99.5%
Knowledge Gap	**84.5%**	**100.0%**

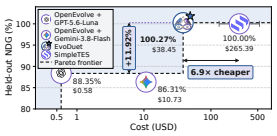
(c) Search-method comparison

Method	Molecule (↑)	Burgers (↑)	CP (n=26)
OpenEvolve	0.8496	0.6937	**2.635983**
Joint-level	0.8474	0.6919	2.635980
DeepEvolve	0.8149	0.6666	2.581971
EvoDuet	**0.8524**	**0.7846**	**2.635983**

(d) Cross-scaffold ΔNDG.

Scaffold	Sums/Diffs	Denoising
OpenEvolve	+23.5%	+84.5%
Top-K	+46.7%	+63.6%
EvoX	+55.1%	+37.8%

(e) Denoising Pareto frontier.



5.2 Additional Analysis

The knowledge-gap-based gate achieves the highest NDG. Table 1b compares random gating ($p = 0.5$), a stagnation heuristic (Liu et al., 2026a), and our knowledge-gap-based gate. Our gate performs best on both Denoising and Erdős, with a particularly large gain of 23.9% over the stagnation heuristic on Denoising. This highlights the importance of the model's assessment of its own knowledge state when deciding whether to search the web.

Bi-level optimization is an effective way to couple web search with solution evolution. We compare four variants: OpenEvolve without web search, OpenEvolve with an in-loop search tool (joint-level), DeepEvolve (sequential), and EvoDuet (bi-level). A detailed comparison of these variants is provided in Appendix B.5. In Table 1c, EvoDuet outperforms all other methods on three tasks, highlighting the importance of the bi-level design.

EvoDuet integrates effectively with existing scaffolds. Table 1d compares OpenEvolve, Top-K, and EvoX with and without EvoDuet. EvoDuet improves NDG on Sums/Diffs and Denoising across all three scaffolds, with respective gains of 23.5 and 84.5 for OpenEvolve, 46.7 and 63.6 for Top-K, and 55.1 and 37.8 for EvoX. Although EvoX already includes a solution loop and a meta-level strategy loop, adding EvoDuet's web search loop yields substantial gains, highlighting the benefits of *packing the loop* (Figure 1).

Cost efficiency. Table 1e compares SimpleTES (WILL Team, 2026), the best observed OpenEvolve runs, and EvoDuet on Denoising. EvoDuet achieves 100.27% NDG at $38.45, a 6.9× reduction in cost compared with SimpleTES (100% NDG at an estimated API-equivalent cost of $265.39). These results suggest that bi-level co-evolution enables cost-efficient performance by retrieving relevant external knowledge when the model needs it, despite the additional cost of web search.

6 Related Work

LLM-driven evolutionary scaffolds use prior candidates and evaluation feedback to improve programs (Novikov et al., 2025; Sharma, 2025; Lange et al., 2025; Wang et al., 2025; Qu et al., 2026; Yuksekgonul et al., 2026; Lee et al., 2026b). GEPA (Agrawal et al., 2025) optimizes prompts, while other methods evolve agent or harness code (Zhang et al., 2025; 2026; Lee et al., 2026a). Most of these methods operate within a single solution loop. EvoX (Liu et al., 2026a) adds a meta-level loop that evolves search strategies alongside solutions, but both loops draw on the LLM's existing knowledge and the run's evolutionary history. DeepEvolve (Liu et al., 2025) adds a web search loop, but couples search and solution optimization sequentially rather than through a bi-level formulation.

7 Conclusion

We introduce EvoDuet, a bi-level method that co-evolves solutions and web search queries, letting the LLM's knowledge state decide *when* to search and *what* to ask. Across 21 tasks, EvoDuet improves OpenEvolve with GPT-5.6-Luna and Gemini-3.8-Flash (but not Qwen3.5-9B), surpasses the best previously reported scores on eight tasks, and improves Top-K and EvoX on Sums/Diffs and Denoising.

AI USE STATEMENT

In this work, we used generative AI tools for trajectory analysis. We have not used generative AI tools for idea proposal or method development, and proof-related tasks are not applicable to this work. Additionally, we used generative AI tools for paper writing and figure preparation. We also used generative AI tools to retrieve and summarize oracle documents, as described in Appendix C.2.

We have reviewed all AI-assisted work. We checked the AI-assisted text and figures for consistency with the reported experimental results. For trajectory analysis, we checked the interpretations against recorded search queries, retrieved documents, parent and child programs, and evaluator outputs, with worked cases provided in Appendix I. We also inspected cases of public artifact reuse and documented the sources of reused solutions, as detailed in Appendices E.1 and K.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work evaluates LLM-guided computational optimization across a set of tasks, using publicly available web sources to inform the search. To support proper attribution, we document cases of public artifact reuse, including the original sources and subsequent optimization steps, in Appendices E.1 and K. The reported gains reflect performance under the task evaluators used in this study. The resulting programs require independent validation before use in other settings. Appendix H discusses source attribution, the reliability of retrieved information, program validation, and human oversight of downstream applications.

REPRODUCIBILITY STATEMENT

We describe EVODUET in Section 3 and provide pseudocode, inference settings, and retrieval budgets in Appendix B. Task definitions, evaluation procedures, and oracle-document construction are documented in Appendix C, and the prompt templates are provided in Appendix J. Detailed experimental results appear in Appendix D. Appendix I links selected search trajectories to retrieved sources, code revisions, and evaluation results. Appendix K provides complete source listings and identifies the runs underlying Table 1a. Detailed trajectories, prompt templates, and complete program listings are provided in the supplementary materials (Appendices I–K).

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning. *arXiv preprint arXiv:2507.19457*, 2025. URL https://arxiv.org/abs/2507.19457.

APMonitor. Batch reactor optimization. APMonitor Documentation, 2025. URL https://www.apmonitor.com/wiki/index.php/Apps/BatchReactor. Last modified March 25, 2025. Accessed September 9, 2026.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*, 2023.

Henrique Assumpção, Diego Ferreira, Leandro Campos, and Fabricio Murai. CodeEvolve: an open source evolutionary coding agent for algorithmic discovery and optimization. *arXiv preprint arXiv:2510.14150*, 2025. URL https://arxiv.org/abs/2510.14150.

Lorenz M. Baumgartner, Joseph M. Dennis, Nicholas A. White, Stephen L. Buchwald, and Klavs F. Jensen. Use of a droplet platform to optimize Pd-catalyzed C–N coupling reactions promoted by organic bases. *Organic Process Research & Development*, 23(8):1594–1601, 2019. doi: 10.1021/acs.oprd.9b00236. URL https://doi.org/10.1021/acs.oprd.9b00236.

Daniil A. Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624:570–578, 2023. doi: 10.1038/s41586-023-06792-0. URL https://www.nature.com/articles/s41586-023-06792-0.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6:525–535, 2024. doi: 10.1038/s42256-024-00832-8. URL https://www.nature.com/articles/s42256-024-00832-8.

Nathan Brown, Marco Fiscato, Marwin H. S. Segler, and Alain C. Vaucher. GuacaMol: Benchmarking models for de novo molecular design. *Journal of Chemical Information and Modeling*, 59(3):1096–1108, 2019. doi: 10.1021/acs.jcim.8b00839. URL https://doi.org/10.1021/acs.jcim.8b00839.

Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016. doi: 10.1073/pnas.1517384113. URL https://doi.org/10.1073/pnas.1517384113.

Cambridge Cluster Database. Table of Lennard-Jones cluster global minima, n.d. URL https://www-wales.ch.cam.ac.uk/~jon/structures/LJ/tables.150.html. University of Cambridge. Accessed September 9, 2026.

Mert Cemri, Shubham Agrawal, Akshat Gupta, Shu Liu, Audrey Cheng, Qiuyang Mang, Ashwin Naren, Lutfi Eren Erdogan, Koushik Sen, Matei Zaharia, et al. Adaevolve: Adaptive llm driven zeroth-order optimization. *arXiv preprint arXiv:2602.20133*, 2026.

Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. RQ-RAG: Learning to Refine Queries for Retrieval Augmented Generation. *arXiv preprint arXiv:2404.00610*, 2024. URL https://arxiv.org/abs/2404.00610.

Haodong Chen, Shuai Wang, Yu Yin, Shengyao Zhuang, Guido Zuccon, and Teerapong Leelanupab. ITER: Interaction-Aware Retrieval for Agentic Search. *arXiv preprint arXiv:2608.27912*, 2026. URL https://arxiv.org/abs/2608.27912.

Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. ReSearch: Learning to Reason with Search for LLMs via Reinforcement Learning. *arXiv preprint arXiv:2503.19470*, 2025. URL https://arxiv.org/abs/2503.19470.

He Du, Qiming Ge, Jiakai Hu, Aijun Yang, Zheng Cai, Zixian Huang, Sheng Yuan, Qinxiu Cheng, Xinchen Xie, Yicheng Chen, et al. Kernel-Smith: A Unified Recipe for Evolutionary Kernel Optimization. *arXiv preprint arXiv:2603.28342*, 2026. URL https://arxiv.org/abs/2603.28342.

Paul Erdős. Some remarks on number theory. *Riveon Lematematika*, 9:45–48, 1955.

Kobi C. Felton, Jan G. Rittig, and Alexei A. Lapkin. Summit: Benchmarking machine learning methods for reaction optimisation. *Chemistry–Methods*, 1(2):116–122, 2021. doi: 10.1002/cmtd.202000051. URL https://doi.org/10.1002/cmtd.202000051.

Ali Essam Ghareeb, Benjamin Chang, Ludovico Mitchener, Angela Yiu, Caralyn J Szostkiewicz, Dmytro Shved, Gavin J Gyimesi, Jon M Laurent, Samantha M Wright, Muhammed T Razzak, et al. A multi-agent system for automating scientific discovery. *Nature*, 655:497–505, 2026. doi: 10.1038/s41586-026-10652-y. URL https://www.nature.com/articles/s41586-026-10652-y.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Petar Sirkovic, Artiom Myaskovsky, Grzegorz Glowaty, Felix Weissenberger, Alessio Orlandi, Dan Popovici, et al. Accelerating scientific discovery with Co-Scientist. *arXiv preprint arXiv:2502.18864*, 2025.

Florian Häse, Matteo Aldeghi, Riley J. Hickman, Loïc M. Roch, and Alán Aspuru-Guzik. Gryffin: An algorithm for Bayesian optimization of categorical variables informed by expert knowledge. *Applied Physics Reviews*, 8:031406, 2021a. doi: 10.1063/5.0048164. URL https://doi.org/10.1063/5.0048164.

Florian Häse, Matteo Aldeghi, Riley J. Hickman, Loïc M. Roch, Melodie Christensen, Elena Liles, Jason E. Hein, and Alán Aspuru-Guzik. Olympus: A benchmarking framework for noisy optimization and experiment planning. *Machine Learning: Science and Technology*, 2:035021, 2021b. doi: 10.1088/2632-2153/abedc8. URL https://doi.org/10.1088/2632-2153/abedc8.

Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, Ryan Li, Yusuf Roohani, Lin Qiu, Shiyi Cao, Gavin Li, et al. Autonomous biomedical research with an artificial intelligence agent. *Science*, 393(6813):eadz4351, 2026. doi: 10.1126/science.adz4351. URL https://www.science.org/doi/10.1126/science.adz4351.

Yuki Imajuku, Kohki Horie, Yoichi Iwata, Kensho Aoki, Naohiro Takahashi, and Takuya Akiba. Ale-bench: A benchmark for long-horizon objective-driven algorithm engineering. *arXiv preprint arXiv:2506.09050*, 2025.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. *arXiv preprint arXiv:2403.14403*, 2024.

Qile Jiang and George Karniadakis. Agenticsciml: Collaborative multi-agent systems for emergent discovery in scientific machine learning. *npj Artificial Intelligence*, 2:57, 2026. doi: 10.1038/s44387-026-00102-5. URL https://www.nature.com/articles/s44387-026-00102-5.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983*, 2023.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. Shinkaevolve: Towards open-ended and sample-efficient program evolution. *arXiv preprint arXiv:2509.19349*, 2025.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-End Optimization of Model Harnesses. *arXiv preprint arXiv:2603.28052*, 2026a. URL https://arxiv.org/abs/2603.28052.

Young-Jun Lee, Seungone Kim, Minki Kang, Alistair Cheong Liang Chuen, Zerui Chen, Seungho Han, Taehee Jung, and Dongyeop Kang. Evolution fine-tuning: Learning to discover across 371 optimization tasks. *arXiv preprint arXiv:2606.29082*, 2026b.

Gang Liao, Hongsen Qin, Ying Wang, Alicia Golden, Michael Kuchnik, Yavuz Yetim, Jia Jiunn Ang, Chunli Fu, Yihan He, Samuel Hsia, et al. KernelEvolve: Scaling Agentic Kernel Coding for Heterogeneous AI Accelerators at Meta. *arXiv preprint arXiv:2512.23236*, 2025. URL https://arxiv.org/abs/2512.23236.

Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of Heuristics: Towards Efficient Automatic Algorithm Design Using Large Language Model. *arXiv preprint arXiv:2401.02051*, 2024. URL https://arxiv.org/abs/2401.02051.

Gang Liu, Yihan Zhu, Jie Chen, and Meng Jiang. Scientific algorithm discovery by augmenting AlphaEvolve with deep research. *arXiv preprint arXiv:2510.06056*, 2025.

Shu Liu, Shubham Agarwal, Monishwaran Maheswaran, Mert Cemri, Zhifei Li, Qiuyang Mang, Ashwin Naren, Ethan Boneh, Audrey Cheng, Melissa Z Pan, et al. Evox: Meta-evolution for automated discovery. *arXiv preprint arXiv:2602.23413*, 2026a.

Shu Liu, Mert Cemri, Shubham Agarwal, Alexander Krentsel, Ashwin Naren, Qiuyang Mang, Zhifei Li, Akshat Gupta, Monishwaran Maheswaran, Audrey Cheng, et al. SkyDiscover: A Flexible, Adaptive Framework for AI-Driven Scientific and Algorithmic Discovery, 2026b. URL https://skydiscover-ai.github.io/blog.html. Official project documentation.

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

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting for retrieval-augmented large language models. *arXiv preprint arXiv:2305.14283*, 2023.

Yoshitomo Matsubara, Naoya Chiba, Ryo Igarashi, and Yoshitaka Ushiku. Rethinking symbolic regression datasets and benchmarks for scientific discovery. *Journal of Data-centric Machine Learning Research*, 1(3):1–38, 2024. URL https://data.mlr.press/assets/pdf/v01-3.pdf.

National Institute of Standards and Technology. StRD dataset Chwirut2. Statistical Reference Datasets, n.d. URL https://www.itl.nist.gov/div898/strd/nls/data/chwirut2.shtml. Accessed September 9, 2026.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition, July 2026. URL https://openai.com/index/gpt-5-6/. Accessed: 2026-09-24.

Anne Ouyang, Simon Guo, Simran Arora, Alex L Zhang, William Hu, Christopher Ré, and Azalia Mirhoseini. Kernelbench: Can llms write efficient gpu kernels? *arXiv preprint arXiv:2502.10517*, 2025.

Tom Packebusch and Stephan Mertens. Low autocorrelation binary sequences. *Journal of Physics A: Mathematical and Theoretical*, 49(16):165001, 2016. doi: 10.1088/1751-8113/49/16/165001. URL https://arxiv.org/abs/1512.02475.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic Prompt Optimization with "Gradient Descent" and Beam Search. *arXiv preprint arXiv:2305.03495*, 2023. URL https://arxiv.org/abs/2305.03495.

Ao Qu, Han Zheng, Zijian Zhou, Yihao Yan, Yihong Tang, Shao Yong Ong, Fenglu Hong, Kaichen Zhou, Chonghe Jiang, Minwei Kong, et al. CORAL: Towards Autonomous Multi-Agent Evolution for Open-Ended Discovery. *arXiv preprint arXiv:2604.01658*, 2026. URL https://arxiv.org/abs/2604.01658.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Samuel Schmidgall and Michael Moor. AgentRxiv: Towards Collaborative Autonomous Research. *arXiv preprint arXiv:2503.18102*, 2025. URL https://arxiv.org/html/2503.18102.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. Agent Laboratory: Using LLM Agents as Research Assistants. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 5977–6043, 2025. doi: 10.18653/v1/2025.findings-emnlp.320. URL https://aclanthology.org/2025.findings-emnlp.320/.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Enhancing Retrieval-Augmented Large Language Models with Iterative Retrieval-Generation Synergy. *arXiv preprint arXiv:2305.15294*, 2023. URL https://arxiv.org/abs/2305.15294.

Asankhaya Sharma. OpenEvolve: An open-source evolutionary coding agent. https://github.com/algorithmicsuperintelligence/openevolve, 2025. GitHub repository.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. REPLUG: Retrieval-Augmented Black-Box Language Models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8371–8384, 2024. doi: 10.18653/v1/2024.naacl-long.463. URL https://aclanthology.org/2024.naacl-long.463/.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-Searcher: Incentivizing the Search Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2503.05592*, 2025. URL https://arxiv.org/abs/2503.05592.

Jiabin Tang, Lianghao Xia, Zhonghang Li, and Chao Huang. AI-Researcher: Autonomous Scientific Innovation. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0320. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/0d904d300a105809a2114d727851e759-Abstract-Conference.html.

Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10014–10037, 2023. doi: 10.18653/v1/2023.acl-long.557. URL https://aclanthology.org/2023.acl-long.557/.

Yiping Wang, Shao-Rong Su, Zhiyuan Zeng, Eva Xu, Liliang Ren, Xinyu Yang, Zeyi Huang, Xuehai He, Luyao Ma, Baolin Peng, et al. Thetaevolve: Test-time learning on open problems. *arXiv preprint arXiv:2511.23473*, 2025.

WILL Team. Evaluation-driven scaling for scientific discovery. *arXiv preprint arXiv:2604.19341*, 2026. URL https://arxiv.org/abs/2604.19341.

Minghao Yan, Bo Peng, Benjamin Coleman, Ziqi Chen, Zhouhang Xie, Shuo Chen, Zhankui He, Noveen Sachdeva, Isabella Ye, Weili Wang, et al. PACEvolve: Enabling Long-Horizon Progress-Aware Consistent Evolution. *arXiv preprint arXiv:2601.10657*, 2026. URL https://arxiv.org/abs/2601.10657.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective Retrieval Augmented Generation. *arXiv preprint arXiv:2401.15884*, 2024. URL https://arxiv.org/abs/2401.15884.

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

Xin-She Yang, Christian Huyck, Mehmet Karamanoglu, and Nawaz Khan. True global optimality of the pressure vessel design problem: A benchmark for bio-inspired optimisation algorithms. *International Journal of Bio-Inspired Computation*, 5(6):329–335, 2013. doi: 10.1504/IJBIC.2013.058910. URL https://arxiv.org/abs/1403.7793.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Tian Yu, Shaolei Zhang, and Yang Feng. Auto-RAG: Autonomous Retrieval-Augmented Generation for Large Language Models. *arXiv preprint arXiv:2411.19443*, 2024. URL https://arxiv.org/abs/2411.19443.

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

Mert Yuksekgonul, Daniel Koceja, Xinhao Li, Federico Bianchi, Jed McCaleb, Xiaolong Wang, Jan Kautz, Yejin Choi, James Zou, Carlos Guestrin, et al. Learning to discover at test time. *arXiv preprint arXiv:2601.16175*, 2026.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Godel Machine: Open-Ended Evolution of Self-Improving Agents. *arXiv preprint arXiv:2505.22954*, 2025. URL https://arxiv.org/abs/2505.22954.

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

APPENDIX CONTENTS

A Extended Related Work **17**

B Method Details **18**
B.1 Outer Loop: Solution Optimization . . . 18
B.2 Knowledge-Gap-Based Retrieval Gating . . . 18
B.3 Inner Loop: Query Optimization . . . 19
B.4 Inference Settings and Call Budget . . . 20
B.5 Coupling Web Search with Solution Evolution . . . 21

C Experimental Setup Details **22**
C.1 Task Descriptions . . . 22
C.2 Oracle Document Construction . . . 27

D Additional Experimental Results **28**
D.1 OpenEvolve with and without Oracle Documents . . . 28
D.2 OpenEvolve with and without Parallel Generation . . . 29
D.3 Inner-loop Budget Ablations . . . 30
D.4 Native Scores . . . 30

E Behavior and Search Analysis **35**
E.1 Behavior Flags and Additional Examples . . . 35
E.2 Language, Resources, and Reuse in Web Search . . . 37
E.3 Failure Analysis of Retrieved Evidence Use . . . 43

F Limitations **44**

G Discussions **44**

H Broader Impact **44**

I Trajectories of the Bi-Level Loop **45**
I.1 Swap Reduction: depth weighting persists despite repeated search misses . . . 46
I.2 Erdős minimum overlap: a radar codeword seeds later improvements . . . 52
I.3 Voyager 2: documentation lookup repeats defaults already in the prompt . . . 58
I.4 Circle packing (n=26): a generated solver exceeds the queried target . . . 61
I.5 Erdős minimum overlap, second run: runtime witness reuse improves the best . . . 65
I.6 Galileo: a promising retrieval produces a worse child . . . 69
I.7 Rosetta: published seeds and implementation details support a better tour . . . 71

J Prompt Templates **76**

K Best Programs of EVODUET **83**
K.1 Swap Reduction: short lookahead reduces routing cost on Q20 . . . 83
K.2 Rosetta: refining a published tour lowers maneuver cost . . . 95
K.3 Voyager 2: constrained refinement improves the existing tour . . . 108
K.4 Denoising: mixing diffusion operators improves average accuracy . . . 116
K.5 Domain Mixture: separate domain fits improve held-out prediction . . . 120
K.6 Parallel Scaling: a fixed basis supports accurate extrapolation . . . 121
K.7 Erdős: public witnesses support a small numerical refinement . . . 122
K.8 Hadamard: structured constructions recover the reference determinant . . . 129
K.9 Sums/Diffs: construction sweeps and local search improve the reference . . . 135
K.10 Circle Packing ($n = 26$): joint refinement matches the reference . . . 142
K.11 Circle Packing ($n = 32$): multiple layouts recover the reference score . . . 147
K.12 Discovered Objects . . . 153

A Extended Related Work

This appendix expands the comparisons in § 6, focusing on what discovery methods optimize and how they acquire external knowledge.

Evolving solutions and search strategies. LLM-driven discovery methods differ in whether they optimize candidate programs, search procedures, prompts, or model parameters. FunSearch (Romera-Paredes et al., 2024), AlphaEvolve (Novikov et al., 2025), OpenEvolve (Sharma, 2025), and CodeEvolve (Assumpção et al., 2025) use execution feedback to evolve programs. EoH (Liu et al., 2024) jointly evolves heuristic descriptions and code. At the search level, ShinkaEvolve (Lange et al., 2025), PACEvolve (Yan et al., 2026), and AdaEvolve (Cemri et al., 2026) adapt selection, context management, and exploration, respectively, while EvoX (Liu et al., 2026a) jointly evolves solutions and search strategies. The Darwin Gödel Machine (Zhang et al., 2025), Hyperagents (Zhang et al., 2026), and Meta-Harness (Lee et al., 2026a) optimize agent or harness code. CORAL (Qu et al., 2026) coordinates collaborative evolution through shared memory, and SkyDiscover (Liu et al., 2026b) provides modular discovery infrastructure. OPRO (Yang et al., 2023), APO (Pryzant et al., 2023), TextGrad (Yuksekgonul et al., 2024), and GEPA (Agrawal et al., 2025) optimize prompts or textual variables using performance feedback or textual critiques. ThetaEvolve (Wang et al., 2025) and TTT-Discover (Yuksekgonul et al., 2026) update model parameters through reinforcement learning during search, while Kernel-Smith (Du et al., 2026) and Evolution Fine-Tuning (Lee et al., 2026b) train on evolutionary trajectories.

Web-augmented scientific discovery. DeepEvolve (Liu et al., 2025) sequentially couples deep research with program revision and evaluation, using prior evaluation outcomes to guide subsequent research. KernelEvolve (Liao et al., 2025) uses runtime diagnostics to guide retrieval for kernel optimization. AI co-scientist (Gottweis et al., 2025) integrates literature search into hypothesis generation and scientific review. Robin (Ghareeb et al., 2026) uses literature search to propose and assess candidate hypotheses, then incorporates experimental feedback into subsequent refinement. The AI Scientist (Lu et al., 2026), AI-Researcher (Tang et al., 2025), and Agent Laboratory (Schmidgall et al., 2025) connect literature search, computational experiments, and reporting. AgentRxiv (Schmidgall & Moor, 2025) and AgenticSciML (Jiang & Karniadakis, 2026) reuse accumulated research reports or method memories. Coscientist (Boiko et al., 2023), ChemCrow (Bran et al., 2024), and Biomni (Huang et al., 2026) combine external information with scientific tools.

Retrieval control and query refinement. Retrieval methods address when to search and how to refine queries. ReAct (Yao et al., 2023), IRCoT (Trivedi et al., 2023), Iter-RetGen (Shao et al., 2023), and Auto-RAG (Yu et al., 2024) guide successive retrieval steps using intermediate reasoning, answers, or observations. Self-RAG (Asai et al., 2023), FLARE (Jiang et al., 2023), Adaptive-RAG (Jeong et al., 2024), and CRAG (Yan et al., 2024) base retrieval decisions on reflection, uncertainty, question complexity, and evidence quality, respectively. Rewrite-Retrieve-Read (Ma et al., 2023) learns query reformulation, while RQ-RAG (Chan et al., 2024) learns rewriting, decomposition, and disambiguation. Search-R1 (Jin et al., 2025), R1-Searcher (Song et al., 2025), and ReSearch (Chen et al., 2025) learn search policies from outcome rewards. REPLUG (Shi et al., 2024) uses feedback from a frozen LM to train a retriever, while ITER (Chen et al., 2026) trains a retriever conditioned on reasoning and search histories. EvoDuet couples solution and query evolution through a shared task objective. Its knowledge-state gate chooses whether to retrieve new documents, reuse stored documents, or skip retrieval. The inner loop selects evidence using predicted candidate scores and refines queries without generating or evaluating programs. The outer loop records measured outcomes to inform subsequent retrieval decisions. Model parameters and prompt templates remain fixed throughout.

Algorithm 1 EvoDuet: overall procedure.

Input: Task $(I, x_0, \mathcal{E})$, selection policy ϕ, outer iterations T
Input: Candidate count N, retrieval settings (R, J, M, D, K)
$\mathcal{D}_e \leftarrow \{(x_0, \mathcal{E}(x_0))\}$; $\mathcal{D}_s \leftarrow \emptyset$
for $t = 0, \ldots, T-1$ **do** ▷ Outer loop: solution optimization
  $(x_t, \mathcal{H}_t) \sim \phi(\mathcal{D}_e)$; $c_t \leftarrow [\mathcal{H}_t; x_t]$
  $(K_t, g_t) \leftarrow \textsc{Gate}_{\mathcal{M}_\theta}(c_t, \mathcal{D}_s)$ ▷ Retrieval gating
  $\mathcal{S}_t \leftarrow \emptyset$ ▷ NO-OP uses no documents
  if $g_t =$ LOOK-UP **then**
    $\mathcal{S}_t \leftarrow$ stored documents identified by the gate
  else if $g_t =$ RETRIEVE **then**
    $\mathcal{S}_t \leftarrow \textsc{OptimizeQueries}(c_t, K_t, \mathcal{D}_e, \mathcal{E}(x_t))$ ▷ Inner loop: query optimization
  end if
  $N_t \leftarrow N$ if $g_t \in \{$LOOK-UP, RETRIEVE$\}$, otherwise 1
  Generate $x_{t+1}^{(n)} \sim \mathcal{M}_\theta(I, c_t, \mathcal{S}_t)$, $n = 1, \ldots, N_t$, in parallel
  Evaluate candidates with $\mathcal{E}$ and collect valid candidates in $\mathcal{C}_t^{\text{valid}}$
  if $\mathcal{C}_t^{\text{valid}} \neq \emptyset$ **then**
    $x_{t+1} \leftarrow \arg\operatorname{opt}_{x \in \mathcal{C}_t^{\text{valid}}} \mathcal{E}(x)$
    $\mathcal{D}_e \leftarrow \phi(\mathcal{D}_e \cup \{(x_{t+1}, \mathcal{E}(x_{t+1}))\})$
    Update $\mathcal{D}_s$ with evidence and the measured outcome if $g_t \neq$ NO-OP
  else
    Apply the scaffold's retry policy
  end if
end for
return best evaluated solution

B Method Details

Algorithm 1 presents the overall EvoDuet procedure, and Algorithm 2 details its query optimization loop. The inner loop refines queries using predicted candidate scores; the outer loop evaluates solutions and records their actual scores. All model calls use the same frozen $\mathcal{M}_\theta$.

B.1 Outer Loop: Solution Optimization

Candidate generation and selection. The gate sets $N_t = 1$ for NO-OP and $N_t = N$ for LOOK-UP or RETRIEVE, including retrievals that return no usable documents. Candidates are sampled independently in parallel from the same prompt $(I, c_t, \mathcal{S}_t)$ and evaluated after successful generation and parsing. Only the best valid candidate under the task's optimization direction is passed to ϕ; ties favor the earlier candidate. If all candidates fail, the scaffold applies its existing retry policy.

Search database update. The search database $\mathcal{D}_s$ associates the selected candidate's outcome with the documents it received. A record contains the executed queries, retained documents and their predicted scores, document identifiers, actual parent and candidate scores $\mathcal{E}(x_t)$ and $\mathcal{E}(x_{t+1})$, and evaluator feedback. Each prediction assumes one document alone, whereas the candidate uses the retained set jointly; the measured outcome therefore belongs to that complete attempt. Intermediate rounds receive no separate measured outcome. Failed evaluations leave the outcome unknown. Candidate code, responses, evaluations, and intermediate retrieval results are retained in the iteration trace.

B.2 Knowledge-Gap-Based Retrieval Gating

Gate inputs and outputs. A single call reads the parent solution, the scaffold's evolutionary history, and recent search records. It returns the knowledge state K_t, a decision g_t, and document identifiers when $g_t =$ LOOK-UP. The knowledge state distinguishes prior model knowledge, findings from searches and evaluations, and unresolved questions about the current solution. The gate runs before population analysis, so NO-OP and LOOK-UP require no additional analysis call.

Stored search context. The gate, query construction, and evidence scoring share a snapshot of the ten most recent records in $\mathcal{D}_s$, fixed throughout the outer iteration. Each record shows its queries,

Algorithm 2 Inner loop: query optimization.

Input: R rounds, J queries per round, M documents per query, local pool capacity D

function OPTIMIZEQUERIES($c_t, K_t, \mathcal{D}_e, \mathcal{E}(x_t)$)
  $\mathcal{D}_p \leftarrow$ retained solutions in $\mathcal{D}_e$; $A_t \leftarrow \mathcal{M}_\theta(\text{stats}(\mathcal{D}_p))$
  $K_t^0 \leftarrow K_t$; $\mathcal{S}_t^0 \leftarrow \emptyset$; initialize an empty score cache $\hat{s}_t$
  $\tilde{c}_t^0 \leftarrow (c_t, K_t^0, A_t)$
  for $r = 1, \ldots, R$ **do**
    $q_r^{(j)} \sim \mathcal{M}_\theta(\tilde{c}_t^{r-1}, \mathcal{S}_t^{r-1}, \{\hat{s}_t(d)\}_{d \in \mathcal{S}_t^{r-1}}), j = 1, \ldots, J$, in parallel
    $Q_r \leftarrow (q_r^{(j)})_{j=1}^J$ ▷ Query construction
    $\mathcal{W}_r \leftarrow$ SEARCH(Q_r) ▷ Up to M documents per distinct query
    $\mathcal{P}_r \leftarrow$ DEDUPLICATE($\mathcal{S}_t^{r-1} \cup \mathcal{W}_r$)
    $\mathcal{U}_r \leftarrow \{d \in \mathcal{P}_r : \hat{s}_t(d)$ is undefined$\}$
    $K_t^r \leftarrow K_t^{r-1}$
    if $\mathcal{U}_r \neq \emptyset$ **then** ▷ Evidence scoring and knowledge update
      $\{\hat{s}_t(d)\}_{d \in \mathcal{U}_r}, K_t^r \leftarrow \text{PREDICT}_{\mathcal{M}_\theta}(\tilde{c}_t^{r-1}, \mathcal{E}(x_t), \mathcal{P}_r)$
      Cache validated predictions; preserve previously cached scores
    end if
    $\mathcal{S}_t^r \leftarrow \text{TopD}_{\hat{s}_t}(\mathcal{P}_r)$ ▷ Rank documents with validated scores
    $\tilde{c}_t^r \leftarrow (c_t, K_t^r, A_t)$
  end for
  return $\mathcal{S}_t^R$ ▷ Pass retained documents to the solution prompt
end function

retained predictions, document identifiers, and measured outcomes, with up to three document bodies. Bodies are truncated to 10,000 characters, preferring full text over a snippet when available. This context lets the model relate previously retrieved evidence to actual optimization outcomes.

No-op and look-up. Under NO-OP, the solution prompt receives $\mathcal{S}_t = \emptyset$. Under LOOK-UP, the gate selects stored document identifiers, and their bodies are added to the solution prompt. Both decisions bypass the inner loop and make no web-search call.

B.3 INNER LOOP: QUERY OPTIMIZATION

Algorithm 2 is invoked only under RETRIEVE and returns the documents used for solution generation.

Population state descriptor. After a RETRIEVE decision, deterministic statistics summarize the retained population $\mathcal{D}_p$ in $\mathcal{D}_e$: its score distribution, recent parent-to-child outcomes, and parent-selection frequencies. Following EvoX (Liu et al., 2026a), $\mathcal{M}_\theta$ converts these statistics into factual observations A_t without additional interpretation. The call is skipped if the population is empty. The parent, its score, the evolutionary history, and A_t remain fixed across all R inner rounds.

Query construction. Each round generates J independent responses to the same query prompt in parallel. A response contains one query, its keywords, target source type, intended discovery direction, and rationale; the query text is limited to 500 characters. Round 1 uses $\tilde{c}_t^0 = (c_t, K_t, A_t)$. Later rounds additionally receive the retained documents and their predicted scores, and use the updated knowledge state K_t^{r-1} to target the remaining gaps.

Web search and document pooling. Queries that differ only in whitespace are executed once per round. Each distinct valid query is sent to Tavily with advanced search depth and returns up to M documents. The results are merged with $\mathcal{S}_t^{r-1}$ and deduplicated using the URL or provider identifier, title, and raw body. The pool $\mathcal{P}_r$ therefore contains at most JM documents in the first round and $D + JM$ thereafter. A discarded document re-enters the pool only if a later search returns it again.

Hypothetical evidence scoring and knowledge update. When unscored documents are present, one model call predicts their absolute candidate scores $\hat{s}_t(d)$ on the evaluator's scale and updates the knowledge state. Each prediction estimates the score obtained by improving the fixed parent using document d alone. Validated predictions are cached for the current inner loop and reused if the document appears again. Up to D documents with the best predicted scores are retained, with

ties resolved in pool order. If no document needs scoring, the call is skipped and $K_t^r = K_t^{r-1}$. After round R, the final local pool of up to D documents is passed to the outer loop. Their bodies, titles, and URLs enter the solution prompt without a separate summarization call.

Validation and failures. Only finite predictions for valid, previously unscored document identifiers enter the ranking; existing predictions cannot be overwritten. A complete valid response updates the knowledge state. Partial responses retain valid predictions while preserving the previous knowledge state, and a failed scoring call preserves both the previous retained set and state. Failed searches consume their attempts, and malformed query samples are not replaced. All R rounds run without early stopping. If no usable evidence is retained, the outer loop receives $\mathcal{S}_t = \emptyset$.

B.4 Inference Settings and Call Budget

Settings. The default retrieval budget is $R = 3$ rounds, $J = 1$ query per round, $M = 5$ documents per query, and $D = 3$ documents retained in the local pool. The entire final local pool enters the solution prompt. We evaluate $N = 1$ and $N = 8$ for candidate generation. All model calls use temperature 0.7, top-p 0.95, medium reasoning effort, and up to 32,768 output tokens. Gate, query, and evidence-scoring calls use user-only prompts with tools disabled. Population analysis uses EvoX's system prompt and supplies the deterministic statistics in the user message.

Per-iteration budget. The following counts exclude retries. Under RETRIEVE, the model-call budget includes one gate, at most one population analysis, RJ query constructions, and up to R evidence-scoring calls. With the default settings, the limits are eight retrieval-related model calls and three web searches. Candidate generation and evaluation follow separately.

Decision	Retrieval LLM calls	Web searches	Generations
NO-OP	1	0	1
LOOK-UP	1	0	N
RETRIEVE	$\leq RJ + R + 2$	$\leq RJ$	N

B.5 Coupling Web Search with Solution Evolution

Figure 9 compares three ways to integrate web search into OpenEvolve: joint-level retrieval (Table 2), sequential retrieval as in DeepEvolve (Liu et al., 2025), and bi-level retrieval in EvoDuet. They differ in when they search, what guides their queries, and how they reuse search results.

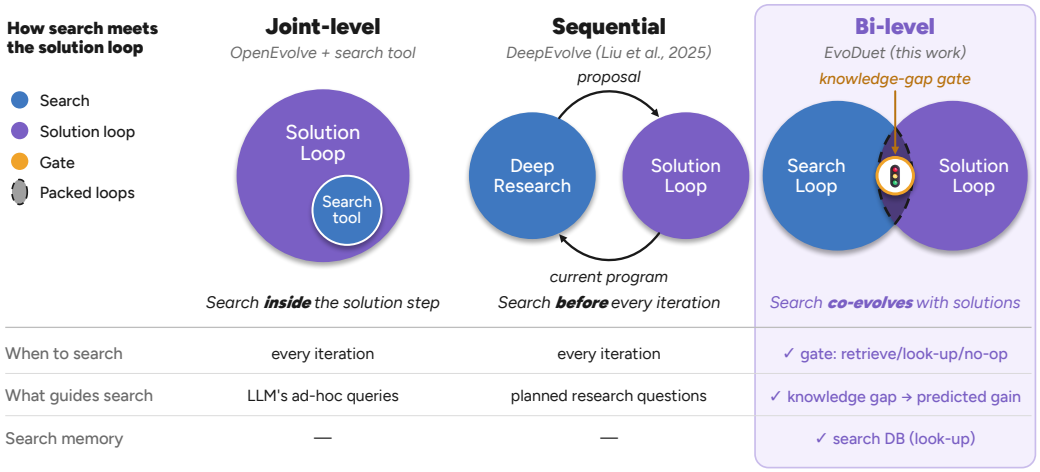


Figure 9: **Three ways to couple web search with solution evolution.** *Top:* search runs within generation (joint-level), precedes generation (sequential), or co-evolves with solutions (bi-level). Blue denotes search, violet solution evolution, and amber the knowledge-gap gate; the dashed overlap marks the coupling between the two loops. *Bottom:* search timing, query guidance, and memory for reuse across iterations. Checkmarks highlight EvoDuet's design choices.

Search placement changes how evidence enters the solution loop. *Joint-level* retrieval runs during program generation: the LLM writes queries alongside program edits and must call Tavily at least once per iteration, with up to three tool rounds. *Sequential* retrieval, as in DeepEvolve, runs a deep-research agent before generation at every iteration. Given the current program and programs selected as inspiration, the agent plans research questions, searches the web, and writes a proposal. Reflection can revisit planning or search when it judges the report incomplete; a coding agent then implements the proposal, followed by debugging and evaluation. Both joint-level and sequential retrieval are single-level: neither scores queries or documents against the task objective.

EvoDuet couples query and solution evolution through a knowledge-gap gate (§ 3). The gate reads the current knowledge state and selects RETRIEVE, LOOK-UP, or NO-OP. Under RETRIEVE, the inner loop refines queries over R rounds and selects documents using predicted program scores. The outer loop generates and evaluates programs, then stores the documents and measured outcomes in the search database to guide later gate decisions and queries.

Table 2: Joint-level versus bi-level retrieval (NDG, %).

Method	Denoising	Sums/Diffs
No search	0.0%	14.8%
Joint-level	43.2%	42.9%
Bi-level	**84.5%**	**76.0%**

Joint-level search repeatedly returns the same pages on Denoising. In the GPT-5.6-Luna trajectory analyzed in Section 2.3, joint-level retrieval returns 85 distinct URLs versus 248 for EvoDuet; 88.1% of its returned URLs are repeats. Across eligible four runs, mean NDG at iteration 100 is 43.2% on Denoising and 42.9% on Sums/Diffs, compared with 84.5% and 76.0% for EvoDuet (Table 2).

The DeepEvolve comparison uses published results. On Molecule and Burgers, DeepEvolve's released scores are 0.8149 and 0.6666, respectively, below our OpenEvolve baseline (0.8496 and 0.6937); EvoDuet reaches 0.8524 and 0.7846 (Table 1c). The local scores are the best across available four runs with GPT-5.6-Luna at iteration 100; DeepEvolve was not rerun with the same model or search budget.

C Experimental Setup Details

We evaluate on 31 tasks in two collections (Table 3). The descriptions below specify each task's optimization variables, constraints, and evaluation procedure. Arrows indicate whether lower (↓) or higher (↑) values are better. Unless stated otherwise, we report the evaluator's native objective.

Table 3: Task coverage.

Domain	Tasks	#
Simple Scientific Optimization	Pressure Vessel, SINDy, Batch Reactor, LABS-27, LJ-13, Chwirut2, Perovskites, SRSD, GuacaMol, Summit	10
Quantum compilation	Swap Reduction	1
Astrodynamics	Cassini, Galileo, Mariner 10, Rosetta, Voyager 2	5
Scientific algorithms	Denoising	1
AI foundations	Domain Mixture, U Shape, LR & BSZ, Parallel	4
Algorithm engineering	AHC039, AHC058	2
Mathematics	Erdős, AC1–AC3, CP (n=26, 32), Hadamard, Sums/Diffs	8
Total		31

C.1 Task Descriptions

C.1.1 Simple Scientific Optimization

This collection consists of ten tasks constructed from public sources. For each task, an oracle document was retrieved when the task was created (Appendix C.2). We describe the optimization problem and report the scores of both the initial program and the solution stated in the oracle document.

Pressure Vessel Design (Yang et al., 2013) (↓ cost)

The program specifies a vessel by its shell thickness d_s, head thickness d_h, inner radius r, and length L. The two thicknesses must be multiples of 0.0625 in, and the length is restricted to $L \leq 200$ in. The objective is to minimize the fabrication cost

$$C(d_s, d_h, r, L) = 0.6224 d_s r L + 1.7781 d_h r^2 + 3.1661 d_s^2 L + 19.84 d_s^2 r,$$

subject to the evaluator's thickness, volume, and dimension constraints. The initial program has a cost of 10,905.34. The oracle solution, taken from a SlideShare derivation of the optimum, has a cost of 6,059.71.

SINDy Cubic 2D (Brunton et al., 2016) (↓ NRMSE)

The input consists of 2,500 samples from a two-dimensional trajectory. The program returns the coefficients of a polynomial vector field $\dot{\mathbf{x}} = f(\mathbf{x})$ of degree at most three. The evaluator measures how accurately the recovered field describes the dynamics and how accurately its integrated trajectory reproduces the reference trajectory. The objective is to minimize the mean of the vector-field and trajectory normalized root mean squared errors (NRMSEs).
The initial program's field and trajectory NRMSEs are 0.670 and 1.197, respectively. The cubic-oscillator equation in the PySINDy oracle document gives corresponding errors of approximately zero and 1.7×10^{-8}.

Batch Reactor Control (APMonitor, 2025) (↑ $B(1)$)

The program controls the temperature of a batch reactor with consecutive reactions $A \to B \to C$. It returns a schedule of 501 temperature values over $t \in [0, 1]$, each constrained to $[298, 398]$ K. The evaluator integrates the reaction kinetics under this schedule, and the objective is to maximize the terminal concentration $B(1)$ of the intermediate product. The initial constant-temperature policy, $T(t) \equiv 350$ K, achieves 0.57706. The APMonitor GEKKO solution supplied by the oracle achieves 0.61076.

LABS-27 (Packebusch & Mertens, 2016) (↓ energy)

The low-autocorrelation binary sequence problem asks for a sequence $s = (s_1, \ldots, s_{27}) \in \{-1, +1\}^{27}$ with small correlations at nonzero shifts. Its aperiodic autocorrelation at lag k and its energy are

$$C_k(s) = \sum_{i=1}^{27-k} s_i s_{i+k}, \qquad E(s) = \sum_{k=1}^{26} C_k(s)^2.$$

The program returns the sequence, and the objective is to minimize $E(s)$. The initial sequence has energy 177. The run-length-encoded optimum reported by Packebusch and Mertens in the oracle document has energy 37.

Lennard–Jones-13 (Cambridge Cluster Database, n.d.) (↓ energy)

The program returns the positions $\mathbf{x}_1, \ldots, \mathbf{x}_{13}$ of 13 atoms in three-dimensional space. Writing $r_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$, the objective is to minimize the reduced Lennard–Jones energy

$$E(\mathbf{x}_1, \ldots, \mathbf{x}_{13}) = 4 \sum_{1 \le i < j \le 13} \left(r_{ij}^{-12} - r_{ij}^{-6} \right).$$

Coincident atoms are invalid. The initial straight-chain configuration has energy -12.374, while the configuration supplied by the Cambridge Cluster Database oracle has energy -44.327.

NIST Chwirut2 (National Institute of Standards and Technology, n.d.) (↓ RSS)

Given NIST's 54 observations (x_i, y_i), the program estimates the three parameters of the nonlinear regression model $\widehat{y}(x; b) = e^{-b_1 x}/(b_2 + b_3 x)$. The objective is to minimize the residual sum of squares

$$\mathrm{RSS}(b) = \sum_{i=1}^{54} \left(y_i - \frac{e^{-b_1 x_i}}{b_2 + b_3 x_i} \right)^2.$$

The initial program uses NIST Start 1 and has an RSS of 14,794.79. The NIST-certified parameters in the oracle document give an RSS of 513.048.

OLYMPUS Perovskites (Häse et al., 2021a;b) (↓ bandgap)

The search space contains 192 compositions, each specified by an organic component, a cation, and a halide. The program selects one composition, and the evaluator returns its published HSE06 bandgap from the dataset. The objective is to minimize this bandgap. The initial composition has a bandgap of 5.3704 eV. The hydrazinium–Sn–I composition identified in the Gryffin tutorial oracle has a bandgap of 1.5249 eV.

SRSD Feynman I.27.6 (Matsubara et al., 2024) (↓ error)

The program returns a symbolic expression that predicts the focal distance from the three inputs (d_1, n, d_2). We use the official split of 8,000 training, 1,000 validation, and 1,000 test examples. The optimization objective is the mean squared relative error on the validation set. The initial expression has an error of 0.905. Equation 27.6 of the Feynman Lectures, supplied by the oracle, gives zero error.

GuacaMol $C_{11}H_{24}$ (Brown et al., 2019) (↑ score)

The program proposes molecular structures as SMILES strings, with $C_{11}H_{24}$ as the target molecular formula. The evaluator canonicalizes and deduplicates the proposals, then averages their formula-agreement scores over the top 159 structures. Missing structures contribute zero, so the objective rewards a collection of distinct matching structures. The initial program supplies 5 of the 159 structures and scores 0.0315. The PubChem isomer list supplied by the oracle gives a score of 1.0.

Summit Baumgartner (Felton et al., 2021; Baumgartner et al., 2019) (↑ yield)

The program specifies reaction conditions for an aniline C–N coupling: catalyst, base, base loading, temperature, and residence time. Five released emulators predict the yield under these conditions. Each prediction is clipped to $[0, 1]$, and the objective is to maximize their mean. The initial conditions score 0.655. The conditions in a row of the released dataset, identified by the oracle, score 1.0.

C.1.2 SIMPLETES

We use 23 tasks from SIMPLETES (WILL Team, 2026), together with their released initial programs and evaluators. For each task, the released best program or construction is re-evaluated with the same evaluator and serves as the SOTA anchor for NDG. The tasks cover the six domains below.

Quantum compilation. These two tasks optimize different parts of the compilation process. Swap Reduction searches over initial qubit mappings and routing decisions.

Swap Reduction (↓ SWAPs)

Given a logical circuit and a device connectivity graph, a Rust routing policy chooses the initial assignment of logical qubits to physical qubits and inserts SWAP operations as the circuit is processed. The resulting mapping must allow each two-qubit gate to execute on a hardware edge. The optimization target is to reduce the number of inserted SWAPs.
The evaluation scaffold contains 72 circuit–device cases across three topologies. Its native score measures the CNOT overhead saved relative to reference routings. In the paper, we report the number of added SWAPs on the 24 circuits assigned to the 20-qubit device.

Astrodynamics. Each program returns a spacecraft trajectory as a sequence of launch, flyby, deep-space maneuver, and arrival events, with an epoch and a state for each event. The evaluator checks the trajectory using DE430 ephemerides and two-body propagation, enforces the task's time windows and minimum flyby altitudes, and charges any speed mismatch at a flyby. The common objective is to minimize the total velocity-change budget, in km/s,

$$\Delta v_{\text{total}} = \Delta v_{\text{launch}} + \Delta v_{\text{flybys}} + \Delta v_{\text{maneuvers}} + \Delta v_{\text{arrival}}.$$

The five tasks differ in their destinations, permitted flyby bodies, and arrival conditions.

Cassini (↓ Δv)

The trajectory begins at Earth and ends with capture into a 115-day orbit around Saturn. Flybys may use Venus, Earth, Mars, or Jupiter. A feasible trajectory may contain at most six flybys and six deep-space maneuvers.

Galileo (↓ Δv)

The trajectory begins at Earth and ends with capture into a 210-day orbit around Jupiter. Venus and Earth are the permitted flyby bodies, and the trajectory is limited to five flybys and five deep-space maneuvers.

Mariner 10 (↓ Δv)

The trajectory travels from Earth to Mercury, with Venus as the only permitted flyby body. The arrival contribution to Δv is zero when the arrival speed relative to Mercury is below 10 km/s.

Rosetta ($\downarrow \Delta v$)

The trajectory travels from Earth to a rendezvous with comet 67P at a fixed arrival epoch. Flybys may use Earth or Mars, and each flyby must maintain an altitude of at least 300 km.

Voyager 2 ($\downarrow \Delta v$)

The trajectory travels from Earth to a Neptune flyby, with intermediate flybys permitted at Jupiter, Saturn, and Uranus. The arrival contribution to Δv is zero when the arrival speed relative to Neptune is below 30 km/s.

Scientific algorithms. These tasks evaluate algorithms for statistical estimation. Single-Cell RNA-seq Denoising maximizes reconstruction quality.

Single-Cell RNA-seq Denoising ($\uparrow$ score)

The input is a matrix of noisy unique molecular identifier (UMI) counts. The program returns a denoised matrix of the same shape. For each dataset, the evaluator averages a normalized mean-squared-error score and a normalized Poisson score, with the additional requirement that the Poisson score be at least 0.97. The objective is to maximize this combined score. Program search uses the Pancreas dataset, while the reported result is the mean score on the held-out PBMC and Tabula datasets.

AI foundations. Each task asks the program to define a parametric predictive law and a routine for fitting it. The law is fitted separately for each evaluation group, subject to a task-specific parameter budget. Search is guided by training R^2, whereas the paper reports held-out R^2, clipped to $[-1, 1]$. The tasks differ in their input variables, prediction targets, and held-out regimes.

Domain Mixture ($\uparrow R^2$)

The law predicts losses on five domains from the proportions of those domains in a training mixture. Evaluation covers four model sizes, with at most 35 fitted parameters per size. The test set contains 24 unseen mixtures, measuring how well the fitted law predicts losses for mixtures absent from training.

U Shape ($\uparrow R^2$)

The law predicts Brier score as a function of log compute on nine benchmarks. Each benchmark permits at most six fitted parameters. The held-out set contains 127 observations at higher compute, so the reported score measures extrapolation beyond the training regime.

LR & BSZ ($\uparrow R^2$)

The law predicts language-model loss from learning rate, batch size, data size, and model parameter count, using at most 26 fitted parameters. The test set contains 117 runs at a model size and data budget absent from training, assessing transfer of the fitted relationship to that held-out setting.

Parallel ($\uparrow R^2$)

The law predicts loss from model parameter count and the number P of parallel streams. It is fitted on two datasets, with at most four parameters per dataset. Training uses observations with $P \leq 4$, and evaluation uses $P = 8$, testing extrapolation to a larger number of streams.

Algorithm engineering. We use two AtCoder Heuristic Contest problems from ALE-Bench (Imajuku et al., 2025). For each task, a C++ solver is evaluated on 150 public inputs with a 2 s limit per case. A run receives the sum of the official case scores, or zero if any case fails. The reported task score is the average over four runs.

AHC039: Purse Seine Fishing (↑ score)

Given the positions of mackerels and sardines, the solver returns an axis-parallel polygon describing a fishing region. The polygon may have at most 1,000 vertices and a perimeter of at most 4×10^5. The objective is to maximize the official score by enclosing as many mackerels and as few sardines as possible. We use ALE-Agent's fifth-place solution as the initial program.

AHC058: Apple Incremental Game (↑ score)

The solver chooses a sequence of machine-upgrade decisions over 500 turns. On each turn, it may perform one upgrade or take no action. If S is the final apple count, the objective is to maximize the official score

$$\mathrm{round}\left(10^5 \log_2 S\right).$$

The initial program performs no upgrades and receives a score of zero.

Mathematics. These tasks search for finite representations of mathematical constructions: step functions, circle packings, sign matrices, and integer sets. Each evaluator checks the construction and computes the objective defined below. Circle Packing at $n = 26$ and $n = 32$ constitutes two separate tasks.

Erdős Minimum Overlap (↓ C_5)

The program specifies a step function $h : [0, 2] \to [0, 1]$ satisfying

$$\int_0^2 h(x)\,dx = 1.$$

For this function, the evaluator computes C_5, the maximum overlap between h and its shifted complement. The objective is to minimize C_5 while maintaining the range and integral constraints.

First Autocorrelation Inequality, AC1 (↓ C_1)

The program represents a nonnegative step function on $[-1/4, 1/4]$ by n heights $h_1, \ldots, h_n$. Writing $(h * h)_k$ for the discrete convolution of the height sequence with itself, the objective is

$$\min_h C_1(h), \qquad C_1(h) = \frac{2n \max_k (h * h)_k}{\left(\sum_{i=1}^n h_i\right)^2}.$$

The heights must have a positive sum so that the normalized objective is defined.

Second Autocorrelation Inequality, AC2 (↑ R_2)

The program specifies a nonzero, nonnegative step function f supported on $[-1/4, 1/4]$. Let $f * f$ denote its self-convolution. The objective is to maximize

$$R_2(f) = \frac{\|f * f\|_2^2}{\|f * f\|_1 \, \|f * f\|_\infty}.$$

Third Autocorrelation Inequality, AC3 (↓ C_3)

The program represents a step function f on $[-1/4, 1/4]$ using signed heights $h_1, \ldots, h_n$. The objective is to minimize

$$C_3(f) = \frac{\|f * f\|_\infty}{\left(\int f\right)^2}, \qquad C_3(h) = \frac{2n \max_k |(h * h)_k|}{\left(\sum_{i=1}^n h_i\right)^2}.$$

The second expression is the evaluator's form for the step-height representation. Heights may be positive or negative, but their sum must be nonzero for the objective to be defined.

Circle Packing, n=26 and 32 ($\uparrow \sum_i r_i$)

For each of the two values of n, the program returns centers $\mathbf{c}_i = (x_i, y_i)$ and nonnegative radii r_i for n circles. Each circle must lie inside the unit square, and no two circles may overlap:

$$r_i \le x_i \le 1 - r_i, \qquad r_i \le y_i \le 1 - r_i,$$
$$\|\mathbf{c}_i - \mathbf{c}_j\|_2 \ge r_i + r_j \quad (i \ne j).$$

The objective is to maximize $\sum_{i=1}^{n} r_i$. The evaluator checks the geometric constraints with tolerance 10^{-12}.

Hadamard, Order 29 ($\uparrow$ ratio)

The program returns a matrix $H \in \{-1, +1\}^{29\times 29}$. The objective is to maximize $|\det H|$. The evaluator computes the determinant exactly and reports its absolute value divided by a fixed normalization constant.

Sums and Differences ($\uparrow c$)

The program returns a set A containing between 2 and 512 distinct integers in $[-10^6, 10^6]$. Define its sumset and difference set as $A+A = \{a+b : a, b \in A\}$ and $A-A = \{a-b : a, b \in A\}$. The objective is to maximize

$$c(A) = \frac{\log(|A+A|/|A|)}{\log(|A-A|/|A|)}.$$

The evaluator computes both sets from the submitted integers and scores their cardinalities through this ratio.

C.2 Oracle Document Construction

SimpleTES. For each task, we give Claude Code (Claude Opus 5 with maximum reasoning effort, restricted to its WebSearch and WebFetch tools) the task instructions, the initial program, the evaluator, and the best program released by SimpleTES. It identifies the functional differences that let the best program score higher, writes a web-search query for each, and returns the documents it fetched, each with a summary limited to what the document states. The resulting oracle documents are semantic rather than verbatim. Although the prompt asks for each document's text as fetched, Claude Code's WebFetch returns the output of a smaller model that reads the page rather than the page itself, so a document carries what its source says about an improvement, such as a method, a formula, or a parameter setting, but not necessarily its wording or its full content.

Simple Scientific Optimization. The oracle documents of these ten tasks were collected with Claude Code when the tasks were created. Each states the task's solution directly, for example NIST's certified parameters for Chwirut2 or the Cambridge Cluster Database coordinates for Lennard–Jones-13, and we validate the stated solution with the task's evaluator; the boxes in Appendix C.1 give its score.

D Additional Experimental Results

D.1 OpenEvolve with and without Oracle Documents

Table 4 reports the per-task mean and best NDG for the runs underlying Figure 3. We evaluate simple scientific optimization tasks at iteration 20 and the remaining tasks at iteration 100. Each condition reports the mean and maximum NDG over its available seed/edit runs independently.

Table 4: Full results for OpenEvolve with and without oracle documents. Entries report **Mean** ± **Std / Best** NDG (%; higher is better) over each condition's available seed/edit runs, where Std is the sample standard deviation and Best is the maximum run NDG. Δ reports oracle minus OpenEvolve for Mean / Best, computed before rounding. Bold marks the higher value for each statistic at the displayed precision. Row colors distinguish task groups. The overall row averages task means and task bests across all 31 tasks.

	Qwen3.5-9B			GPT-5.6-Luna		
Task	**OpenEvolve**	**+ Oracle**	**Δ (%)**	**OpenEvolve**	**+ Oracle**	**Δ (%)**
Simple scientific optimization (10)						
Pressure Vessel	49.9 ±57.6 / 100.0	**99.9** ±0.1 / 100.0	+50.1 / 0.0	100.0 ±0.0 / 100.0	100.0 ±0.0 / 100.0	0.0 / 0.0
SINDy (2D)	99.8 ±0.2 / 100.0	99.8 ±0.1 / 100.0	−0.1 / 0.0	100.0 ±0.0 / 100.0	100.0 ±0.0 / 100.0	0.0 / 0.0
Batch Reactor	38.1 ±46.3 / **100.0**	**87.2** ±20.0 / 99.4	+49.1 / −0.6	99.5 ±0.6 / 100.1	**100.0** ±0.0 / 100.1	+0.5 / 0.0
LABS (27)	61.8 ±0.0 / 61.8	**74.9** ±30.6 / **100.0**	+13.0 / +38.2	100.0 ±0.0 / 100.0	100.0 ±0.0 / 100.0	0.0 / 0.0
LJ (13)	96.0 ±7.9 / 100.0	**100.0** ±0.1 / 100.0	+4.0 / 0.0	100.0 ±0.0 / 100.0	100.0 ±0.0 / 100.0	0.0 / 0.0
Chwirut2	100.0 ±0.0 / 100.0	100.0 ±0.0 / 100.0	0.0 / 0.0	100.0 ±0.0 / 100.0	100.0 ±0.0 / 100.0	0.0 / 0.0
Perovskites	68.3 ±22.4 / 96.7	**81.3** ±19.6 / 96.7	+12.9 / 0.0	97.4 ±0.8 / 98.1	**97.9** ±0.6 / 98.1	+0.5 / 0.0
Feynman I.27.6	19.8 ±30.1 / 63.5	**100.0** ±0.0 / **100.0**	+80.2 / +36.5	100.0 ±0.0 / 100.0	100.0 ±0.0 / 100.0	0.0 / 0.0
GuacaMol	56.7 ±50.4 / 100.0	**100.0** ±0.0 / 100.0	+43.3 / 0.0	100.0 ±0.0 / 100.0	100.0 ±0.0 / 100.0	0.0 / 0.0
Baumgartner	102.0 ±4.8 / 106.8	**103.1** ±2.7 / **107.1**	+1.1 / +0.3	**94.7** ±16.8 / **107.1**	94.3 ±11.6 / 104.2	−0.5 / −2.9
Quantum compilation (1)						
Swap Reduction	5.8 ±4.9 / 10.7	**11.9** ±16.8 / **23.8**	+6.1 / +13.1	21.9 ±29.3 / 63.2	**63.4** ±20.1 / **77.6**	+41.5 / +14.4
Astrodynamics (5)						
Cassini	31.4 ±12.9 / 43.4	**39.8** ±29.4 / **60.5**	+8.4 / +17.1	73.8 ±9.3 / 81.7	**77.4** ±14.3 / **96.2**	+3.5 / +14.5
Galileo	**12.2** ±8.7 / **24.2**	8.0 ±11.3 / 16.0	−4.2 / −8.2	37.6 ±1.0 / 38.3	**48.0** ±16.7 / **59.8**	+10.4 / +21.6
Mariner 10	**93.8** ±8.9 / **100.0**	93.3 ±9.3 / 99.9	−0.5 / −0.1	100.0 ±0.0 / 100.0	100.0 ±0.0 / 100.0	0.0 / 0.0
Rosetta	**52.5** ±22.3 / **96.7**	43.5 ±11.5 / 51.6	−9.0 / −45.0	**81.4** ±19.1 / **97.8**	58.1 ±4.1 / 61.0	−23.3 / −36.9
Voyager 2	47.2 ±26.6 / **84.7**	**60.4** ±18.0 / 73.1	+13.2 / −11.6	99.3 ±1.3 / 100.0	**100.0** ±0.0 / 100.0	+0.7 / 0.0
Scientific algorithms (1)						
Denoising	**17.9** ±32.1 / **79.5**	17.5 ±24.7 / 35.0	−0.4 / −44.5	22.1 ±44.2 / **88.4**	**29.9** ±22.2 / 45.6	+7.8 / −42.8
AI foundations (4)						
Domain Mixture	65.1 ±35.5 / **90.2**	**85.1** ±2.6 / 86.9	+20.0 / −3.3	**97.5** ±0.5 / **98.1**	93.6 ±5.8 / 97.7	−4.0 / −0.4
U Shape	**34.3** ±53.2 / **109.7**	33.3 ±47.2 / 66.7	−0.9 / −43.0	0.0 ±0.0 / 0.0	**50.0** ±70.7 / **100.0**	+50.0 / +100.0
LR & BSZ	**52.9** ±33.1 / **87.0**	31.5 ±44.5 / 62.9	−21.4 / −24.0	23.9 ±30.1 / 62.4	**38.3** ±54.1 / **76.5**	+14.3 / +14.1
Parallel	26.2 ±10.1 / 40.2	**78.0** ±10.4 / **85.4**	+51.8 / +45.1	**91.3** ±15.5 / **99.5**	85.4 ±0.0 / 85.4	−6.0 / −14.2
Mathematics (8)						
Erdős	**93.1** ±5.8 / **99.8**	91.6 ±9.9 / 98.5	−1.6 / −1.3	95.8 ±7.1 / 99.9	**99.9** ±0.1 / **100.0**	+4.1 / 0.0
AC1	**27.3** ±15.8 / **50.8**	20.3 ±17.9 / 32.9	−7.0 / −17.9	42.5 ±7.3 / 52.8	**51.1** ±15.4 / **62.3**	+8.6 / +9.5
AC2	**35.1** ±17.6 / **54.8**	33.6 ±19.8 / 47.6	−1.5 / −7.2	41.2 ±24.8 / **72.0**	**57.9** ±8.2 / 66.3	+16.7 / −5.7
AC3	79.8 ±11.1 / 91.8	**81.5** ±18.6 / **94.6**	+1.7 / +2.9	**99.7** ±0.1 / **99.8**	99.6 ±0.1 / 99.7	−0.1 / −0.1
CP ($n = 26$)	56.9 ±24.9 / 81.9	**79.9** ±13.3 / **89.3**	+23.0 / +7.4	100.0 ±0.0 / 100.0	100.0 ±0.0 / 100.0	0.0 / 0.0
CP ($n = 32$)	75.1 ±4.1 / 78.7	**85.0** ±2.2 / **86.6**	+9.9 / +7.8	**100.0** ±0.0 / 100.0	99.9 ±0.2 / 100.0	−0.1 / 0.0
Hadamard	44.3 ±3.3 / **47.9**	**46.1** ±1.9 / 47.5	+1.8 / −0.4	73.1 ±17.5 / 90.8	**75.8** ±25.9 / **98.2**	+2.7 / +7.4
Sums/Diffs	**5.9** ±13.0 / **32.4**	0.0 ±0.0 / 0.0	−5.9 / −32.4	**15.4** ±1.2 / 16.0	14.8 ±4.4 / **20.5**	−0.6 / +4.5
Algorithm engineering (2)						
AHC039	2.5 ±6.1 / 15.0	**11.5** ±16.2 / **22.9**	+8.9 / +7.9	**1.3** ±1.8 / **3.8**	0.0 ±0.0 / 0.0	−1.3 / −3.8
AHC058	**57.6** ±31.7 / **86.0**	30.4 ±18.0 / 43.2	−27.1 / −42.8	**92.1** ±0.4 / 92.6	91.2 ±8.8 / **97.5**	−0.9 / +4.9
Overall average (31 tasks)	51.9 / **75.3**	**62.2** / 71.9	+10.3 / −3.4	74.2 / 82.7	**78.3** / **85.4**	+4.0 / +2.7

D.2 OpenEvolve with and without Parallel Generation

Parallel generation yields larger gains with oracle documents for GPT-5.6-Luna, but without them for Qwen3.5-9B (Table 5). Increasing N from 1 to 8 raises average NDG by 15.0% with oracle documents and 9.8% without on the nine GPT-5.6-Luna tasks; the corresponding gains on the 12 Qwen3.5-9B tasks are 2.2% and 11.8%.

Table 5: Parallel generation with and without oracle documents. Entries show mean ± sample standard deviation / best NDG (%) at iteration 100. Δ is $N = 8$ minus $N = 1$ within each method; bold marks the higher value.

	OpenEvolve			OpenEvolve + Oracle		
Task	$N = 1$	$N = 8$	Δ (%)	$N = 1$	$N = 8$	Δ (%)
Qwen3.5-9B (12 tasks)						
Quantum compilation						
Swap Reduction	5.8 ±4.9 / 10.7	5.8 ±6.8 / **13.0**	+0.1 / +2.3	11.9 ±16.8 / 23.8	**41.2** ±27.1 / **78.2**	+29.3 / +54.4
Astrodynamics						
Cassini	31.4 ±12.9 / 43.4	**44.0** ±29.3 / **60.4**	+12.6 / +17.0	**39.8** ±29.4 / **60.5**	17.2 ±29.0 / 60.3	−22.5 / −0.2
Galileo	12.2 ±8.7 / 24.2	**32.8** ±2.2 / **35.9**	+20.6 / +11.7	8.0 ±11.3 / 16.0	**24.5** ±13.0 / **35.5**	+16.5 / +19.5
Mariner 10	**93.8** ±8.9 / 100.0	73.1 ±48.9 / 100.0	−20.7 / 0.0	**93.3** ±9.3 / 99.9	48.1 ±55.6 / **100.0**	−45.2 / +0.1
Rosetta	52.5 ±22.3 / 96.7	**72.8** ±17.2 / **98.6**	+20.4 / +2.0	43.5 ±11.5 / 51.6	**58.7** ±27.0 / **77.7**	+15.2 / +26.1
Voyager 2	47.2 ±26.6 / 84.7	**95.5** ±2.4 / **97.4**	+48.3 / +12.7	**60.4** ±18.0 / 73.1	46.0 ±53.2 / **92.1**	−14.3 / +19.0
Scientific algorithms						
Denoising	26.8 ±37.5 / 79.5	**78.5** ±8.6 / **85.7**	+51.7 / +6.3	17.5 ±24.7 / 35.0	**58.7** ±12.2 / **67.3**	+41.2 / +32.4
AI foundations						
U Shape	**34.3** ±53.2 / **109.7**	21.3 ±25.1 / 48.8	−13.0 / −61.0	**33.3** ±47.2 / 66.7	23.3 ±46.6 / **93.2**	−10.0 / +26.5
LR & BSZ	52.5 ±33.8 / 87.0	**62.9** ±42.7 / **92.8**	+10.4 / +5.8	31.5 ±44.5 / 62.9	**46.1** ±37.7 / **86.5**	+14.6 / +23.6
Mathematics						
AC2	**35.1** ±17.6 / **54.8**	14.5 ±20.2 / 44.4	−20.6 / −10.4	**33.6** ±19.8 / 47.6	19.1 ±31.5 / **66.0**	−14.5 / +18.4
Sums/Diffs	**5.9** ±13.0 / **32.4**	4.4 ±3.3 / 7.6	−1.4 / −24.8	0.0 ±0.0 / 0.0	**12.2** ±7.9 / **16.5**	+12.2 / +16.5
Algorithm engineering						
AHC058	57.6 ±31.7 / 86.0	**91.1** ±1.9 / **92.2**	+33.5 / +6.2	30.4 ±18.0 / 43.2	**34.4** ±6.9 / 43.2	+4.0 / 0.0
Overall average (12 tasks)	37.9 / **67.4**	**49.7** / 64.7	+11.8 / −2.7	33.6 / 48.4	**35.8** / **68.0**	+2.2 / +19.7
GPT-5.6-Luna (9 tasks)						
Astrodynamics						
Galileo	37.6 ±1.0 / 38.3	**43.6** ±11.1 / **60.3**	+6.1 / +22.0	**48.0** ±16.7 / **59.8**	40.1 ±2.6 / 42.0	−7.9 / −17.9
Rosetta	81.4 ±19.1 / 97.8	**87.9** ±13.7 / **99.8**	+6.5 / +2.0	58.1 ±4.1 / 61.0	**83.1** ±23.2 / **99.5**	+25.0 / +38.5
Scientific algorithms						
Denoising	22.1 ±44.2 / **88.4**	**60.5** ±31.5 / 80.3	+38.4 / −8.1	29.9 ±22.2 / 45.6	**80.6** ±8.7 / **86.8**	+50.7 / +41.2
AI foundations						
U Shape	0.0 ±0.0 / 0.0	**42.6** ±50.6 / **100.0**	+42.6 / +100.0	50.0 ±70.7 / **100.0**	**72.1** ±23.3 / 88.6	+22.1 / −11.4
Mathematics						
Erdős	95.8 ±7.1 / 99.9	**99.9** ±0.1 / **100.0**	+4.1 / 0.0	99.9 ±0.1 / **100.0**	99.9 ±0.0 / 99.9	0.0 / −0.1
AC1	**42.5** ±7.3 / **52.8**	21.4 ±7.5 / 28.5	−21.1 / −24.4	51.1 ±15.4 / 62.3	**73.8** ±13.1 / **83.1**	+22.7 / +20.8
AC2	**41.2** ±24.8 / **72.0**	40.5 ±27.2 / 58.2	−0.8 / −13.8	57.9 ±8.2 / **66.3**	**58.9** ±0.0 / 58.9	+1.0 / −7.4
Sums/Diffs	15.4 ±1.2 / 16.0	**25.8** ±5.2 / **31.3**	+10.4 / +15.3	14.8 ±4.4 / 20.5	**40.7** ±5.1 / **44.3**	+25.9 / +23.8
Algorithm engineering						
AHC058	92.1 ±0.4 / 92.6	**93.9** ±1.1 / **94.5**	+1.8 / +1.9	**91.2** ±8.8 / **97.5**	86.7 ±6.0 / 90.9	−4.6 / −6.6
Overall average (9 tasks)	47.6 / 62.0	**57.3** / **72.5**	+9.8 / +10.6	55.7 / 68.1	**70.6** / **77.1**	+15.0 / +9.0

D.3 Inner-loop Budget Ablations

Table 6: Inner-loop budget ablations with GPT-5.6-Luna.

(a) Rounds R on Denoising.

Setting	NDG (%)
$R = 1$	58.35%
$R = 3$ (default)	**81.77%**
$R = 5$	34.28%

(b) Queries J on Denoising.

Setting	NDG (%)
$J = 1$ (default)	**81.77%**
$J = 3$ (k=3)	80.15%
$J = 3$ (k=5)	71.86%
$J = 5$ (k=3)	57.30%

More inner-loop search is not better, in either direction. The inner loop is controlled by two budgets: the number of search rounds R within one iteration, and the number of queries J issued per round. Table 6a and Table 6b vary each one on Denoising with GPT-5.6-Luna. Neither budget rewards being raised. Rounds peak at the default: R=1 reaches 58.35% NDG, R=3 reaches 81.77%, and R=5 falls to 34.28%, so a single round gathers too little to act on while five spend the iteration reading instead of editing. Queries are worse than flat: J=1 gives 81.77%, J=3 gives 80.15% at k=3 and 71.86% at k=5, and J=5 gives 57.30%. Issuing several queries at once broadens the retrieved set without deepening it, and the extra documents crowd the generation prompt. The behavior analysis of Section 5.1 points the same way: what moves a run is one document the optimizer can act on, not more documents.

D.4 Native Scores

Tables 7–10 report native scores at iteration 100 for OpenEvolve with and without EvoDuet, organized by task group.

Table 7: Quantum compilation and astrodynamics: native scores at iteration 100.

Model / Method	Quantum compilation	Astrodynamics	
	Swap ↓	Cassini ↓	Galileo ↓
	Mean ± Std / Best	Mean ± Std / Best	Mean ± Std / Best
Qwen3.5-9B ($N = 1$)	**19,802** ± 353 / **19,487**	4.4977 ± 0.691 / 3.8518	**2.4036** ± 0.159 / **2.1835**
+ EvoDuet	20,279 ± 393 / 20,001	**3.3296** ± 0.097 / **3.2613**	2.9228 ± 0.872 / 2.3577
Δ	−478 / −514	+1.1681 / +0.5905	−0.5192 / −0.1742
Qwen3.5-9B ($N = 8$)	19,829 ± 482 / **19,374**	3.8227 ± 1.573 / **2.9411**	**2.0271** ± 0.040 / 1.9687
+ EvoDuet	**19,697** ± 92 / 19,632	**3.1084** ± 0.221 / 2.9525	2.1751 ± 0.325 / **1.9454**
Δ	+132 / −258	+0.7143 / −0.0114	−0.1480 / +0.0233
GPT-5.6-Luna ($N = 1$)	18,950 ± 1,414 / 16,958	2.2221 ± 0.500 / 1.8019	1.9389 ± 0.017 / **1.9261**
+ EvoDuet	**17,894** ± 1,579 / **16,246**	**1.5083** ± 0.671 / **0.8756**	**1.9359** ± 0.011 / 1.9264
Δ	+1,055 / +712	+0.7138 / +0.9263	+0.0030 / −0.0003
GPT-5.6-Luna ($N = 8$)	18,032 ± 1,472 / 16,354	2.8027 ± 2.331 / **0.8830**	1.8276 ± 0.204 / 1.5223
+ EvoDuet	**17,690** ± 1,997 / **14,835**	**1.7475** ± 0.548 / 1.0238	**1.7617** ± 0.213 / **1.4863**
Δ	+342 / +1,519	+1.0552 / −0.1408	+0.0659 / +0.0361
Gemini-3.8-Flash ($N = 1$)	**17,274** ± 602 / **16,848**	6.1800 ± 0.000 / 6.1800	2.6272 ± 0.000 / 2.6272
+ EvoDuet	18,829 ± 187 / 18,697	**3.8116** ± 3.349 / **1.4433**	**1.9318** ± 0.006 / **1.9274**
Δ	−1,555 / −1,849	+2.3684 / +4.7367	+0.6954 / +0.6998
Gemini-3.8-Flash ($N = 8$)	17,363 ± 0.000 / 17,363	**0.8200** ± 0.000 / **0.8200**	**0.7302** ± 0.010 / **0.7228**
+ EvoDuet	**17,131** ± 1,009 / **15,650**	0.8375 ± 0.024 / 0.8202	2.1030 ± 0.349 / 1.9261
Δ	+232 / +1,713	−0.0175 / −0.0002	−1.3728 / −1.2033

Model / Method	Astrodynamics		
	Mariner 10 ↓	Rosetta ↓	Voyager 2 ↓
	Mean ± Std / Best	Mean ± Std / Best	Mean ± Std / Best
Qwen3.5-9B ($N = 1$)	**0.5648** ± 0.340 / **0.3274**	**5.0725** ± 1.649 / **1.7998**	**3.7582** ± 0.165 / **3.5255**
+ EvoDuet	0.6607 ± 0.412 / 0.3691	5.1472 ± 0.836 / 4.3908	3.8968 ± 0.179 / 3.7227
Δ	−0.0959 / −0.0418	−0.0747 / −2.5910	−0.1386 / −0.1972
Qwen3.5-9B ($N = 8$)	1.3569 ± 1.874 / 0.3274	**3.5636** ± 1.274 / **1.6533**	**3.4582** ± 0.015 / **3.4464**
+ EvoDuet	**0.3345** ± 0.010 / 0.3274	3.9347 ± 1.581 / 2.8168	3.8236 ± 0.265 / 3.6363
Δ	+1.0224 / 0.0000	−0.3711 / −1.1635	−0.3655 / −0.1899
GPT-5.6-Luna ($N = 1$)	0.3274 ± 0.000 / 0.3274	**2.9331** ± 1.416 / 1.7125	**3.4343** ± 0.008 / 3.4302
+ EvoDuet	0.3274 ± 0.000 / 0.3274	3.0555 ± 1.535 / **1.5486**	3.4383 ± 0.009 / 3.4302
Δ	0.0000 / 0.0000	−0.1224 / +0.1640	−0.0040 / 0.0000
GPT-5.6-Luna ($N = 8$)	1.2857 ± 1.917 / 0.3274	**2.4516** ± 1.016 / **1.5644**	**3.4302** ± 0.000 / 3.4302
+ EvoDuet	**0.3274** ± 0.000 / 0.3274	2.5281 ± 1.153 / 1.6152	3.4343 ± 0.008 / 3.4302
Δ	+0.9584 / 0.0000	−0.0766 / −0.0508	−0.0040 / 0.0000
Gemini-3.8-Flash ($N = 1$)	0.3274 ± 0.000 / 0.3274	8.9578 ± 0.000 / 8.9578	**3.7407** ± 0.439 / **3.4302**
+ EvoDuet	0.3274 ± 0.000 / 0.3274	**5.6932** ± 4.617 / **2.4285**	3.7488 ± 0.428 / 3.4464
Δ	0.0000 / 0.0000	+3.2647 / +6.5293	−0.0081 / −0.0162
Gemini-3.8-Flash ($N = 8$)	0.3274 ± 0.000 / 0.3274	**1.5168** ± 0.000 / 1.5168	**3.4302** ± 0.000 / 3.4302
+ EvoDuet	0.3274 ± 0.000 / 0.3274	2.3869 ± 1.363 / **1.4227**	3.7407 ± 0.439 / 3.4302
Δ	0.0000 / 0.0000	−0.8700 / +0.0941	−0.3105 / 0.0000

Table 8: Scientific algorithms and AI foundations: native scores at iteration 100.

Model / Method	Scientific algorithms	AI foundations	
	Denoising ↑	Domain Mix ↑	U Shape ↑
	Mean ± Std / Best	Mean ± Std / Best	Mean ± Std / Best
Qwen3.5-9B ($N = 1$)	0.6378 ± 0.0453 / **0.7060**	0.9212 ± 0.0821 / **0.9768**	**-0.5470** ± 0.704 / **0.4512**
+ EvoDuet	**0.6543** ± 0.0190 / 0.6705	**0.9322** ± 0.0552 / 0.9713	-1.0000 ± 0.000 / -1.0000
Δ	+0.0165 / −0.0356	+0.0110 / −0.0055	−0.4530 / −1.4512
Qwen3.5-9B ($N = 8$)	**0.7052** ± 0.0070 / **0.7111**	0.9119 ± 0.0799 / 0.9566	**-0.7185** ± 0.332 / **-0.3552**
+ EvoDuet	0.6752 ± 0.0466 / 0.7081	**0.9317** ± 0.0466 / **0.9646**	-1.0000 ± 0.000 / -1.0000
Δ	−0.0301 / −0.0029	+0.0198 / +0.0080	−0.2815 / −0.6448
GPT-5.6-Luna ($N = 1$)	0.6065 ± 0.0711 / 0.7132	0.9919 ± 0.0010 / 0.9930	-1.0000 ± 0.000 / -1.0000
+ EvoDuet	**0.7019** ± 0.0201 / **0.7152**	**0.9920** ± 0.0007 / 0.9930	-1.0000 ± 0.000 / -1.0000
Δ	+0.0953 / +0.0020	+0.0001 / 0.0000	0.0000 / 0.0000
GPT-5.6-Luna ($N = 8$)	0.6906 ± 0.0256 / 0.7067	0.9929 ± 0.0016 / 0.9948	**-0.4368** ± 0.670 / **0.3225**
+ EvoDuet	**0.6995** ± 0.0366 / **0.7229**	**0.9940** ± 0.0021 / **0.9971**	-1.0000 ± 0.000 / -1.0000
Δ	+0.0089 / +0.0162	+0.0011 / +0.0023	−0.5632 / −1.3225
Gemini-3.8-Flash ($N = 1$)	**0.7112** ± 0.0005 / **0.7116**	0.9923 ± 0.0002 / 0.9924	-1.0000 ± 0.000 / -1.0000
+ EvoDuet	0.7101 ± 0.0002 / 0.7102	**0.9928** ± 0.0002 / **0.9929**	**-0.5252** ± 0.671 / **-0.0504**
Δ	−0.0011 / −0.0014	+0.0005 / +0.0005	+0.4748 / +0.9496
Gemini-3.8-Flash ($N = 8$)	**0.7100** ± 0.000 / **0.7100**	**0.9962** ± 0.000 / **0.9962**	**-0.9642** ± 0.000 / **-0.9642**
+ EvoDuet	0.6915 ± 0.0333 / 0.7093	0.9928 ± 0.0001 / 0.9929	-1.0000 ± 0.000 / -1.0000
Δ	−0.0186 / −0.0007	−0.0035 / −0.0034	−0.0358 / −0.0358

Model / Method	AI foundations	
	LR & BSZ ↑	Parallel ↑
	Mean ± Std / Best	Mean ± Std / Best
Qwen3.5-9B ($N = 1$)	**0.0233** ± 0.660 / **0.6966**	**0.999579** ± 0.000054 / **0.999654**
+ EvoDuet	-0.3085 ± 0.422 / 0.1899	0.999552 ± 0.000067 / 0.999612
Δ	−0.3318 / −0.5068	−0.000027 / −0.000042
Qwen3.5-9B ($N = 8$)	**0.2270** ± 0.833 / **0.8107**	0.999750 ± 0.000108 / 0.999883
+ EvoDuet	-0.5298 ± 0.387 / -0.2563	**0.999876** ± 0.000119 / **0.999960**
Δ	−0.7568 / −1.0670	+0.000126 / +0.000077
GPT-5.6-Luna ($N = 1$)	-0.5333 ± 0.587 / 0.2172	**0.999924** ± 0.000082 / **0.999967**
+ EvoDuet	**-0.4481** ± 0.652 / **0.2740**	0.999883 ± 0.000089 / 0.999960
Δ	+0.0852 / +0.0568	−0.000041 / −0.000007
GPT-5.6-Luna ($N = 8$)	-0.8987 ± 0.203 / -0.5950	**0.999971** ± 0.000009 / **0.999981**
+ EvoDuet	**-0.3040** ± 0.662 / **0.3224**	0.999965 ± 0.000006 / 0.999970
Δ	+0.5947 / +0.9174	−0.000006 / −0.000011
Gemini-3.8-Flash ($N = 1$)	-0.9403 ± 0.084 / -0.8806	0.999875 ± 0.000115 / 0.999956
+ EvoDuet	**0.3445** ± 0.613 / **0.7781**	**0.999926** ± 0.000048 / **0.999960**
Δ	+1.2848 / +1.6587	+0.000051 / +0.000004
Gemini-3.8-Flash ($N = 8$)	-0.0528 ± 1.340 / **0.8945**	0.999964 ± 0.000007 / 0.999969
+ EvoDuet	**-0.0457** ± 0.934 / 0.8666	**0.999972** ± 0.000004 / **0.999975**
Δ	+0.0070 / −0.0279	+0.000008 / +0.000006

Table 9: Mathematics: native scores at iteration 100.

Model / Method	Mathematics			
	Erdős ↓	AC1 ↓	AC2 ↑	AC3 ↓
	Mean ± Std / Best	Mean ± Std / Best	Mean ± Std / Best	Mean ± Std / Best
Qwen3.5-9B ($N = 1$)	0.389821 ± 0.007589 / **0.381152**	**1.5128** ± 0.0019 / **1.5099**	**0.9253** ± 0.0101 / **0.9366**	**1.7978** ± 0.1889 / **1.5940**
+ EvoDuet	**0.389166** ± 0.003805 / 0.383647	1.5305 ± 0.0293 / 1.5159	0.9184 ± 0.0099 / 0.9282	1.8663 ± 0.2128 / 1.7158
Δ	+0.000655 / −0.002495	−0.0177 / −0.0060	−0.0068 / −0.0085	−0.0685 / −0.1218
Qwen3.5-9B ($N = 8$)	0.386232 ± 0.003623 / 0.382699	**1.5521** ± 0.0725 / 1.5159	0.9134 ± 0.0117 / **0.9306**	1.6036 ± 0.0202 / 1.5737
+ EvoDuet	**0.384427** ± 0.003728 / **0.381791**	1.5768 ± 0.0906 / **1.5127**	**0.9265** ± 0.0011 / 0.9273	**1.5920** ± 0.0283 / **1.5720**
Δ	+0.001804 / +0.000908	−0.0246 / +0.0032	+0.0131 / −0.0034	+0.0116 / +0.0017
GPT-5.6-Luna ($N = 1$)	0.386347 ± 0.009176 / 0.380967	**1.5109** ± 0.0009 / 1.5097	0.9288 ± 0.0143 / 0.9465	**1.4586** ± 0.0018 / **1.4571**
+ EvoDuet	**0.380937** ± 0.000101 / **0.380859**	1.5111 ± 0.0022 / **1.5085**	**0.9430** ± 0.0040 / 0.9465	1.4623 ± 0.0020 / 1.4600
Δ	+0.005410 / +0.000108	−0.0002 / +0.0011	+0.0142 / 0.0000	−0.0036 / −0.0030
GPT-5.6-Luna ($N = 8$)	**0.381008** ± 0.000082 / 0.380917	1.5135 ± 0.0009 / 1.5127	0.9281 ± 0.0162 / 0.9386	1.4604 ± 0.0036 / 1.4581
+ EvoDuet	0.382840 ± 0.003888 / **0.380859**	**1.5082** ± 0.0012 / **1.5068**	**0.9466** ± 0.0014 / **0.9481**	**1.4569** ± 0.0008 / **1.4564**
Δ	−0.001832 / +0.000058	+0.0053 / +0.0058	+0.0185 / +0.0095	+0.0036 / +0.0017
Gemini-3.8-Flash ($N = 1$)	0.380950 ± 0.000004 / 0.380947	1.5127 ± 0.0028 / 1.5107	0.9197 ± 0.0184 / 0.9327	1.4994 ± 0.0201 / 1.4852
+ EvoDuet	**0.380905** ± 0.000005 / **0.380902**	**1.5069** ± 0.0005 / **1.5066**	**0.9272** ± 0.0275 / **0.9466**	**1.4597** ± 0.0011 / **1.4589**
Δ	+0.000045 / +0.000046	+0.0057 / +0.0041	+0.0075 / +0.0139	+0.0397 / +0.0263
Gemini-3.8-Flash ($N = 8$)	0.380903 ± 0.000009 / 0.380896	1.5085 ± 0.000 / 1.5085	**0.9455** ± 0.0075 / **0.9508**	1.4586 ± 0.000 / 1.4586
+ EvoDuet	**0.380898** ± 0.000007 / **0.380893**	**1.5065** ± 0.0015 / **1.5055**	0.9429 ± 0.0066 / 0.9476	**1.4579** ± 0.0022 / **1.4564**
Δ	+0.000004 / +0.000003	+0.0020 / +0.0030	−0.0026 / −0.0032	+0.0006 / +0.0022

Model / Method	Mathematics			
	CP(n=26) ↑	CP(n=32) ↑	Hadamard ↑	Sums/Diffs ↑
	Mean ± Std / Best	Mean ± Std / Best	Mean ± Std / Best	Mean ± Std / Best
Qwen3.5-9B ($N = 1$)	2.0995 ± 0.310 / 2.4103	2.6367 ± 0.050 / 2.6810	**0.4946** ± 0.0265 / **0.5225**	1.064772 ± 0.011103 / 1.087358
+ EvoDuet	**2.6313** ± 0.006 / **2.6353**	**2.9206** ± 0.010 / **2.9280**	0.4373 ± 0.0628 / 0.4786	**1.086398** ± 0.011955 / **1.102270**
Δ	+0.5318 / +0.2250	+0.2838 / +0.2470	−0.0573 / −0.0438	+0.021626 / +0.014912
Qwen3.5-9B ($N = 8$)	1.9488 ± 0.645 / 2.5171	2.7595 ± 0.069 / 2.8081	0.2264 ± 0.1662 / 0.4756	**1.063549** ± 0.002782 / 1.066260
+ EvoDuet	**2.5352** ± 0.077 / **2.5895**	**2.8647** ± 0.071 / **2.9149**	**0.5256** ± 0.0203 / **0.5399**	1.063292 ± 0.004949 / **1.066792**
Δ	+0.5864 / +0.0724	+0.1052 / +0.1068	+0.2992 / +0.0643	−0.000256 / +0.000531
GPT-5.6-Luna ($N = 1$)	**2.6360** ± 0.000 / 2.6360	**2.9396** ± 0.000 / 2.9396	**0.7221** ± 0.1390 / 0.8626	1.072913 ± 0.001035 / 1.073430
+ EvoDuet	2.6356 ± 0.001 / 2.6360	2.9369 ± 0.005 / 2.9396	0.6606 ± 0.1737 / **0.9211**	**1.087436** ± 0.010649 / **1.098610**
Δ	−0.0004 / 0.0000	−0.0027 / 0.0000	−0.0615 / +0.0585	+0.014524 / +0.025179
GPT-5.6-Luna ($N = 8$)	2.3246 ± 0.623 / 2.6360	2.6354 ± 0.608 / 2.9396	0.8381 ± 0.1659 / 0.9211	1.081721 ± 0.004439 / 1.086459
+ EvoDuet	**2.6360** ± 0.000 / 2.6360	**2.9396** ± 0.000 / 2.9396	**0.9247** ± 0.0073 / **0.9357**	**1.105613** ± 0.021767 / **1.124462**
Δ	+0.3114 / 0.0000	+0.3042 / 0.0000	+0.0866 / +0.0146	+0.023892 / +0.038004
Gemini-3.8-Flash ($N = 1$)	2.6360 ± 0.000 / 2.6360	2.9396 ± 0.000 / 2.9396	0.8904 ± 0.0434 / 0.9211	1.089561 ± 0.014543 / 1.099845
+ EvoDuet	2.6360 ± 0.000 / 2.6360	2.9396 ± 0.000 / 2.9396	**0.9357** ± 0.0000 / **0.9357**	**1.135834** ± 0.006336 / **1.140314**
Δ	0.0000 / 0.0000	0.0000 / 0.0000	+0.0453 / +0.0146	+0.046272 / +0.040469
Gemini-3.8-Flash ($N = 8$)	2.6360 ± 0.000 / 2.6360	2.9396 ± 0.000 / 2.9396	0.8596 ± 0.000 / 0.8596	1.124049 ± 0.000 / 1.124049
+ EvoDuet	2.6360 ± 0.000 / 2.6360	2.9396 ± 0.000 / 2.9396	**0.9357** ± 0.0000 / **0.9357**	**1.143770** ± 0.001737 / **1.144999**
Δ	0.0000 / 0.0000	0.0000 / 0.0000	+0.0760 / +0.0760	+0.019721 / +0.020950

Table 10: Algorithm engineering: native scores at iteration 100.

Model / Method	Algorithm engineering	
	AHC039 ↑	AHC058 ↑
	Mean ± Std / Best	Mean ± Std / Best
Qwen3.5-9B ($N = 1$)	**554,499** ± 2,471 / **558,387**	489,794,634 ± 269,878,148 / 731,576,599
+ EvoDuet	545,315 ± 9,468 / 553,414	**564,391,050** ± 288,462,123 / **732,606,033**
Δ	−9,184 / −4,973	+74,596,416 / +1,029,434
Qwen3.5-9B ($N = 8$)	**554,619** ± 2,448 / **556,857**	**774,976,672** ± 16,261,216 / **784,638,268**
+ EvoDuet	450,025 ± 137,654 / 547,361	750,634,788 ± 31,914,083 / 773,201,452
Δ	−104,594 / −9,496	−24,341,884 / −11,436,816
GPT-5.6-Luna ($N = 1$)	**549,694** ± 8,635 / **556,948**	784,199,030 ± 3,491,251 / 787,862,153
+ EvoDuet	547,362 ± 1,702 / 548,660	**784,263,233** ± 7,949,405 / **790,891,264**
Δ	−2,332 / −8,288	+64,202 / +3,029,111
GPT-5.6-Luna ($N = 8$)	**552,132** ± 5,993 / **558,223**	**799,284,884** ± 9,407,424 / 804,359,003
+ EvoDuet	536,812 ± 12,818 / 546,430	742,512,760 ± 144,722,471 / **816,714,532**
Δ	−15,320 / −11,793	−56,772,124 / +12,355,529
Gemini-3.8-Flash ($N = 1$)	544,205 ± 1,484 / 545,254	801,414,461 ± 46,248,047 / 834,116,769
+ EvoDuet	**550,097** ± 9,545 / **556,846**	**806,467,666** ± 39,888,838 / **834,673,334**
Δ	+5,892 / +11,592	+5,053,206 / +556,565
Gemini-3.8-Flash ($N = 8$)	**562,175** ± 0.000 / 562,175	839,834,047 ± 0.000 / 839,834,047
+ EvoDuet	561,054 ± 3,577 / **563,583**	**845,831,924** ± 4,738,384 / **849,182,468**
Δ	−1,121 / +1,408	+5,997,878 / +9,348,421

E Behavior and Search Analysis

E.1 Behavior Flags and Additional Examples

We define the six behavior flags in Figure 8, report additional retrieval patterns, and examine what the flags capture in individual runs. Appendix I provides the corresponding queries, sources, code edits, and evaluation results for seven worked cases.

E.1.1 Audit Setup and Counting Rules

Records and scope. The GPT-5.6-Luna audit covers 8,200 iterations from 82 runs on the 21 tasks in the main results. The Gemini-3.8-Flash audit covers 14 runs on seven tasks and uses iterations with recorded gate decisions. For each iteration, we inspect the gate decision, queries, retained documents, predicted child scores, and the selected child's changes relative to its parent. A document is *promising* when its predicted child score exceeds the parent's score. We call a retrieval *predicted to help* if it retains at least one such document.

Behavior definitions and flags. Each run is counted once per behavior, regardless of how often it occurs; categories can overlap. Regular expressions classify the final query as a known-result lookup, a method or implementation search, a problem-statement lookup, or other. The following rules identify candidate instances of each behavior. The case reviews below distinguish these flags from evidence that a retrieved idea was implemented and retained.

- **Method transfer:** A retrieved method or implementation detail becomes a program component. We flag a run when a method or implementation query retrieves a promising document and the child sets a new run best.
- **Cross-domain transfer:** The model adapts a method or artifact from another scientific field. The flag requires a new run best with a retained document from another field, identified by its title and host. General-purpose numerical tools, such as L-BFGS, trust-region methods, and SciPy, are excluded.
- **Live documentation lookup:** Retrieving documentation to check an interface. The flag requires an API, signature, version, release-note, or documentation query; a retained documentation page; and a child that beats its parent. It does not require a verified API correction.
- **Target-informed reconstruction:** A published value or construction guides development of a solver. We flag a run when a known-result query retrieves a promising document and the child sets a new run best, with no runtime download, copied code line, or pasted numerical solution.
- **Public artifact reuse:** A selected child downloads a published witness or solution at run time, or embeds numerical values from a published solution. Both forms are manually checked.
- **Evidence-inspired over-reach:** A retrieval predicted to help produces a child whose score falls below its parent's by more than 10% of the run's total gain. Runs with no positive total gain do not receive this flag.

Artifact checks. Direct code copying occurs in 19 GPT-5.6-Luna iterations (0.2%): 16 copy a single line containing an import, path, or function signature, and three copy part of a Sums/Diffs integer set. The numeric-copy check flags added code containing at least three numbers with five or more significant digits that appear together, exactly or after rounding, on a page retrieved by that iteration. Manual review identifies 59 GPT-5.6-Luna iterations (0.7%) in four runs: three reuse published Rosetta trajectories or flyby dates, and one reuses an AC2 step-function construction. Only one of these four runs retains a copied solution in its final best program. One Gemini-3.8-Flash iteration reuses an AC1 construction. We also inspect runtime network calls; an AHC039 identifier that matches the download detector is excluded as a false positive.

Additional patterns and denominators. *Search miss with own progress* denotes a new run best after a retrieval with no promising document (57 of 82 GPT-5.6-Luna runs). *Identifier fixation* denotes a PR or issue number, or an arXiv identifier, appearing in at least 30% of a run's retrieval queries; this is assessed only for runs with at least ten retrievals (6 of 75). Figure 8(b) separately measures whether the selected child beats its parent or the previous run best, including at iterations without search. Appendix E.3 compares program revision outcomes with and without documents across all three models.

E.1.2 What the Worked Cases Establish

Method transfer: Swap Reduction (Appendix I.1). At iteration 5, the gate describes the stored documents as "largely conceptual". The retrieved dSABRE paper supplies exponential depth weighting, which becomes a term in the SWAP scorer and remains in every later best program. The final router adds 14,835 SWAPs on Q20, compared with 15,186 for the released SimpleTES program under the same evaluator. This case verifies implementation and persistence of a retrieved mechanism, beyond the method-transfer flag assigned to 55 of 82 runs.

Cross-domain transfer: Erdős minimum overlap (Appendix I.2). The model retrieves a low-sidelobe radar codeword, `04CF5A2471657C6F`, and uses it to seed an optimizer for an additive number-theory problem. All four subsequent improvements to the run best retain the codeword, reducing C_5 from 0.381374 to 0.380915. The first improvement follows a look-up two iterations after retrieval. Inspection of the 13 flagged runs also finds weaker cases: the Anscombe transform is retrieved in three Denoising runs but appears in no candidate, while a quadratic-residue initialization appears in ten selected Erdős children but is absent from that run's best program. By comparison, a variable-projection formulation introduced on Parallel Scaling at iteration 30, which accounts for 56% of that run's gain, remains in its best program at iteration 84. A cross-domain source alone therefore does not establish lasting transfer.

Live documentation lookup: Voyager 2 (Appendix I.3). The rule flags 72 retrievals in 27 runs; 13 of these iterations set a new best. Of the 72, 21 ask for an astrodynamics wrapper already documented in the task prompt, and 18 target a pull-request diff. In Voyager 2, 28 of 46 retrievals name the wrapper, but the retained pages describe public Lambert solvers. At iteration 41, a new tour reduces Δv from 3.457 to 3.447 km/s and explicitly passes `lowpath=True, M=0`. These are the existing defaults, so they do not change the solver branch; the final best program does not descend from this child. Across 300 wrapper-targeting retrievals, only two retain a page from the benchmark's source, and neither improves on its parent. The flag thus captures documentation-seeking behavior more broadly than verified API corrections.

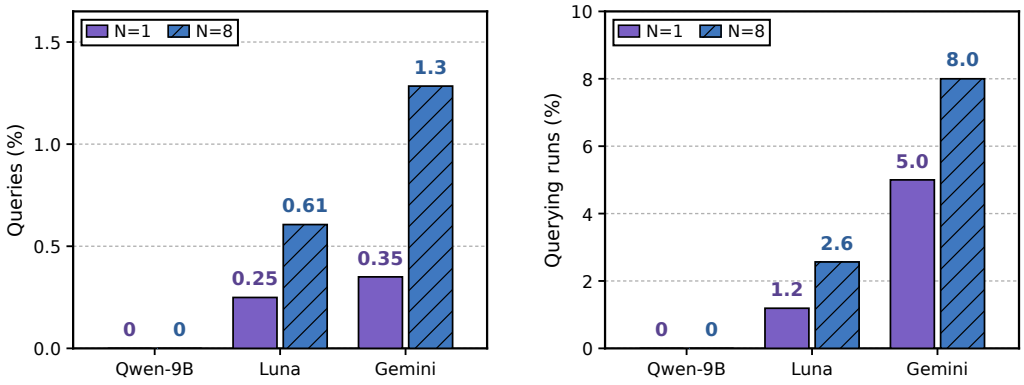
Target-informed reconstruction: circle packing, n=26 (Appendix I.4). The first query names a best-known sum of radii, 2.635977. The returned pages supply no coordinates, and the model writes an LP-plus-SLSQP solver that raises the objective from 1.390 to 2.628 in one iteration. At iteration 7, it reaches 2.635983, exceeding the queried target and matching the SimpleTES reference to six decimals. Later queries seek the record holder's source file, but the best program contains a generated solver with no retrieved coordinates or runtime download. Target-informed reconstruction is flagged in 40 of 82 GPT-5.6-Luna runs and 12 of 14 Gemini-3.8-Flash runs.

Public artifact reuse: Erdős minimum overlap, second run (Appendix I.5). At iteration 17, the child downloads a published witness and retains its optimizer as a fallback, reducing the run's best C_5 from 0.381017 to 0.380859. The best program without a runtime download reaches 0.381004. The program reported in the Erdős row of Table 1a retains this download, so its result depends on access to the published artifact (Appendix K). Two of 82 GPT-5.6-Luna runs download artifacts at run time; four others embed published numerical solutions, including three Rosetta trajectories or sets of flyby dates.

Evidence-inspired over-reach: Galileo (Appendix I.6). At iteration 33, a document's predicted child score is 0.63701 (parent: 0.63692), but the rewrite scores 0.37713, raising Δv from 1.926 to 3.873 km/s. The run best remains 0.63692; iteration 34 recovers this score from a different parent without retrieval. Across GPT-5.6-Luna runs, 20% of retrievals predicted to help produce a worse child, and 29 of 82 runs contain a drop exceeding 10% of their total gain. Predicted gains therefore require validation by the evaluator.

E.2 Language, Resources, and Reuse in Web Search

Multilingual queries are rare and confined to contest tasks. Figure 10 shows Japanese phrases in 134 of 32,008 parseable queries (0.42%), all on AHC039 or AHC058. These occur in 7 of 338 querying runs (2.1%), with none from Qwen3.5-9B. We inspect non-English-script matches and Latin-script language-detector candidates; the latter contain only English terms, names, and identifiers. This conservative screen covers query text only.

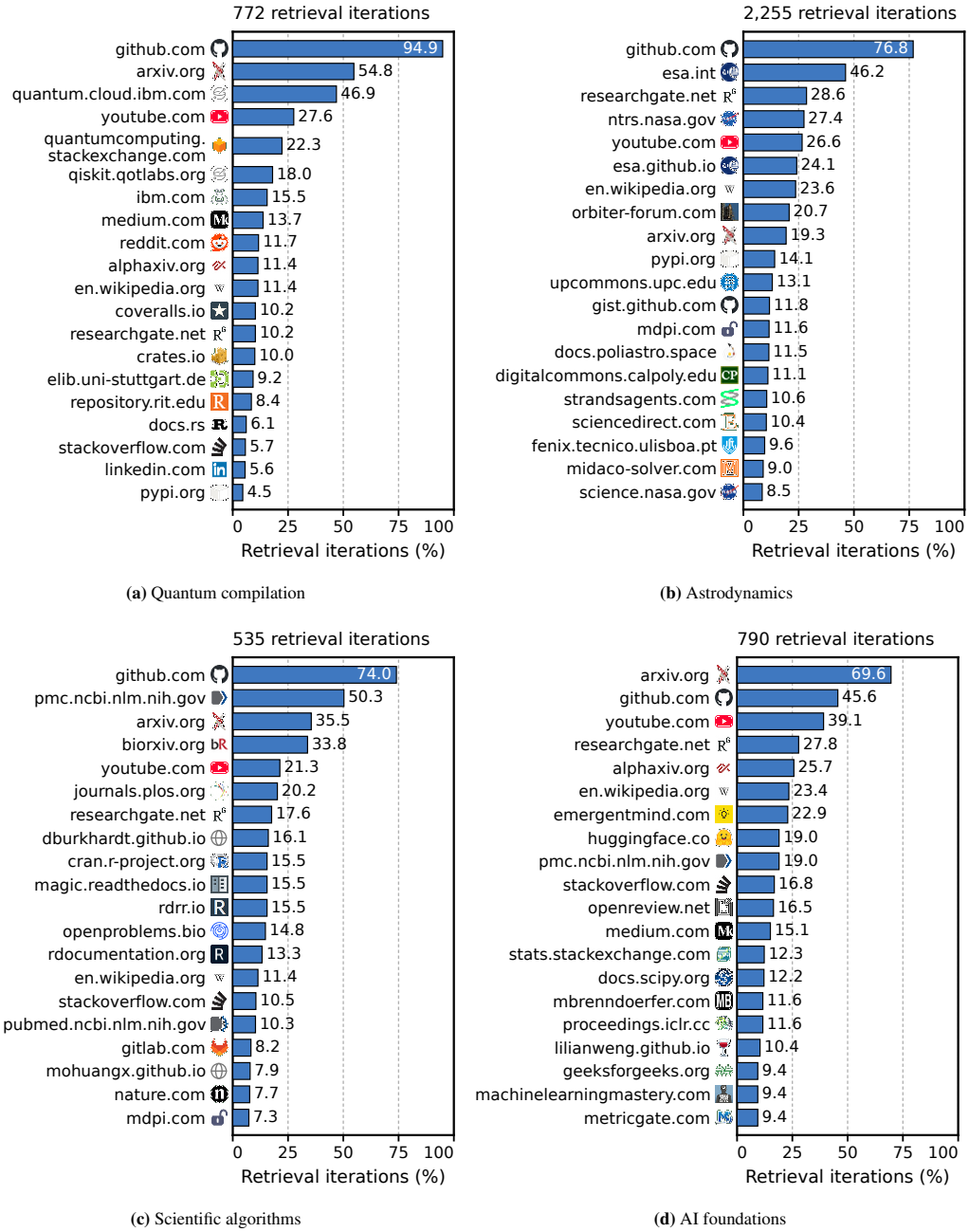


(a) Queries with Japanese phrases

(b) Runs with Japanese phrases

Figure 10: **Japanese phrases in search queries.** (a) Share of parseable queries containing Japanese phrases, including repeated outputs. (b) Share of querying runs with at least one such query. Purple and hatched blue bars denote $N = 1$ and $N = 8$. Accented names such as Erdős are excluded.

Resource preferences vary across task domains. GitHub is the most exposed host in four of the six task domains, reaching 94.9% of retrieval iterations in quantum compilation. arXiv leads AI foundations (69.6%) and algorithm engineering (48.5%; Figure 11). These rates measure source exposure only.



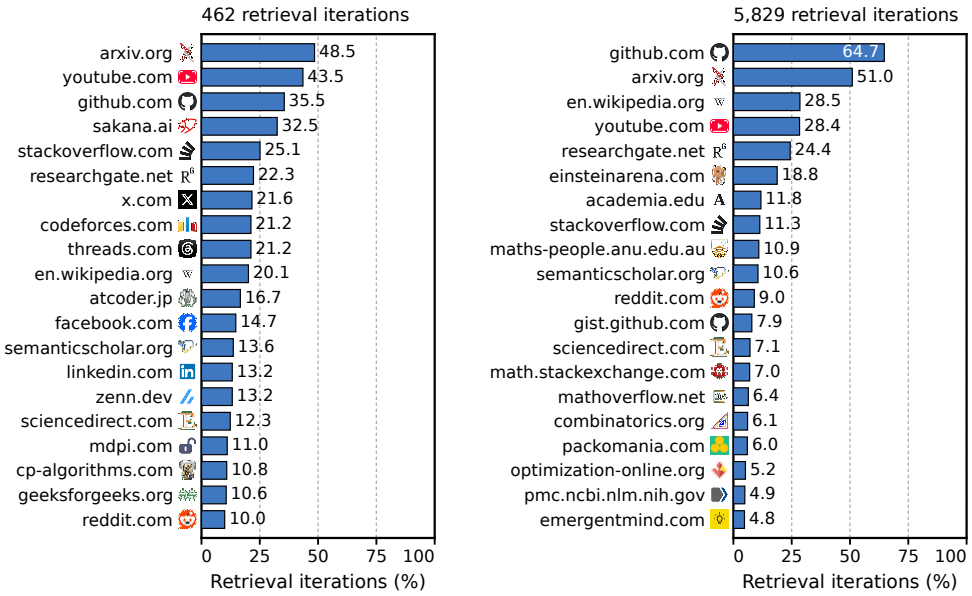
(a) Quantum compilation

(b) Astrodynamics

(c) Scientific algorithms

(d) AI foundations

Figure 11: **Top-20 web sources by task domain.** Hosts are ranked by the share of retrieval iterations in which they appear in the scorer prompt, pooling all models, budgets, seeds, and edit modes. Each host counts once per iteration; hosts can co-occur. All panels use a common 0–100% scale and report retrieval counts.



(e) Algorithm engineering

(f) Mathematics

Figure 11: **Top-20 web sources by task domain (continued).** Exposure rates pool all available runs within each domain, using the same counting rule and scale.

Models differ in how they balance retrieval and reuse. Figure 12 aggregates runs within each task group and model. In mathematics, Qwen's reuse share rises from 25.7% in iterations 1–10 to 55.0% in iterations 91–100, while Luna retrieves in 70.8% of the final ten slots.

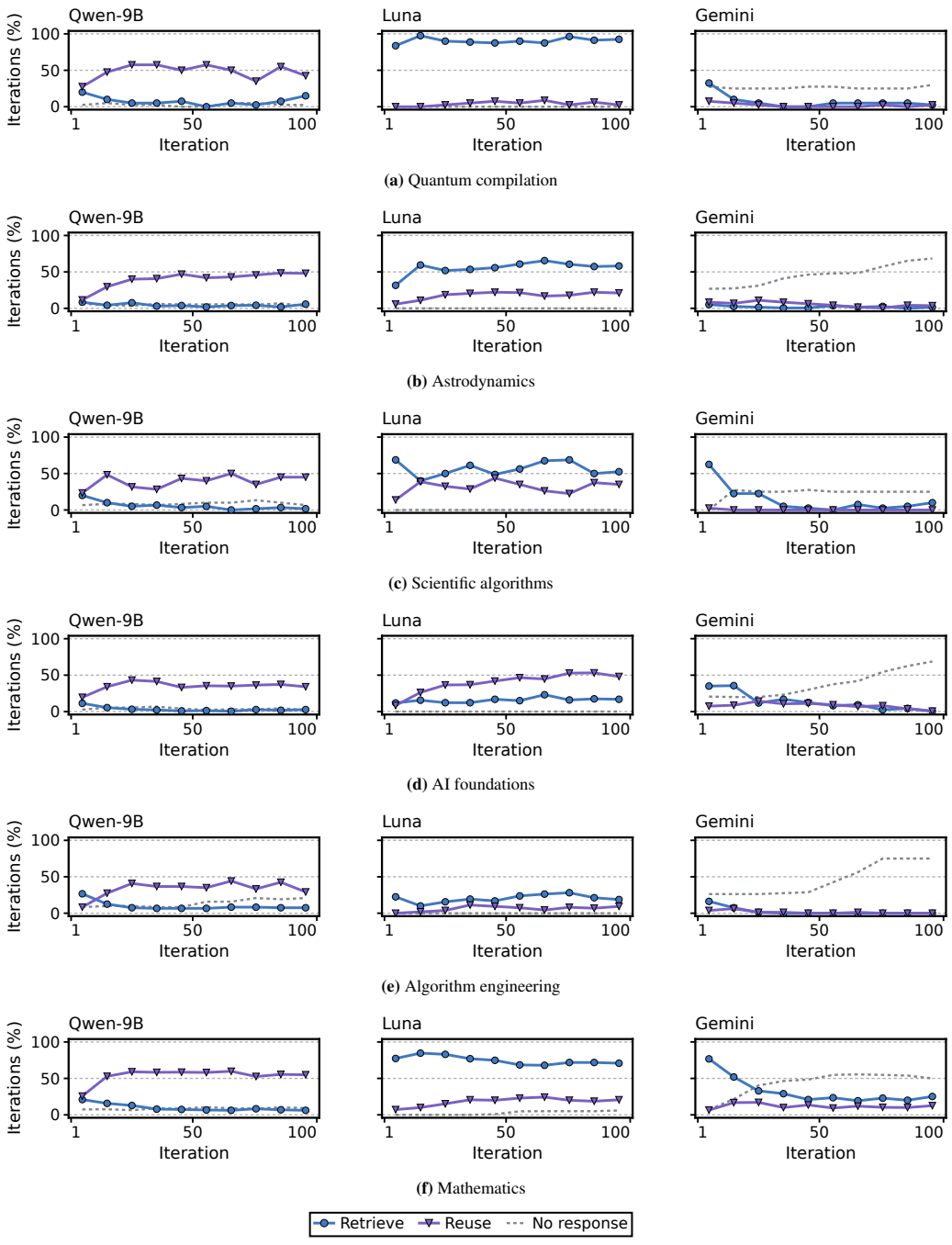


Figure 12: **Aggregated search dynamics by task group and model.** Curves show retrieval (blue), reuse (purple), and missing gate responses (gray) in ten-iteration bins. Each panel pools all available runs for its task group and model, across seeds, edit modes, and $N \in \{1, 8\}$. Percentages use all scheduled slots, including no-op and malformed decisions. All panels share the same axes.

Late lookups can reuse documents retrieved at the start. In quantum compilation, 12 of Luna's 21 resolved document requests during iterations 81–100 reuse documents first retrieved during iterations 1–10 (Figure 13).

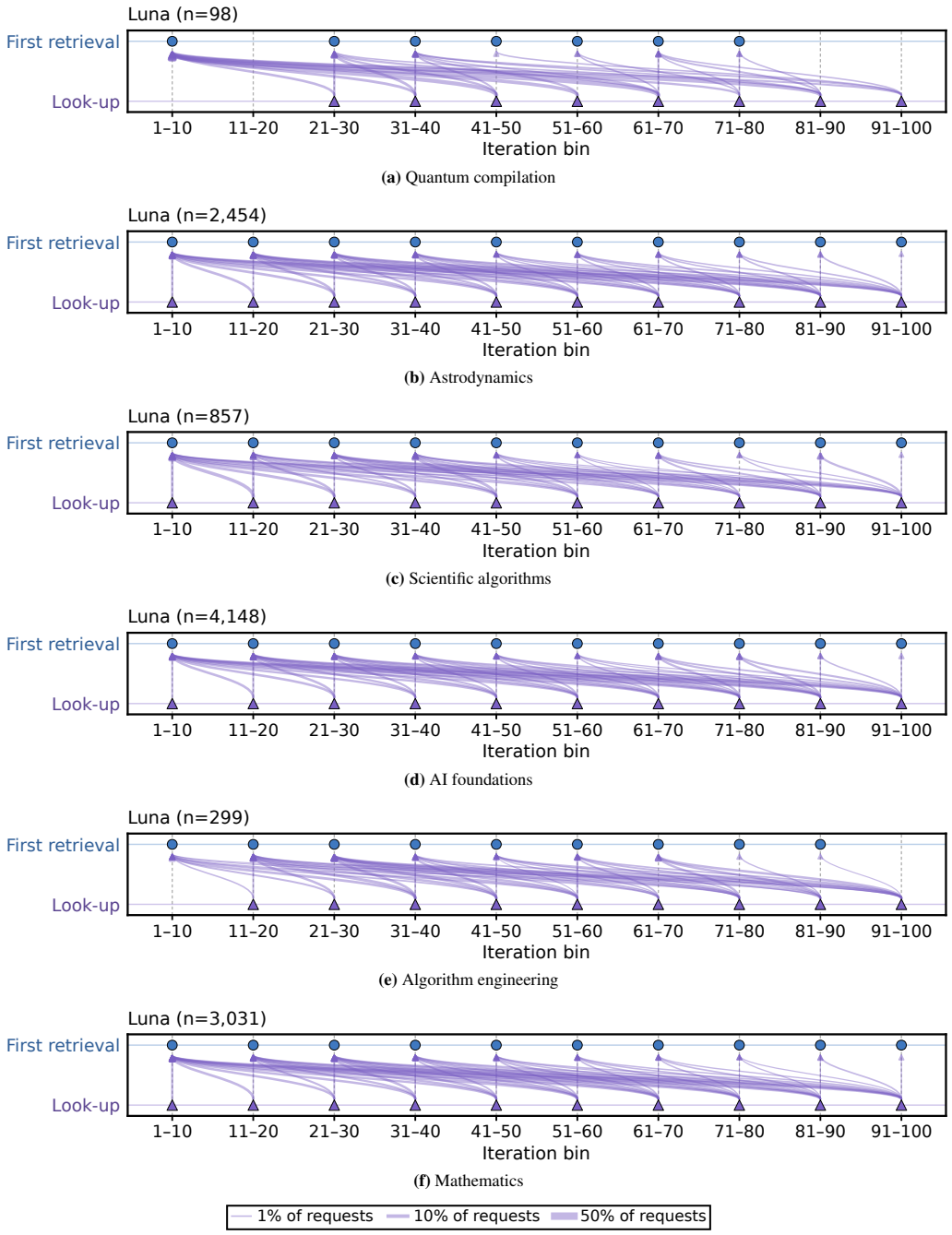


Figure 13: **Luna's lookup provenance by task group.** Arrows point from each lookup (purple, bottom) back to the document's first recorded retrieval (blue, top), pooling all available Luna runs across seeds, edit modes, and budgets. Both endpoints use ten-iteration bins; vertical arrows represent reuse within the same bin. Width encodes the within-panel share of resolved document ID–iteration pairs on a common square-root scale; n is their total. All 10,887 resolved pairs are shown; 63 unresolved pairs are excluded.

Stored documents remain in use across many iterations. Luna reuses documents with a median age of 17 iterations in scientific algorithms and 44 in quantum compilation (Figure 14). These distributions pool resolved requests within each task group and model.

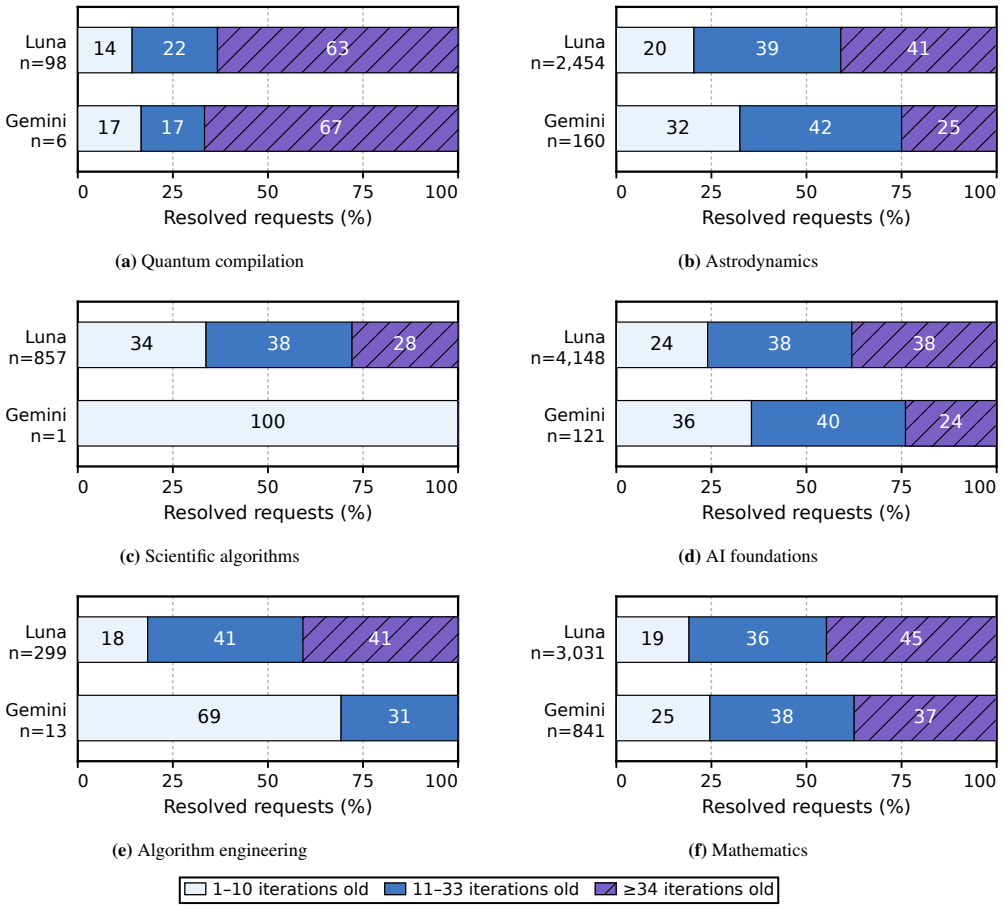
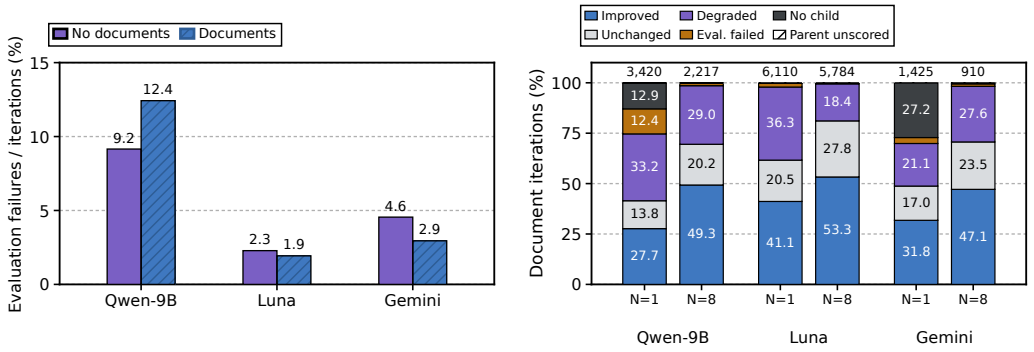


Figure 14: **Source age at reuse by task group and model.** Bars aggregate resolved document requests from available Luna and Gemini checkpoints, pooling seeds, edit modes, and both candidate budgets. Age is the reuse iteration minus the first retrieval iteration; n counts document ID–iteration pairs. We exclude 63 unresolved Luna pairs. Qwen lacks saved checkpoint provenance and is omitted from this figure.

E.3 Failure Analysis of Retrieved Evidence Use

We analyze 33,784 iterations from 362 EvoDuet runs across the 21 tasks with Qwen3.5-9B, GPT-5.6-Luna, and Gemini-3.8-Flash at $N \in \{1, 8\}$. We consider each run's first 100 iterations and exclude iterations with recorded external service errors. Document exposure is verified from the generation prompt. Outcomes compare the selected child's recorded search score with its parent's.



(a) Evaluation failures at $N = 1$

(b) Outcomes with documents

Figure 15: **Program revision outcomes with retrieved documents. (a)** Evaluation failure rates with and without documents at $N = 1$; both conditions use EvoDuet. **(b)** Outcomes of 19,866 document iterations, with counts above the bars. All rates include unsuccessful generations in the denominator. The $N = 8$ policy selects among eight candidates at document steps. Comparisons are observational because the gate selects document exposure.

Qwen's failures with documents extend beyond evaluation errors. At $N = 1$, Qwen fails evaluation in 12.4% of document iterations versus 9.2% without documents (Figure 15(a)). Only 27.7% improve on the parent, versus 41.1% for Luna and 31.8% for Gemini; 33.2% produce a lower score and 12.9% retain no child. Evaluation errors alone therefore miss revisions that execute but fail to improve.

Qwen improves more often when selecting from parallel candidates. At $N = 8$, Qwen's document iterations improve on the parent in 49.3% of cases and fail evaluation in 1.1%. Luna and Gemini improve in 53.3% and 47.1%, respectively. Since RETRIEVE and LOOK-UP generate eight candidates per attempt while NO-OP generates one, these outcomes reflect both generation and selection.

Table 11: **Examples of unsuccessful document use by Qwen at $N = 1$.** Supplied documents, parent and child code, and evaluator records are inspected together. Scores are recorded search scores (↑). These cases illustrate mechanisms rather than their population frequencies.

Task	Supplied evidence	Observed revision	Outcome
Sums/Diffs	A construction using the Chinese Remainder Theorem.	Adds random local search without implementing the supplied modular construction.	1.05979 → 1.05979
Circle Packing (26)	Joint optimization of centers and radii using SLSQP.	Adopts SLSQP but mixes grouped and interleaved parameter layouts; a returned circle extends outside the square.	Invalid
Denoising	The Anscombe transform $2\sqrt{x + 3/8}$ and its inverse.	Omits the factor of two in the forward transform, while the inverse still divides by two before squaring.	0.28849 → 0.28607

Retrieved methods are unused or incorrectly implemented in 26% of sampled Qwen revisions. Codex inspected 50 revisions sampled uniformly from 2,980 eligible Qwen revisions with documents and retained code at $N = 1$. Methods were unused in 3 (6%) and incorrectly implemented in 10 (20%); 22 incorporated or retained a method, 7 lacked actionable evidence, and 8 were uncertain. These model-assessed sample proportions exclude external errors and count only specific new discrepancies with supplied methods. Table 11 provides separate examples; Denoising's score decrease cannot be attributed to its transform error alone.

F LIMITATIONS

Generalization Across Backbone Models. Our experiments cover GPT-5.6-Luna, Gemini-3.8-Flash, and Qwen3.5-9B. The benefits of EVODUET vary across these models: Qwen3.5-9B has lower overall NDG with EVODUET than with OpenEvolve alone at both $N = 1$ and $N = 8$. Our analysis also identifies cases in which this model ignores retrieved methods or implements them incorrectly. Evaluating more model families and sizes, and examining whether training for evidence selection and implementation reduces these failures, would help establish when a backbone can benefit from bi-level co-evolution.

Limited Gains on Algorithm Engineering Tasks. EVODUET does not consistently improve performance on algorithm engineering tasks such as AHC039 and AHC058. Figure 6b shows lower group-average NDG than OpenEvolve in four of the six model/budget settings, with an average decline of 1.6% across all six. These losses occur with both GPT-5.6-Luna and Qwen3.5-9B at $N = 1$ and $N = 8$. Further work is needed to determine how evidence selection, adaptation to task-specific heuristics, and the allocation of search effort contribute to these failures, and to improve the framework on these tasks.

G DISCUSSIONS

The benefits of oracle documents vary across models and tasks. On Swap Reduction, the same documents raise NDG by 6.1% for Qwen3.5-9B and 41.5 for GPT-5.6-Luna, while both models lose performance on Rosetta (Figure 3c). Qwen's positive average oracle gain also shows that its losses with EVODUET do not imply a general inability to use documents. Those losses may reflect difficulties in identifying knowledge gaps, retrieving relevant evidence, or implementing it. The unused methods and incorrect implementations in Appendix E.3 provide examples of the last difficulty. These observations motivate training and evaluating evidence selection and implementation as separate skills, while accounting for baseline performance and search budget.

H BROADER IMPACT

EVODUET may make computational scientific discovery more efficient by helping researchers find and apply relevant knowledge from the web. However, unreliable sources can lead to incorrect conclusions, while reusing published solutions can result in overstated claims of novelty. To support auditing, we record retrieval and optimization trajectories. We also recommend explicit source attribution, independent validation of generated programs, and human oversight of downstream applications.