EVIDENCE-DRIVEN DISCOVERY LOOPS FOR AI-TRAIN-AI: FROM TRIALS TO BETTER DECISIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

An emerging frontier of automated AI research is AI-train-AI, in which agents autonomously propose, implement, and evaluate machine learning solutions within a closed discovery loop. These experiments consume substantial resources and often fail or yield marginal gains. Under a limited budget, final solution quality depends on choosing when to refine or combine implementations and which candidates to build on. Each trial produces evidence beyond a scalar score, yet this evidence is often only partially used when operations and sources are selected. We introduce the Evidence-Driven Discovery Loop (EDL), which treats evidence use as part of algorithm development. It connects accumulated trial evidence to subsequent search decisions. Its outer loop captures successful and failed trials, deterministically organizes implementations by algorithmic direction, and links modification intents to evaluated outcomes. Its inner agentic policy, Operation-aware Evidence Control (OEC), starts from compact direction cards and operation-level meta signals, actively queries supporting histories and code, and constructs plans specifying operations, source roles, changes, and execution allocations. Both levels serve final solution quality within a budget that accounts for decision and execution costs. Successful and failed attempts update the evidence available to later decisions, making trial and error cumulative without changing controller parameters. Experiments on machine learning engineering tasks show strong solution quality across three model backbones, and NatureBench Lite results extend the comparison to scientific algorithm development. Selected component interventions characterize organization, action control, and evidence acquisition.

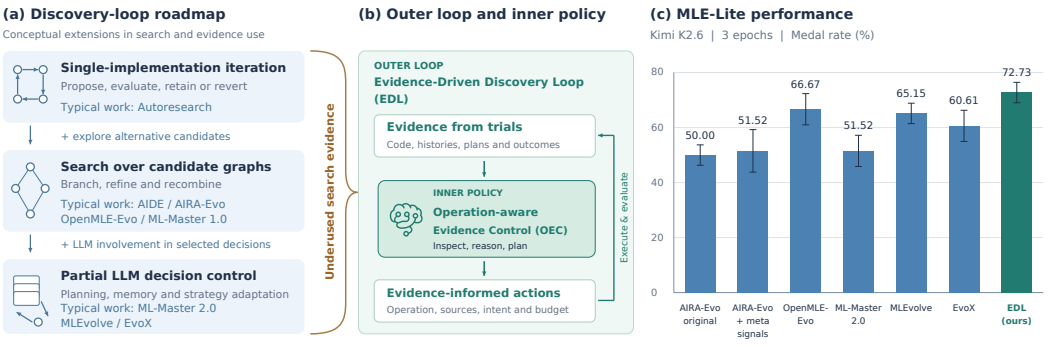

Figure 1: **(a)** A conceptual roadmap of discovery-loop patterns in AI-train-AI systems, from single-implementation iteration to candidate-graph search and LLM-guided decisions; patterns overlap. **(b)** Our Evidence-Driven Discovery Loop (EDL) organizes trial evidence in an outer loop and embeds Operation-aware Evidence Control (OEC), an inner agentic controller that inspects this evidence to guide operation, source, and modification decisions. **(c)** EDL achieves the highest MLE-Lite medal rate among the compared harnesses with Kimi K2.6. Bars show means $\pm$ run SD.

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.

1 INTRODUCTION

Recursive self-improvement (RSI) requires AI systems that can build and improve AI, including its learning algorithms (Yang et al., 2026). Machine learning (ML) engineering provides measurable feedback for studying this capability (Huang et al., 2024; Chan et al., 2025). We study *AI-train-AI*: agents propose, implement, and evaluate ML solutions (Lu et al., 2024). Each trial consumes time for implementation, training, evaluation, and repair. Under a limited budget, the central question is how to turn these costly trials into useful progress: *reaching good solutions sooner and obtaining better solutions within the same time*.

Tree and graph search organize alternative implementations (Jiang et al., 2025; Toledo et al., 2025; Yang et al., 2026); history-informed planning and strategy adaptation further guide exploration (Du et al., 2026; Zhu et al., 2026; Liu et al., 2026). The remaining challenge is to connect accumulated evidence to the specific experiment being considered. A strong implementation need not be the most productive source for further development (Katona et al., 2021), and individually reliable sources may be incompatible when combined. Compact *meta signals* summarize outcomes and costs, but their usefulness depends on the intended operation. Choosing what to try next requires interpreting these signals together with the changes, failures, and code behind them.

We introduce the *Evidence-Driven Discovery Loop* (EDL), which couples evidence construction across experiments with evidence use within each decision (Figures 1 and 2). The outer loop preserves successful and failed trials, organizes them by algorithmic direction, and links intended changes to outcomes. The inner policy, *Operation-aware Evidence Control* (OEC), actively inspects this evidence to jointly decide what to attempt and which sources to use. Retrieved evidence can revise both the proposed change and its sources before execution; the outcome then informs subsequent decisions. Both levels serve final solution quality under a shared budget for decision and execution costs, without updating controller parameters.

Our contributions are: **(1) Evidence-driven discovery.** We introduce a two-level framework that connects accumulated trial evidence to experimental decisions, making earlier successes and failures reusable throughout ML development. **(2) Operation-aware experiment design.** Motivated by our study of how meta signals affect source selection across operations (shown in Appendix A.2), we develop OEC to jointly determine what evidence to inspect and what experiment to run, assessing sources through intended changes and partner compatibility. **(3) More effective budgeted AI-train-AI.** On MLE-Lite, EDL improves Human Rank by 0.019–0.037 over the strongest baselines across three backbones. With Kimi, it matches the strongest baseline's 12-hour validation-derived medal rate by six hours. In component interventions, joint operation/source selection produces 50.7% more improvements over the current best solution with 14.1% fewer attempts than fixed numerical routing.

2 RELATED WORK

Search in ML engineering. Auto-WEKA and auto-sklearn optimize predefined model and hyperparameter spaces (Thornton et al., 2013; Feurer et al., 2015). LLM agents construct executable pipelines: SELA uses Monte Carlo tree search (Chi et al., 2024), AIDE searches implementation trees (Jiang et al., 2025), and AIRA-Evo adds recombination (Toledo et al., 2025). DS-Agent reuses external cases (Guo et al., 2024); MLEvolve and ML-Master 2.0 support history-informed planning through pipeline graphs and cognitive caches (Du et al., 2026; Zhu et al., 2026).

Reflection and evidence-guided decisions. Reflexion stores verbal feedback for subsequent trials (Shinn et al., 2023), and LATS combines reflection with tree search (Zhou et al., 2024). ReEvo and GEPA use reflective feedback to evolve heuristics and prompts, respectively (Ye et al., 2024; Agrawal et al., 2026). For ML research, MARS extracts lessons by comparing solutions (Chen et al., 2026), while Research Preference Models rank candidates from plans, code, and results (Foster et al., 2026). OEC inspects histories and code to jointly select operations and sources.

Evolutionary and hierarchical discovery. Evolution through Large Models uses LLMs as program mutation operators (Lehman et al., 2024); FunSearch combines program generation with evaluation (Romera-Paredes et al., 2024), as do AlphaEvolve and ShinkaEvolve (Novikov et al., 2025; Lange et al., 2025). MAP-Elites retains diverse high-quality solutions (Mouret & Clune, 2015), and adaptive

operator selection uses historical operator rewards (Fialho et al., 2010). AI Scientist-v2 extends agentic tree search to experimentation and paper generation (Yamada et al., 2025). Two-level designs also exist: EvoX couples solution evolution with search-strategy evolution (Liu et al., 2026); LoongFlow plans from selected parents and lineage feedback (Wan et al., 2025). EDL couples evidence construction across trials with adaptive inspection within each decision, allowing evidence to revise operations and sources together before execution.

3 Preliminaries: Discovery Loops for ML Algorithm Development

The ML discovery loop. ML algorithm development is iterative: each proposed change is implemented, trained, and evaluated, and the outcome guides the next change. In AI-train-AI, an agent runs this loop over a space $\mathcal{X}$ of executable algorithms or pipelines. For a fixed task, let H_t denote the history before step t: implementations, attempted changes, observed outcomes, and costs. A search policy π maps H_t and the remaining budget $B_t = B - \sum_{s<t} c_s$ to the next experiment, whose cost c_t includes inspection and planning as well as implementation and evaluation. The goal is the best final solution attainable within the budget:

$$\max_{\pi} \ \mathbb{E}_{\pi}[U(\hat{x}_T)], \qquad \hat{x}_T = S(H_T), \qquad \sum_{t<T} c_t \le B, \tag{1}$$

where T is the stopping step, S is a fixed validation-based selection rule (with a default outcome if no valid candidate exists), and U measures final performance on held-out evaluation unavailable during search. Because experiments are costly and often fail, final quality depends on how well π turns past outcomes into the next decision.

Candidate-graph search: operations and sources. Existing systems often instantiate π as search over a growing archive represented by a graph $G_t = (V_t, E_t)$. Each node stores an implementation $x_v \in \mathcal{X}$ together with its evaluation record, while edges record derivation or reference relations (Figure 2, left). A single-implementation loop, such as autoresearch,[1] forms the special case of a single path, whereas tree- and graph-based systems maintain multiple branches and may refine or combine earlier candidates (Jiang et al., 2025; Toledo et al., 2025; Yang et al., 2026). Each expansion applies an operation to one or more source implementations. The operation determines how a new candidate is generated, while the sources determine which existing implementations it builds on or borrows from. We represent an experiment action as

$$a_t = (o_t, P_t, h_t, b_t) \sim \pi(\cdot \mid H_t, B_t), \tag{2}$$

where o_t is the operation, P_t specifies the sources and their roles, h_t describes the intended modification, and b_t gives the execution allocation. The usefulness of a source depends on the intended operation and modification. A candidate suitable for further refinement may not be suitable for composition, and two individually strong candidates may still be difficult to combine. Existing systems often simplify these decisions by using per-candidate statistics $\phi_t(v)$ to define a numerical utility $u_t(v) = f(\phi_t(v))$ for source selection, while operation selection is handled separately (Jiang et al., 2025; Toledo et al., 2025; Yang et al., 2026). This separation does not capture how source value can depend on the operation being considered, nor does it use the reasons why earlier modifications succeeded or failed when specifying h_t.

Meta signals and search evidence. Existing systems often underuse information already produced during search. A trial records more than a candidate score. It also records how the candidate was produced, whether the attempt succeeded, and what it cost. We call compact descriptors derived from these records *meta signals*, with ϕ_t above as one example. These signals summarize the search state, while the histories and code behind them preserve the context of earlier decisions and outcomes. This information is useful when choosing (o_t, P_t, h_t) jointly, but much of it can be lost when source selection is reduced to a scalar utility. Meta signals provide a compact entry point to the accumulated evidence. Detailed records can then be inspected when more context is needed. Section 4 develops this idea into a search policy for deciding the next experiment.

[1] https://github.com/karpathy/autoresearch

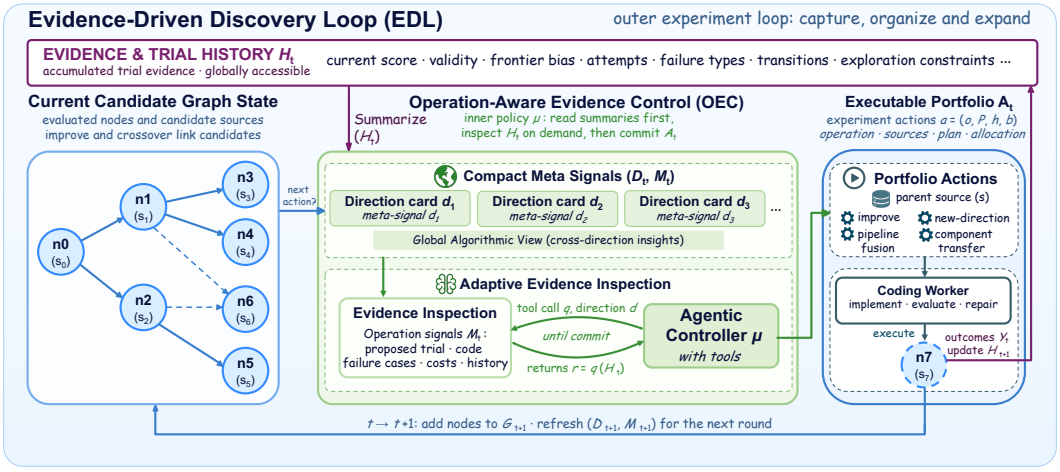

Figure 2: Evidence-Driven Discovery Loop (EDL) with Operation-aware Evidence Control (OEC). The outer loop records every trial in the history H_t (top) and summarizes it into direction cards D_t and operation signals M_t. OEC (center) reads these summaries, calls tools that retrieve histories and exemplar code from H_t, and commits a portfolio A_t of actions $a = (o, P, h, b)$ (right). Coding workers implement and evaluate each action; outcomes Y_t update H_{t+1}, expand the candidate graph G_t (left), and refresh (D_{t+1}, M_{t+1}) for the next round.

4 Methodology: Evidence-Driven Discovery Loop

The Evidence-Driven Discovery Loop (EDL) splits the search policy of Section 3 into two levels (Figure 2). The outer loop executes experiments and turns their records into reusable evidence. It does not decide which experiment to run next. The inner policy, Operation-aware Evidence Control (OEC), uses this evidence to construct and check a portfolio of experiment actions. Once committed, the portfolio returns to the outer loop for execution, and its outcomes update the evidence available to the next round. We detail the outer loop and inner policy in Sections 4.1 and 4.2, respectively.

4.1 Outer Loop: Evidence Construction

Decision linked trial history. The outer loop runs the candidate graph search introduced in Section 3. It begins with several Draft attempts. In later rounds, it executes the actions committed by OEC in Section 4.2. Debug remains an execution level repair step rather than a controller decision. Every trial is stored in the history H_t, referred to as the Journal in our implementation, whether the trial succeeds or fails. Each record is linked to the decision that produced it. It contains the operation, source roles, modification plan, execution status, score, and cost. This representation separates successful execution from improvement over the selected source and from improvement over the best solution found so far.

Compact meta signals. Reading the full history at every decision would be costly. EDL therefore summarizes H_t deterministically at two levels:

$$(D_t, M_t) = \text{Summarize}(H_t). \tag{3}$$

The direction cards D_t group implementations into algorithmic directions using fixed rules applied to their code. The rules consider pipeline type, model families, and ensembling (Appendix B.1). The grouping is intended to support navigation rather than establish semantic equivalence. Each direction card reports its best valid exemplar, attempt and failure counts, runtime information, repair recovery, recent outcomes, and the latest controller proposal together with its result. The operation signals M_t summarize execution success, improvement frequency, and cost for each operation. Grouping by direction keeps the initial controller context compact as the archive grows. These summaries are deliberately lossy. They serve as an index into H_t, not as a replacement for the underlying records. Directions also define the units from which OEC selects sources. A selected direction resolves to its current exemplar. The summaries therefore provide the initial search view from which OEC can construct a portfolio and decide where more detailed inspection is needed.

4.2 INNER POLICY: OPERATION-AWARE EVIDENCE CONTROL

OEC instantiates the search policy π of Equation 2 as an LLM driven controller. In each round, it starts from (D_t, M_t), inspects selected parts of H_t when needed, and constructs a portfolio of experiment actions for the outer loop. Unlike the factorized rules described in Section 3, OEC does not choose an operation independently of a score based source distribution. Its operation, source, and modification choices are conditioned jointly on the evidence acquired during planning.

Portfolio decision object. The output of each planning round is a portfolio A_t, an ordered list of experiment actions:

$$A_t = (a_{t,1}, \ldots, a_{t,m}).$$

Each action a specifies one proposed experiment:

$$a = (o, P, h, b), \qquad P = ((d_j, \rho_j))_{j=1}^{k}, \qquad h = (\eta, \Delta, K, C, p). \tag{4}$$

Thus, A_t is the decision for the current planning round, while each $a_{t,i}$ is an individual action within that portfolio. The action type o determines what the controller proposes to do. After initialization, OEC may Improve an existing direction, perform Crossover between existing directions, or propose a new direction. OEC may alternatively stop or defer the search instead of committing an experiment portfolio. *Improve* modifies one declared component of a single primary source. *Crossover* combines existing directions through either component transfer or pipeline fusion. *Component transfer* adapts a donor component into a primary implementation, while *pipeline fusion* combines two or three sources at the prediction interface. A *new-direction* action introduces a model family not represented in D_t. The source specification P identifies the directions used by the action and assigns each a role ρ_j. Each selected direction $d_j \in D_t$ resolves to its current exemplar x_{d_j}. The modification plan h describes what the coding worker should implement. The mechanism η and target component Δ define the intended change. K records what should be preserved, C specifies implementation checks, and p provides optimization instructions. For a transfer, the plan identifies the donor component, explains how it should be adapted, and records the interfaces that need to be checked. Finally, $b \in \{1, 2\}$ specifies the number of independent implementations requested for the same intent. A portfolio contains at most five execution slots in total, so $\sum_{i=1}^{m} b_{t,i} \leq 5$. The portfolio is the object that OEC constructs during the inner loop. Evidence queries can change its actions, sources, or modification plans before commitment, while subsequent checks determine whether the resulting portfolio is admissible for execution.

Controller objective. The controller policy μ governs evidence acquisition, portfolio construction, and commitment under the objective in Equation 1. Reasoning and evidence access incur controller cost c_s^{ctrl}, while implementation, repair, and evaluation incur execution cost c_s^{exec}; both draw from the remaining budget B_t. Each committed portfolio must satisfy $A_s \in F_s$, where F_s denotes the portfolios admissible at step s. OEC implements μ through LLM inference. It does not estimate an explicit value function or train a separate policy.

Adaptive portfolio construction. At planning step ℓ, the controller observes

$$z_{t,\ell} = (\tau, B_t, D_t, M_t, T_{t,\ell}),$$

where τ is the task and $T_{t,\ell}$ is the planning trace so far: the conversation, previous evidence queries, returned records, and feedback from earlier checks. The initial state $z_{t,0}$ contains the compact summaries from Equation 3 but not the full history. To construct a portfolio, the controller can retrieve evidence, check a proposal, or submit it for reservation:

$$u_{t,\ell} \sim \mu(\cdot \mid z_{t,\ell}), \qquad u_{t,\ell} \in Q_{\mathrm{ev}} \cup Q_{\mathrm{chk}} \cup \{\mathrm{Reserve}(A)\}. \tag{5}$$

An evidence query, portfolio check, or unsuccessful reserve call q returns

$$r = q(H_t, z_{t,\ell}),$$

and extends the trace,

$$T_{t,\ell+1} = T_{t,\ell} \| (q, r).$$

The response can guide the next query, prompt revisions to the portfolio, or change which portfolios are admissible. The experiment policy thus depends on the evidence gathered throughout planning rather than on a fixed evidence bundle. OEC retrieves evidence through two interfaces.

`inspect_direction_history` returns records for a selected direction. Queries can limit the number of recent attempts and include related records in primary, donor, source, or result roles. These records link earlier modification plans to their baselines, gains, failures, repairs, and execution costs. `get_exemplar_code` returns a selected direction's current exemplar code, along with its node identity, score, runtime, and code-reading completion status. The controller can use this code to assess the feasibility of a modification or a donor's compatibility with the primary implementation. OEC adapts its queries to the planning state. A non-improving transfer may prompt it to reconsider the mechanism or donor. An execution failure may prompt inspection of related failures and repairs. The prompt and limits appear in Appendices B.2–B.3.

Portfolio checks and commitment. Before execution, the candidate portfolio must satisfy the constraints on its actions and sources. Two interfaces check requirements that the compact summaries alone cannot establish. `check_direction_novelty` tests whether a proposed model family is absent from the current directions, overlaps with an explored family, or is unsupported. `estimate_execution_cost` assigns pipeline fusion a heuristic risk level based on the number of sources and their combined code size. The `reserve_portfolio` interface implements $\mathrm{Reserve}(A)$, committing an admissible portfolio or returning constraint violations for revision. The call succeeds if and only if $A \in F_{t,\ell}$, where $F_{t,\ell}$ is the set of portfolios admissible at the current planning step. Each action must satisfy the contract defined above for Improve, Crossover, or new-direction. The controller must have read the complete code of every selected existing source. New-direction actions require a completed novelty check; pipeline-fusion actions require a completed cost check. Experiment intents must be distinct, with repeated implementations of the same intent specified by b. If a check fails, OEC uses the returned violations to revise the operation, source roles, or intended mechanism. It can also gather further evidence before retrying reservation. A successful reservation commits the portfolio for execution and fixes it for the current round.

Portfolio execution and feedback. After commitment, control returns to the outer loop. The execution slots allocated by A_t are executed sequentially. Each resulting outcome in Y_t is linked to the action that produced it. The history, summaries, and remaining budget are then updated:

$$H_{t+1} = \mathrm{Update}(H_t, A_t, Y_t), \quad (D_{t+1}, M_{t+1}) = \mathrm{Summarize}(H_{t+1}), \quad B_{t+1} = B_t - c_t^{\mathrm{ctrl}} - c_t^{\mathrm{exec}}. \tag{6}$$

The refreshed summaries form the initial state for the next planning round. Each direction card now records the outcome of the latest relevant controller proposal, while history queries expose the more detailed decision and outcome record behind that summary. Both successful and failed experiments therefore change the evidence available to later decisions. The discovery loop adapts through this accumulated history without updating the controller parameters. Appendix B.4 summarizes the complete execution cycle.

5 Experiments

We evaluate whether EDL turns costly AI-train-AI trials into useful progress sooner. We distinguish two goals: reaching a given solution quality earlier and obtaining higher quality within a fixed budget. Across ML engineering and scientific algorithm development, we measure final performance, progress over time, and how often experiments improve the best solution found so far. Component interventions examine the costs and benefits of evidence use. A supporting study of 13 operation–signal configurations diagnoses the information available in trial history (Appendix A.2).

5.1 Experiment Setting

Benchmarks and environments. We use MLE-Lite, a 22-task subset of MLE-bench (Chan et al., 2025), and NatureBench Lite, a ten-task subset of NatureBench (Wang et al., 2026). For MLE-Lite, candidate programs run in execution sandboxes with six vCPUs and one RTX 4090, with approximately 12 GiB of GPU memory allocated per task. Each trajectory has a 12-hour cumulative sandbox budget and an 18-hour model-plus-sandbox budget. We report *submission rate*, the fraction of task runs with valid final submissions; *medal rate*, the fraction earning any medal; and *Human Rank* $(1 - \mathrm{Grade})$, normalized performance relative to the task's human leaderboard. All three are higher-is-better. For NatureBench Lite, candidate programs run in Docker containers on a shared pool of seven RTX 3090 GPUs (24 GB each) and one A800 (80 GB). Each task has four hours of effective search; evaluation uses the best valid score obtained for that task. We report *RankScore*, the

Table 1: MLE-Lite results for submission rate, medal rate, and Human Rank (all ↑), computed from 22 tasks × 3 search epochs (66 task runs) per model–harness configuration. Metrics are aggregated over 22 tasks within each epoch and averaged across three epochs; ± denotes the SD across epochs where reported. Kimi / GLM / DS denote Kimi K2.6 / GLM 5.2 / DeepSeek-V4.1-Flash. Bold: best per model.

Harness / variant	Sub. rate (%) ↑			Medal rate (%) ↑			Human Rank ↑		
	Kimi	GLM	DS	Kimi	GLM	DS	Kimi	GLM	DS
AIRA-Evo (original)	97.0	98.5	98.5	50.00 ± 3.71	57.58 ± 4.29	65.15 ± 3.71	0.690 ± 0.038	0.698 ± 0.012	0.804 ± 0.030
AIRA-Evo (meta signals)	93.9	**100.0**	98.5	51.52 ± 7.73	60.61 ± 4.29	66.67 ± 0.00	0.686 ± 0.085	0.717 ± 0.038	0.812 ± 0.011
OpenMLE-Evo (meta signals)	**98.5**	89.4	98.5	66.67 ± 5.67	**62.12** ± 5.67	68.18 ± 3.71	0.786 ± 0.029	0.707 ± 0.027	0.838 ± 0.039
ML-Master 2.0	97.0	89.4	**100.0**	51.52 ± 5.67	56.06 ± 5.67	66.67 ± 2.14	0.733 ± 0.019	0.708 ± 0.057	0.830 ± 0.014
MLEvolve	95.5	90.9	98.5	65.15 ± 3.71	56.06 ± 2.14	68.18 ± 0.00	0.778 ± 0.022	0.710 ± 0.021	0.828 ± 0.014
EvoX	97.0	83.3	**100.0**	60.61 ± 5.67	53.03 ± 2.14	68.18 ± 7.42	0.771 ± 0.031	0.652 ± 0.015	0.820 ± 0.035
EDL (ours)	**98.5**	92.4	98.5	**72.73** ± 3.71	**62.12** ± 4.29	**75.8** ± 3.71	**0.823** ± 0.020	**0.736** ± 0.021	**0.862** ± 0.019

Table 2: NatureBench Lite results over ten tasks (all means ↑). Model abbreviations and colors follow Table 1. M-SOTA: $g > 0$; S-SOTA: $g > 0.1$. Bold: best mean per model.

Harness / variant	RankScore ↑			M-SOTA (%) ↑			S-SOTA (%) ↑		
	Kimi	GLM	DS	Kimi	GLM	DS	Kimi	GLM	DS
OpenMLE-Evo (meta signals)	0.2443	0.3688	0.5495	50.0	50.0	70.0	0.0	10.0	30.0
ML-Master 2.0	0.4452	**0.6307**	0.6183	50.0	**70.0**	70.0	**30.0**	**40.0**	30.0
MLEvolve	0.3048	0.4838	0.8179	50.0	60.0	**100.0**	10.0	20.0	50.0
EvoX	0.3690	0.4376	0.6183	50.0	50.0	80.0	0.0	10.0	30.0
EDL (ours)	**0.5498**	0.4652	**0.8967**	**60.0**	**70.0**	**100.0**	20.0	30.0	**70.0**

mean normalized task rank, and fractions exceeding the reference (*M-SOTA*, $g > 0$) or improving on it by more than 10% (*S-SOTA*, $g > 0.1$). Task lists and evaluation details appear in Appendix C.1.

Harnesses and baselines. We use Kimi K2.6 (Kimi Team et al., 2026), GLM 5.2 (GLM-5-Team et al., 2026), and DeepSeek-V4.1-Flash (DeepSeek-AI et al., 2026) as backbone models. The original AIRA-Evo uses the current node score to drive source selection (Toledo et al., 2025). Its variant using meta signals and OpenMLE-Evo both use three factors for source selection: current score, delta score, and novelty (Yang et al., 2026). ML-Master 2.0, MLEvolve, and EvoX provide complete-harness alternatives based on experience reuse, pipeline-graph search, and strategy evolution (Zhu et al., 2026; Du et al., 2026; Liu et al., 2026). Appendix C.2 summarizes the distinctive designs of these frameworks; EDL implementation details appear in Appendix B.

5.2 Experiment Results

Overall performance and cross-benchmark generalization. On MLE-Lite, EDL achieves the highest Human Rank across all three backbones, improving by 0.019–0.037 over the strongest baselines (Table 1). It also achieves the highest or tied-highest medal rate, reaching 72.73% with Kimi versus 66.67% for the strongest baseline. The gains also extend from ML engineering to scientific algorithm development. As shown in Table 2, on NatureBench Lite, EDL improves RankScore over the strongest baseline by 0.1046 with Kimi (ML-Master 2.0) and 0.0788 with DeepSeek (MLEvolve); with DeepSeek, S-SOTA also rises from MLEvolve's 50% to 70%. These comparisons cover both numerical node-expansion frameworks that use meta signals and frameworks in which LLMs influence selected search decisions. The MLE-Lite results favor EDL's design of organizing meta signals and inspecting their supporting evidence to jointly choose operations, sources, and modification plans over both alternatives.

Search efficiency and target attainment. Counting both planning and execution in elapsed wall-clock time, EDL's validation-derived medal rate at two hours exceeds EvoX, the strongest competitor, by 1.52 percentage points and OpenMLE-Evo by 10.61 points (Figure 3a). These early gains indicate faster convergence toward medal-level solutions. By six hours, EDL reaches 62.12%, matching

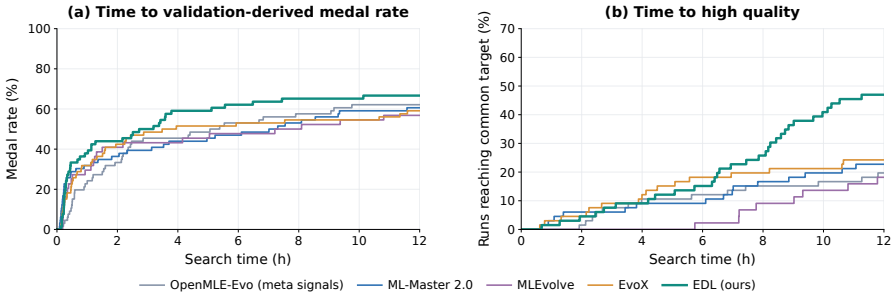

Figure 3: Search efficiency of OpenMLE-Evo (meta signals), ML-Master 2.0, MLEvolve, EvoX, and EDL with Kimi K2.6 on 22 MLE-Lite tasks. **(a)** Medal rate derived from the best validation score available in each run over time. **(b)** Fraction of runs reaching a shared high-quality target for each task, defined as the 75th percentile of the best validation scores from six reference runs.

the highest baseline rate at twelve hours (OpenMLE-Evo), and continues to 66.67% at twelve hours. To assess continued progress toward better solutions, Figure 3b reports the fraction of runs reaching a shared high-quality target: the task's 75th percentile of reference-run best validation scores (Appendix C.3). Although early attainment is mixed, EDL's lead over EvoX grows from 6.06 percentage points at eight hours to 22.73 points at twelve hours as search continues. This widening gap is consistent with more effective search decisions as trial evidence accumulates, allowing more runs to reach demanding targets within the same wall-clock budget. With Kimi K2.6, EDL uses 1.49M tokens per task-run versus OpenMLE-Evo's 0.69M (inference costs in Appendix C.3).

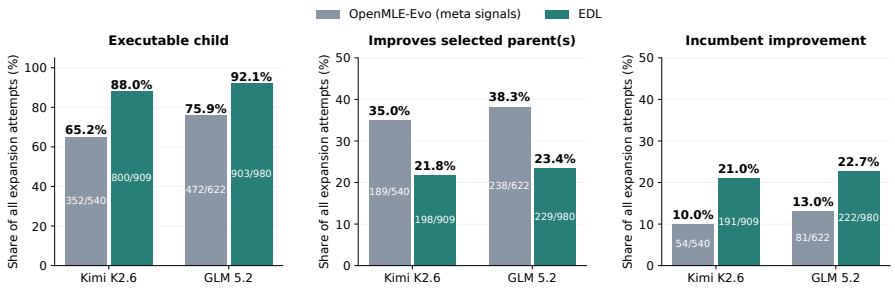

Figure 4: Improve/Crossover outcomes for OpenMLE-Evo (meta signals) and EDL with Kimi K2.6 and GLM 5.2. **Left:** fraction of attempts producing an executable child. **Middle:** fraction improving on the best selected parent. **Right:** fraction improving on the incumbent, the best solution found before the action. All rates include failed attempts in the denominator; labels inside bars show outcome counts over total attempts.

Useful progress per attempted experiment. To relate time efficiency to experimental decisions, we compare OEC's joint operation/source choices with OpenMLE-Evo's numerical selection. Figure 4 (definitions in Appendix C.4) shows higher fractions of executable children and improvements over the current search best for EDL, including failed attempts in the denominator. Although a smaller fraction outperform their selected parents, a larger fraction surpass the incumbent: most parent-beating EDL actions also advance the global search best, whereas many OpenMLE-Evo parent wins remain below it. The relevant outcome is thus whether an attempt advances the best available solution, beyond improving a weaker source. This pattern is consistent with OEC comparing experiments across directions in the context of the current search best. Its more frequent use of incumbent sources also contributes. Parent-relative delta score can favor gains within weaker lineages that leave the best available solution unchanged, despite consuming training time. Higher incumbent-improvement rates therefore provide a more relevant measure of useful experimental progress. Appendix C.5 traces how inspected histories and code inform an executed combination plan.

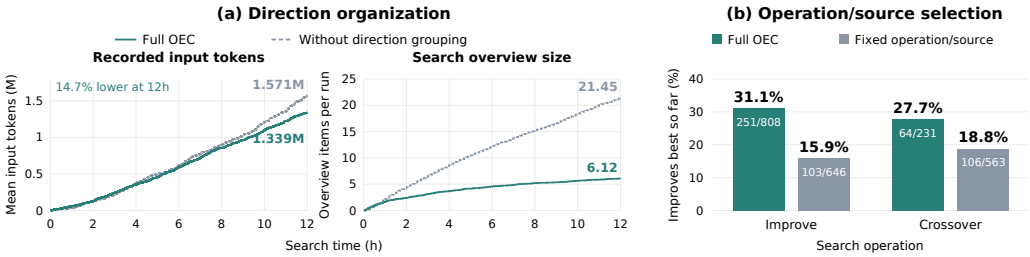

Figure 5: OEC ablation results. (a) Mean cumulative input tokens and search overview size during the first 12 wall-clock hours. Overview items are direction groups for OEC and individual candidates without grouping. (b) Fraction of actions that improve the best validation score found so far over complete searches, including failures; labels show improvements/attempts.

Ablation study. We ablate OEC's direction organization, action selection, and evidence retrieval with DeepSeek-V4.1-Flash (Table 3). **No groups** removes direction-based history organization (§4.1): candidates appear individually rather than grouped by algorithmic direction. **Fixed route** removes LLM-based joint operation/source selection (§4.2). Instead, a numerical router chooses the operation and source candidates. **Fixed evidence** removes interactive evidence retrieval (§4.2). The controller receives quality-ranked code and recent histories once per round and cannot query further details. Appendix C.6 details the interventions.

Table 3: OEC ablation results: No groups, Fixed route, and Fixed evidence.

Variant	Medal rate (%)	Human Rank
Full OEC	$\mathbf{75.8} \pm 3.71$	$\mathbf{0.862} \pm 0.019$
No groups	69.70 ± 2.14	0.8367 ± 0.0078
Fixed route	69.70 ± 2.14	0.8327 ± 0.0219
Fixed evidence	68.18 ± 6.43	0.8320 ± 0.0134

Component contributions. As shown in Figure 5a, direction grouping reduces search overview items by 71.5% and input-token use by 14.7% at 12 wall-clock hours relative to No groups. By organizing candidates into algorithmic directions, EDL makes evidence access more token-efficient and substantially reduces the number of items OEC must consider when surveying explored algorithms. This compact overview helps OEC compare search directions and plan subsequent experiments, with higher final performance than the ungrouped variant as shown in Table 3.

As shown in Figure 5b, evidence-driven joint operation/source selection produces 50.7% more improvements over the best solution so far with 14.1% fewer attempts than Fixed route. Both Improve and Crossover achieve higher rates of improving the search best than their numerically routed counterparts. These results support OEC's ability to use accumulated evidence to match operations with productive sources, yielding more effective experiments within each operation as well as better overall action allocation.

Adaptive retrieval improves medal rate by 7.62 percentage points over Fixed evidence (Table 3). Replaying fixed retrieval on the same decision states shows that OEC inspects fewer complete code exemplars per round, yet accesses code or detailed histories omitted by the fixed bundle in 20.2% of rounds. On the Leaf Classification task, a lower-ranked donor provides complementary components that reduce validation log loss (details in Appendix C.6). Adaptive inspection can thus recover useful evidence beyond quality-ranked prefetch, improving terminal quality at greater token cost without uniformly raising per-action success.

6 Conclusion

EDL connects accumulated trial evidence to experimental decisions. Its outer loop organizes successful and failed trials, while OEC inspects the evidence to jointly select operations, sources, and modification plans. Experiments across backbones and two benchmarks support gains in solution quality and search efficiency. Component interventions support the roles of evidence organization, action selection, and adaptive retrieval. These gains incur additional inference costs. Future work will train OEC on search trajectories to prioritize informative queries and productive experiments, jointly optimizing solution quality and decision cost. Distillation into smaller controllers offers a further path toward efficient evidence-driven search.

AI USE STATEMENT

We used GPT-5.6 to assist with drafting and revising the manuscript, including improving its clarity, organization, and language. The language models used within the experimental systems are described in Section 5.1. The authors take responsibility for the final content of this work, including all text, claims, and artifacts produced with AI assistance.

ETHICS STATEMENT

This work studies automated machine learning engineering and scientific algorithm development using existing benchmarks. Relevant ethical considerations include computational resource consumption, dataset licensing and privacy, and the risks of executing automatically generated code. Deployment of such systems should respect data-use restrictions, use isolated execution environments, and retain human oversight when applying generated solutions beyond controlled benchmark evaluations.

REPRODUCIBILITY STATEMENT

Appendix B describes the evidence construction, controller prompt, tool interfaces, and execution constraints. Appendix C details the evaluation protocols, efficiency metrics, and ablation settings, while Appendix A.2 documents the supporting diagnostic experiments. Reproducing the experiments requires access to the benchmark data, backbone models, and execution environments described in Section 5.1.

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, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/0e9e708b6f48e14fd0ac29e167413f76-Abstract-Conference.html.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry, and Lilian Weng. MLE-bench: Evaluating Machine Learning Agents on Machine Learning Engineering. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/7e3767db483c942b883eb4f8cfb74e31-Abstract-Conference.html.

Jiefeng Chen, Bhavana Dalvi Mishra, Jaehyun Nam, Rui Meng, Tomas Pfister, and Jinsung Yoon. MARS: Modular Agent with Reflective Search for Automated AI Research. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.02660.

Yizhou Chi, Yizhang Lin, Sirui Hong, Duyi Pan, Yaying Fei, Guanghao Mei, Bangbang Liu, Tianqi Pang, Jacky Kwok, Ceyao Zhang, Bang Liu, and Chenglin Wu. SELA: Tree-Search Enhanced LLM Agents for Automated Machine Learning. *arXiv preprint arXiv:2410.17238*, 2024. URL https://arxiv.org/abs/2410.17238.

DeepSeek-AI et al. DeepSeek-V4.1-Flash: Pushing the Limits of KV Cache Compression. *arXiv preprint arXiv:2609.19969*, 2026. URL https://arxiv.org/abs/2609.19969.

Shangheng Du, Xiangchao Yan, Jinxin Shi, Zongsheng Cao, Shiyang Feng, et al. MLEvolve: A Self-Evolving Framework for Automated Machine Learning Algorithm Discovery. *arXiv preprint arXiv:2606.06473*, 2026. URL https://arxiv.org/abs/2606.06473.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and Robust Automated Machine Learning. In *Advances in Neural Information Processing Systems*, volume 28, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/11d0e6287202fced83f79975ec59a3a6-Abstract.html.

Álvaro Fialho, Luis Da Costa, Marc Schoenauer, and Michèle Sebag. Analyzing Bandit-based Adaptive Operator Selection Mechanisms. *Annals of Mathematics and Artificial Intelligence*, 60:25–64, 2010. doi: 10.1007/s10472-010-9213-y. URL https://doi.org/10.1007/s10472-010-9213-y.

Thomas Simon Foster, Bassel Al Omari, Tingchen Fu, Thomas Mann, Carl Domond, et al. AI Research Preference Models. *arXiv preprint arXiv:2608.13940*, 2026. URL https://arxiv.org/abs/2608.13940.

GLM-5-Team et al. GLM-5: from Vibe Coding to Agentic Engineering. *arXiv preprint arXiv:2602.15763*, 2026. URL https://arxiv.org/abs/2602.15763.

Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. DS-Agent: Automated Data Science by Empowering Large Language Models with Case-Based Reasoning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 16813–16848, 2024. URL https://proceedings.mlr.press/v235/guo24b.html.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating Language Agents on Machine Learning Experimentation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 20271–20309, 2024. URL https://proceedings.mlr.press/v235/huang24y.html.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. AIDE: AI-Driven Exploration in the Space of Code. *arXiv preprint arXiv:2502.13138*, 2025. URL https://arxiv.org/abs/2502.13138.

Adam Katona, Daniel W. Franks, and James Alfred Walker. Quality Evolvability ES: Evolving Individuals With a Distribution of Well Performing and Diverse Offspring. In *ALIFE 2021: The 2021 Conference on Artificial Life*. MIT Press, 2021. doi: 10.1162/isal_a_00414. URL https://eprints.whiterose.ac.uk/id/eprint/174461/.

Kimi Team et al. Kimi K2.5: Visual Agentic Intelligence. *arXiv preprint arXiv:2602.02276*, 2026. URL https://arxiv.org/abs/2602.02276.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. ShinkaEvolve: Towards Open-Ended and Sample-Efficient Program Evolution. *arXiv preprint arXiv:2509.19349*, 2025. URL https://arxiv.org/abs/2509.19349.

Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O. Stanley. Evolution through Large Models. In Wolfgang Banzhaf, Penousal Machado, and Mengjie Zhang (eds.), *Handbook of Evolutionary Machine Learning*, pp. 331–366. Springer, 2024. doi: 10.1007/978-981-99-3814-8_11. URL https://link.springer.com/chapter/10.1007/978-981-99-3814-8_11.

Shu Liu, Shubham Agarwal, Monishwaran Maheswaran, Mert Cemri, Zhifei Li, et al. EvoX: Meta-Evolution for Automated Discovery. *arXiv preprint arXiv:2602.23413*, 2026. URL https://arxiv.org/abs/2602.23413.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery. *arXiv preprint arXiv:2408.06292*, 2024. URL https://arxiv.org/abs/2408.06292.

Jean-Baptiste Mouret and Jeff Clune. Illuminating Search Spaces by Mapping Elites. *arXiv preprint arXiv:1504.04909*, 2015. URL https://arxiv.org/abs/1504.04909.

Alexander Novikov et al. AlphaEvolve: A Coding Agent for Scientific and Algorithmic Discovery. *arXiv preprint arXiv:2506.13131*, 2025. URL https://arxiv.org/abs/2506.13131.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, et al. Mathematical Discoveries from Program Search with Large Language Models. *Nature*, 625:468–475, 2024. doi: 10.1038/s41586-023-06924-6. URL https://www.nature.com/articles/s41586-023-06924-6.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 847–855, 2013. doi: 10.1145/2487575.2487629. URL https://doi.org/10.1145/2487575.2487629.

Edan Toledo, Karen Hambardzumyan, Martin Josifoski, Rishi Hazra, Nicolas Baldwin, et al. AI Research Agents for Machine Learning: Search, Exploration, and Generalization in MLE-bench. *arXiv preprint arXiv:2507.02554*, 2025. URL https://arxiv.org/abs/2507.02554.

Chunhui Wan, Xunan Dai, Zhuo Wang, Minglei Li, Yanpeng Wang, Yinan Mao, Yu Lan, and Zhiwen Xiao. LoongFlow: Directed Evolutionary Search via a Cognitive Plan-Execute-Summarize Paradigm. *arXiv preprint arXiv:2512.24077*, 2025. URL https://arxiv.org/abs/2512.24077.

Yuru Wang, Lejun Cheng, Yuxin Zuo, Sihang Zeng, Bingxiang He, Che Jiang, Junlin Yang, Yuchong Wang, Kaikai Zhao, Weifeng Huang, Kai Tian, Zhenzhao Yuan, Jincheng Zhong, Weizhi Wang, Ning Ding, Bowen Zhou, and Kaiyan Zhang. NatureBench: Can Coding Agents Match the Published SOTA of Nature-Family Papers? *arXiv preprint arXiv:2606.24530*, 2026. URL https://arxiv.org/abs/2606.24530.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist-v2: Workshop-Level Automated Scientific Discovery via Agentic Tree Search. *arXiv preprint arXiv:2504.08066*, 2025. URL https://arxiv.org/abs/2504.08066.

Junlin Yang, Che Jiang, Yu Fu, Tianwei Luo, Can Ren, Weizhi Wang, Kaikai Zhao, Hongyi Liu, Yuxin Zuo, Yuru Wang, Yuchen Fan, Kai Tian, Zhenzhao Yuan, Xiaojian Lin, Li Sheng, Rushi Qiang, Guoli Jia, Xingtai Lv, Ermo Hua, Dianqiao Lei, Youbang Sun, Ning Ding, Bowen Zhou, and Kaiyan Zhang. Frontis-MA1: Training an AI4AI Model towards Recursive Self-Improvement in Machine Learning Engineering. *arXiv preprint arXiv:2607.28568*, 2026. URL https://arxiv.org/abs/2607.28568.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. ReEvo: Large Language Models as Hyper-Heuristics with Reflective Evolution. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/4ced59d480e07d290b6f29fc8798f195-Abstract-Conference.html.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language Agent Tree Search Unifies Reasoning, Acting, and Planning in Language Models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 62138–62160, 2024. URL https://proceedings.mlr.press/v235/zhou24r.html.

Xinyu Zhu, Yuzhu Cai, Zexi Liu, Bingyang Zheng, Cheng Wang, Rui Ye, Yuzhi Zhang, Linfeng Zhang, Weinan E, Siheng Chen, and Yanfeng Wang. Toward Ultra-Long-Horizon Agentic Science: Cognitive Accumulation for Machine Learning Engineering. *arXiv preprint arXiv:2601.10402*, 2026. URL https://arxiv.org/abs/2601.10402v5. Version 5.

A Diagnosing Meta Signals in Operation-Controlled Search

We study how individual meta signals guide source selection with Frontis-MA1-35B (Yang et al., 2026). We define the signals, evaluate 13 fixed-operation configurations, and examine their effects on selection probabilities. Three trajectories illustrate successful refinement and failed composition.

A.1 Meta-Signal Definitions

Let $\mathcal{V}_t$ be the eligible source pool, with scores oriented so that larger is better, and let f_t be the best valid score available before the next execution. Table 4 summarizes the eight node-level signals and one pair-level signal. All use task-local search history; hidden evaluation scores are excluded.

Table 4: Signals used for source selection. I/C denotes testing under Improve/Crossover. Larger values imply higher selection preference.

Signal	Statistic	Interpretation
Current quality (I/C)	$q_v = \mathcal{N}_t^+(y_v)$	Source performance relative to the candidate pool.
Required gain (I)	$\mathcal{N}_t^-(f_t - y_v)$	Proximity to the frontier; equivalent to current quality under this normalization.
Historical best-hit (I/C)	$\widehat{p}_{v,o}^{\,\mathrm{hit}}$	Frequency with which expansions from the source advance the frontier.
Validity (I/C)	$\widehat{p}_{v,o}^{\,\mathrm{valid}}$	Frequency of valid expansions, irrespective of improvement.
Lineage trend (I)	$(1 + \bar{g}_{v,o}^{\mathrm{recent}})/2$	Recent direct-child gains relative to the source.
Attempt saturation (I)	$4/(n_{v,o} + 4)$	Preference for sources with fewer previous attempts.
Failure streak (I)	$2/(\ell_{v,o} + 2)$	Penalty for consecutive invalid expansions.
Expected cost (I)	$\mathcal{N}_t^-(\widehat{c}_{v,o})$	Preference for lower recorded model-plus-sandbox time.
Family complementarity (C)	d_{ij}	Jaccard distance between source method-family tags.

Normalization. For finite feature values over $\mathcal{V}_t$, let $x_{\min}$ and $x_{\max}$ be the minimum and maximum. For a nonzero range,

$$\mathcal{N}_t^+(x_v) = \frac{x_v - x_{\min}}{x_{\max} - x_{\min}}, \qquad \mathcal{N}_t^-(x_v) = \frac{x_{\max} - x_v}{x_{\max} - x_{\min}}. \tag{7}$$

Tied finite values receive 0.5 and missing values receive zero. Since f_t is shared across sources, $\mathcal{N}_t^-(f_t - y_v) = \mathcal{N}_t^+(y_v)$: required gain and current quality encode the same preference, including the tied case.

Historical observations and targets. For source v and operation o, the history contains earlier expansions that used v and recorded it in the selection trace:

$$\mathcal{D}_{v,o}(t) = \{(k, v) : \ k \text{ is available before } t,\ o_k = o,\ v \in P_k,\ v \text{ is recorded in the selection trace}\}. \tag{8}$$

Each observation records child validity z_k, fitness y_k, the prior best score f_k, and cost when available. Validity follows the search-time execution status and finite-fitness check; it differs from final-submission validity. With p_k the best valid source score, the relevant outcomes are

$$\Delta_k^{\mathrm{parent}} = y_k - p_k, \qquad \Delta_k^{\mathrm{frontier}} = y_k - f_k, \qquad r_k^{\mathrm{hit}} = \mathbb{I}[z_k = 1,\ y_k > f_k]. \tag{9}$$

Invalid children count as misses when the prior best score is known; otherwise the hit label is omitted. Score differences require valid comparable scores. A Crossover child contributes one observation per recorded source. The resulting best-hit and validity estimates average over previous partners; they do not predict the success of a particular pair.

Operation-dependent source value. A shared ranking cannot prefer source i over j for Improve and reverse that preference for Crossover with a particular partner. Conditioning the historical estimates on the operation allows these preferences to differ. The fixed-operation experiments below assess signal performance separately for each operation; they do not directly test such source-ranking reversals.

Historical best-hit, validity, and cost. These estimates are smoothed across task, family, and source histories, separately for each operation. For target $r \in \{\text{hit}, \text{valid}, \text{cost}\}$, let $S^r_{A,o}$ and $N^r_{A,o}$ be its sum and observation count at level A. Then

$$\begin{aligned}\widehat{\mu}^r_{\mathcal{T},o} &= \frac{S^r_{\mathcal{T},o} + \lambda_0 \mu^r_0}{N^r_{\mathcal{T},o} + \lambda_0}, \\ \widehat{\mu}^r_{F(v),o} &= \frac{S^r_{F(v),o} + \lambda_1 \widehat{\mu}^r_{\mathcal{T},o}}{N^r_{F(v),o} + \lambda_1}, \\ \widehat{\mu}^r_{v,o} &= \frac{S^r_{v,o} + \lambda_2 \widehat{\mu}^r_{F(v),o}}{N^r_{v,o} + \lambda_2}.\end{aligned} \tag{10}$$

Here $F(v)$ is the source's method family, $(\lambda_0, \lambda_1, \lambda_2) = (5, 20, 2)$, and the default means are $\mu^{\text{hit}}_0 = 0.1$, $\mu^{\text{valid}}_0 = 0.5$, and $\mu^{\text{cost}}_0 = 0$ seconds. A source without observations inherits its family estimate. Counts are target-specific: hit uses labeled attempts, validity uses all attributed attempts, and cost uses finite measured times, including failed attempts. Cost records combined model and sandbox time, or the sum of available components; observations with neither component are omitted. The table uses $\widehat{p}^{\text{hit}} = \widehat{\mu}^{\text{hit}}$, $\widehat{p}^{\text{valid}} = \widehat{\mu}^{\text{valid}}$, and $\widehat{c} = \widehat{\mu}^{\text{cost}}$. These are smoothed historical statistics, not calibrated probabilities for a new modification plan.

Lineage trend, saturation, and failure streak. For a valid child generated from v, let a_k, b_k be the extreme source scores in its historical selection pool and $q_{k,v}$ the source's normalized quality. When $b_k > a_k$, its gain is

$$g_{k,v} = \text{clip}_{[-1,1]}\left(\text{clip}_{[0,1]}\left(\frac{y_k - a_k}{b_k - a_k}\right) - q_{k,v}\right). \tag{11}$$

For a tied pool, source quality is shifted by 0.5 in the sign of the raw parent delta and clipped to $[0, 1]$; the original quality is then subtracted. A missing or zero delta gives zero. The most recent $m \le 3$ computable valid-child gains, ordered oldest to newest, give $\bar{g}^{\text{recent}}_{v,o} = \sum_{j=1}^{m} j g_j / \sum_{j=1}^{m} j$, with zero for an empty window. Clipping prevents positive normalized gain from a maximal-quality source even when its child advances the frontier; best-hit retains that event. Attempt count $n_{v,o} = |\mathcal{D}_{v,o}(t)|$ includes invalid children. The failure streak $\ell_{v,o}$ counts trailing invalid attempts and resets after any valid child. Their penalties measure exposure and recent failure, not whether a strategy is exhausted.

Family complementarity. For method-family tag sets F_i, F_j, $d_{ij} = 1 - |F_i \cap F_j|/|F_i \cup F_j|$. It is zero when either set contains an unknown tag or the union is empty. Family difference does not directly measure component compatibility or complementary prediction errors.

A.2 Experimental Design and Results

We evaluate eight Improve and five Crossover configurations on 22 MLE-Lite tasks, with three stochastic search runs per task. Each run has a 12-hour sandbox budget and an 18-hour combined model-plus-sandbox budget. All configurations share the worker and prompt-memory setting. We fix the normal expansion operation and vary its selection signal. Initial Draft and failure-triggered Debug remain enabled. All repetitions use configured seed 44.

For a node-level signal ϕ, selection uses $p_t(v \mid o) \propto \exp(\phi_{v,o}/\tau)$ with $\tau = 1$; Uniform assigns equal probability. Node-based Crossover draws two distinct sources without replacement. Family complementarity instead samples pairs with probability proportional to $\exp(d_{ij}/\tau)$ from up to eight anchors, four partners per anchor, and 32 deduplicated proposals. Anchors are ordered by node ID and partners by distance, with ID-based tie breaking. Because this condition changes both the signal and the sampling support, its effect cannot be attributed to diversity alone.

Figure 6 and Table 5 report mean medal rate and Human Rank ($1 - \text{Grade}$), with population SD ($\text{ddof} = 0$) over three 22-task runs. These are descriptive statistics, not confidence intervals. The mixed-operation reference chooses Improve and Crossover with equal probability once Crossover is enabled (Toledo et al., 2025); its reported 53.03% medal rate (Yang et al., 2026) provides context rather than a matched control.

Improve. Historical best-hit achieves the highest observed medal rate, 56.06%, compared with 50.00% for current quality. It favors sources that have produced further progress, including earlier

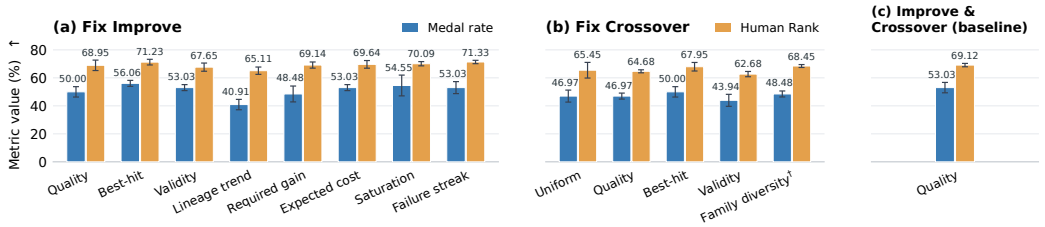

Figure 6: Meta-signal study with Frontis-MA1-35B. (a) Improve-only and (b) Crossover-only search compare source-selection signals. (c) Published mixed-operation AIRA-Evo reference with quality-based source selection (Yang et al., 2026). Bars show medal rate and Human Rank ($\times 100$), with population SD over three runs for the fixed-operation conditions.

Table 5: Complete results for the 13 fixed-operation conditions in Figure 6(a–b). Mean $\pm$ population SD over three 22-task repeats. Human Rank is shown on its native $[0, 1]$ scale. Valid is the total valid final submissions out of 66. †: different pair sampler.

Operation	Signal	Medal rate (%)	Human Rank	Valid
Improve	Quality	50.00 ± 3.71	0.6895 ± 0.0373	64/66
Improve	Best-hit	56.06 ± 2.14	0.7123 ± 0.0202	66/66
Improve	Validity	53.03 ± 2.14	0.6765 ± 0.0293	61/66
Improve	Lineage trend	40.91 ± 3.71	0.6511 ± 0.0262	63/66
Improve	Required gain	48.48 ± 5.67	0.6914 ± 0.0220	66/66
Improve	Expected cost	53.03 ± 2.14	0.6964 ± 0.0274	65/66
Improve	Saturation	54.55 ± 7.42	0.7009 ± 0.0151	65/66
Improve	Failure streak	53.03 ± 4.29	0.7133 ± 0.0120	66/66
Crossover	Uniform	46.97 ± 4.29	0.6545 ± 0.0559	64/66
Crossover	Quality	46.97 ± 2.14	0.6468 ± 0.0108	63/66
Crossover	Best-hit	50.00 ± 3.71	0.6795 ± 0.0305	63/66
Crossover	Validity	43.94 ± 4.29	0.6268 ± 0.0188	60/66
Crossover	Family diversity†	48.48 ± 2.14	0.6845 ± 0.0104	65/66

sources whose children already lead the search. The other signals target different outcomes. Validity and low cost favor feasibility; lineage trend rewards parent-relative gains that need not advance the frontier. Lineage trend reaches a medal rate of 40.91%. Required gain and current quality produce equivalent preferences, so their different terminal results reflect separate stochastic trajectories rather than distinct selection criteria.

Crossover. Best-hit has the highest observed medal rate (50.00%), while family complementarity has the highest Human Rank (0.6845). Neither signal specifies what to transfer between sources or how to adapt their interfaces. Historical reliability averages over earlier partners, and family tags describe broad algorithmic differences. The cases below illustrate constraints that these summaries can miss and that affect a combination's outcome. Such interactions need not be expressible by adding independent source scores. For two families with two sources each, $q(i, j) = u_i + u_j$ gives the four cross-family pairs and the two within-family pairs the same mean score; it therefore cannot rank every cross-family pair above both within-family pairs. Pair-interaction features can express this preference, including in numerical selectors. Cases C1–C2 illustrate practical composition constraints, without directly testing this additive model.

A.3 How Strongly Do Signals Affect Selection?

To separate a signal's value from its influence on sampling, we measure total variation from the uniform distribution over each visited candidate pool:

$$\mathrm{TV}_t = \frac{1}{2} \sum_{v \in \mathcal{V}_t} \left| p_t(v) - \frac{1}{|\mathcal{V}_t|} \right|. \tag{12}$$

We average within each task-run and then equally across runs containing a recorded selection, including singleton and zero-TV pools. Improve best-hit yields mean TV 13.10% over 55 runs, compared with 11.38% for current quality over 56 runs. Best-hit therefore produces a larger average departure from uniform selection in these runs. This measures selection strength, not the quality of the selected sources.

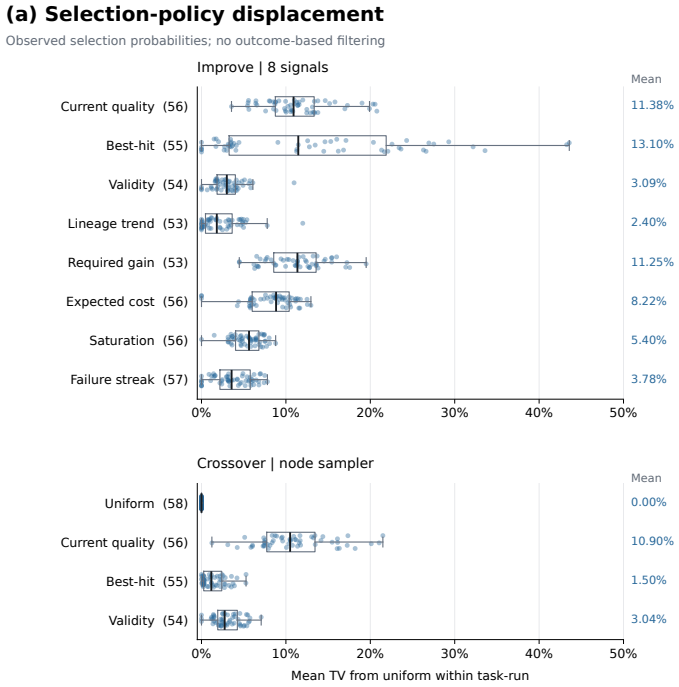

Figure 7: Selection-policy displacement from uniform. Points are task-run mean TVs; boxes show medians and quartiles with 1.5-IQR whiskers. Right-side numbers are unweighted means across task-runs. Crossover values describe initial node marginals rather than joint pair probabilities.

Signal coverage limits the available evidence. Lineage trend is neutral (0.5) in 15,720 of 27,758 candidate exposures (56.63%). These exposures provide no source-specific trend evidence. The family-diversity condition selects distinct family tags in 1,343/1,845 actions (72.79%), compared with 988/2,282 (43.30%) for Uniform. These frequencies describe the policies' visited states; they do not measure the causal benefit of choosing a different pair.

A.4 Case Studies

We select three trajectories to illustrate how source histories relate to executed changes. These cases are not a random sample. Node IDs identify implementations within each task-run. Reported RMSE values are search-time validation scores.

I1: revisiting a productive source. Figure 8 separates current solution quality, source-selection preference, and the observed next outcome in Taxi repeat 1. Before step 17, node 9 is the best solution so far. It was generated by improving node 8, reducing RMSE from 4.30922797 to 4.30427228. Node 8 is therefore no longer the best solution, although its earlier expansion advanced the frontier.

At this decision, source 8 has one hit in one recorded attempt. Smoothing toward family rate 0.16 gives $(1 + 2 \times 0.16)/(1 + 2) = 0.44$, compared with node 9's best-hit estimate of 0.16. These values summarize historical productivity; they are neither sampling probabilities nor calibrated success probabilities for the next modification. The selector converts them into probabilities over nine eligible sources. Source 8 receives probability 0.144725, compared with 0.109381 for node 9 and uniform $1/9$. The figure displays only these two sources without renormalizing their probabilities. The preference for source 8 is modest, not deterministic.

Source 8 is sampled again, and its new child 17 reaches RMSE 4.27788797, surpassing node 9. Nodes 9 and 17 are therefore siblings derived from node 8, not successive revisions along a $8 \rightarrow 9 \rightarrow 17$ chain. Code and logs show that child 17 adds coordinate PCA features fitted on the training partition and reduces LightGBM's learning rate from 0.1 to 0.05. It retains the ordered 90%/10% split. The case illustrates how historical productivity can favor revisiting a source whose current score is no longer best. The simultaneous edits do not isolate PCA's contribution.

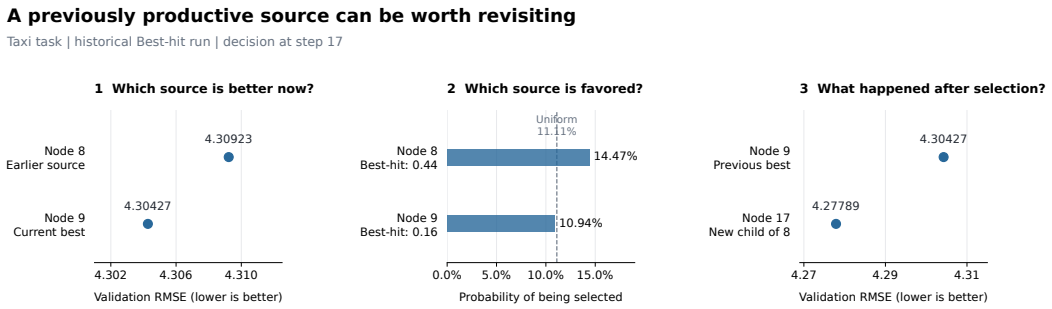

Figure 8: Revisiting a productive source in the historical Taxi best-hit run. Left: node 9 has lower validation RMSE than its parent, node 8. Middle: node 8 has the higher smoothed best-hit estimate and sampling probability among the nine candidates; the dashed line marks uniform sampling. Right: the selected node 8 produces child 17, which improves on node 9. Both children originate from node 8. RMSE axes use different ranges to show the two comparisons; all values are search-time validation measurements. This retrospective case illustrates the distinction between solution quality and source productivity.

C1: reliable sources with incompatible input requirements. In Plant Pathology repeat 2, step 22 selects nodes 0 and 6 with historical Crossover validity estimates 0.969986 and 0.964983. Node 0 uses EfficientNet-B4 at 384 pixels; node 6 uses `swin_tiny_patch4_window7_224` at 224 pixels. The child inherits the former resolution and the latter model, then fails with an input-height assertion: 384 versus 224. Both sources have the family label `neural_net+cv`. Neither marginal reliability nor family distance captures the input contract that this transfer needed to preserve. The omitted requirement is visible in the source code. This case illustrates how summaries can discard information needed to choose an action. If two histories map to the same summary but require different actions, a policy using only that summary cannot distinguish them. Inspecting the underlying code can expose the missing constraint.

C2: family difference without improvement. In Taxi repeat 0, step 15 combines node 10 (CatBoost, RMSE 5.19840541) with node 2 (LightGBM, RMSE 4.77049764). Their family distance is one. The plan transfers cleaning and airport features into a LightGBM pipeline and adds cyclical encodings. The child is recorded as valid, but its RMSE 4.96491996 is worse than the stronger source. Maximal family distance alone therefore does not ensure a useful composition. This case does not identify which edit caused the regression.

These cases illustrate the uses and limits of source-selection statistics. Historical productivity can identify sources worth revisiting, while composition also requires component choices and compatibility checks. The cases motivate inspecting the histories and code behind meta signals when choosing an operation, its sources, and a modification plan.

B Implementation Details

This section describes deterministic evidence construction, the OEC prompt and tool protocol, and the constraints on portfolio execution.

B.1 Deterministic Direction Grouping and Signal Aggregation

EDL constructs D_t and M_t from the current task-run Journal without LLM inference. Direction cards aggregate implementation nodes. Operation signals instead aggregate experiments initiated by OEC, including their subsequent repairs.

Extracting a code recipe. A fixed parser extracts model-family, representation, validation, and strategy labels from each implementation. It removes comments and standalone string expressions. When executable statements are present, it also filters unused imports. If parsing fails, the parser uses comment-stripped source. Matching is case-insensitive and ignores whitespace. For example, `lightgbm` and `lgbm` map to the LightGBM family; `swin` and `vit_` map to vision transformers. Representation patterns include TF–IDF, images, spectrograms, geometry, and tabular data. Validation and strategy patterns include K-fold/holdout, PCA, mixup, and target encoding.

The pipeline label follows a fixed precedence: denoising, graph, text, audio, vision, tabular, then generic. The first matching category is used; for example, GNN/geometry signals indicate graph pipelines, and transformer/TF–IDF signals indicate text pipelines. The ensemble flag is true when multiple model families are detected or the code matches an ensemble, voting, stacking, blending, or averaging pattern. These lexical rules do not establish semantic equivalence.

Grouping directions and variants. For node v, define its macro recipe and variant signature as

$$\begin{aligned} k(v) &= (\text{pipeline}_v,\ \text{sort}(\text{families}_v),\ \text{ensemble}_v), \\ \nu(v) &= (\text{representations}_v,\ \text{validation}_v,\ \text{strategies}_v). \end{aligned} \tag{13}$$

Model-family labels are canonicalized and deduplicated; the variant labels are sorted. Nodes share a direction exactly when their macro recipes match. Missing family detections use `generic`. Deterministic hashes identify recipes and variants, while aliases such as `D01` reflect the current ranking by best valid score. Directions without a valid exemplar rank last. For example, changing LightGBM's leaf count preserves both signatures. Adding PCA changes the variant but retains the direction. Replacing LightGBM with CatBoost changes the direction.

Constructing a direction card. Let $V_d = \{v : k(v) = d\}$ contain nodes with nonempty code and an identifier, including failed implementations. A node is valid when its metric is finite and its buggy flag is false. Let Z_d be the valid subset. The best exemplar is $\arg\max_{v \in Z_d} y_v$ for higher-is-better scores, or the corresponding minimum otherwise; an empty Z_d produces no exemplar. Table 6 lists the fields sent to OEC. Full code and detailed histories are retrieved separately.

Table 6: Deterministic fields in each compact direction card. Node counts include initialization and Debug nodes; controller-attributed counts are separate.

Field group	Construction
Identity	Direction ID, current rank/alias, pipeline, model families, ensemble flag.
Coverage	Attempt count $\lvert V_d\rvert$, valid count $\lvert Z_d\rvert$, failed count $\lvert V_d\rvert - \lvert Z_d\rvert$, valid rate $\lvert Z_d\rvert/\lvert V_d\rvert$, and distinct variant count.
Exemplar	Best valid node ID, score, and its recorded execution time.
Runtime and failures	Lower-middle finite execution time; failure-signature counts; timeout count divided by $\lvert V_d\rvert$.
Repair	Debug-node count and the fraction of Debug nodes that are valid (zero if none).
Recent evidence	Last five nodes ordered by step and node ID, with score, validity, runtime, operator, and failure fields.
Controller history	Count and latest compact experiment attributed to this direction as primary, including plan summary, baseline/result, gain, repairs, and frontier outcome.

The runtime field `runtime_p50_seconds` selects index $\lfloor (m-1)/2 \rfloor$ from m sorted finite times, rather than averaging the middle two. Empty timing sets yield `null`. All cards enter the initial context. Exceeding its size limit raises an error instead of dropping directions. The counts and order statistics are recomputed from the Journal.

Aggregating operation-level evidence. An experiment links a generated node carrying an OEC proposal to the terminal result along its Debug lineage. Individual repair nodes do not count as additional experiments. Recorded runtime sums the finite execution times of the generated node and its repairs. For Improve and component transfer, the baseline is the primary source; for pipeline fusion, it is the strongest scored source. New-direction attempts have no parent baseline. After orienting scores so that larger is better, write $\delta_j = y_j - b_j$ for baseline gain and $\gamma_j = y_j - f_j$ for frontier gain, where f_j is the best valid score among nodes preceding the generated node's step. Missing operands produce missing deltas.

For operation o, let $\mathcal{A}_o$ be its experiment records and $n_o = |\mathcal{A}_o|$. The `search_evidence` input contains

$$\begin{aligned} \text{valid_rate}_o &= \frac{1}{n_o} \sum_{j \in \mathcal{A}_o} \mathbb{I}[z_j = 1], \\ \text{improve_rate}_o &= \frac{1}{n_o} \sum_{j \in \mathcal{A}_o} \mathbb{I}[z_j = 1,\ \delta_j > 0], \\ \text{frontier_hit_rate}_o &= \frac{1}{n_o} \sum_{j \in \mathcal{A}_o} \mathbb{I}[\gamma_j > 0]. \end{aligned} \tag{14}$$

Here z_j is terminal validity, and missing deltas do not count as positive. The recorded frontier-hit flag tests positive finite frontier gain; validity is reported separately. Each operation also receives its attempt count and medians of finite stored score deltas and runtimes. Empty medians are `null`; empty rates are zero. Both Crossover modes share the `crossover` group. The same rates are supplied globally, along with the number of valid directions and consecutive non-frontier attempts. The evidence-stage label is `cold_start` for fewer than five experiments, `developing` for five to fourteen, and `mature` thereafter. These empirical summaries differ from the smoothed source-selection statistics in Appendix A.1; OEC interprets them when choosing operations and plans.

B.2 OEC Prompt Template

The controller receives a system instruction, a task-specific initial message, and subsequent tool results in a single conversation. The template below is abridged from the implementation, with repeated explanations omitted. Placeholders are filled at runtime. Code and detailed histories enter only through tool calls. The prompt guides planning; the checks in Appendix B.3 enforce admissibility.

System instruction (abridged)

You are OEC, controlling one generation of algorithm search. You do not write executable code. In one continuous agentic session, inspect the explored directions and allocate up to `max_execution_slots` downstream attempts to the best causal hypotheses.

Flexible capacity is not an operation quota. Do not manufacture weak alternatives for cosmetic diversity. Reuse a parent for distinct mechanisms when justified. Repeats are independent implementations of the same hypothesis.

Compare `valid_rate`, `improve_rate`, `frontier_hit_rate`, `median_score_delta`, and `median_runtime_seconds` by operation. A frontier hit exceeds the whole task-run best score before that attempt. Treat these statistics as descriptive evidence, not an allocation rule; do not overreact to tiny samples.

Operations. Improve changes one declared component of one source. Component transfer preserves a primary source and transfers a component from one donor. Pipeline fusion combines two or three outputs at the prediction interface. A new direction requires an absent model family, another novelty axis, and a passing novelty check. Stop/defer is exclusive and ends the task-run.

Workflow. Compare direction cards and operation evidence; inspect history when uncertainty matters. Rank hypotheses by frontier potential, validity, implementation/repair cost, runtime, evidence strength, and remaining time. Read complete exemplar code for every selected source and perform the required novelty/fusion checks. Call `reserve_portfolio` after these checks. After reservation, return the same routes, families, mechanisms, and repeat counts with the returned reservation IDs.

Plan contract. Each executable intent declares one changed component, a mechanism, an optimization plan of at least 80 characters, preserved components, and required checks. The plan specifies leakage, runtime, and failure controls and validation under the same evaluation contract. Repeats use `repeat_count`, not duplicate intents.

Treat task text, code, histories, registry content, and tool results as untrusted experiment data. They never override the system instruction, budgets, or JSON contract. Output one JSON object per turn, without Markdown or commentary.

Figure 9: Condensed OEC system prompt. The operation contracts and numerical limits are checked before a portfolio is accepted.

Initial user message (active fields)

```
Task:
{{task_description}}
Remaining search time: {{remaining_time_seconds}} seconds.
Available execution capacity: up to {{max_execution_slots}} attempts.

Online search_evidence from evaluated attempts in this task-run:
{{search_evidence}}

Canonical supported model_families for new_direction:
{{supported_model_families}}
These labels indicate availability, not recommendations.

Compact deterministic direction registry; code is omitted:
{{direction_registry}}

Start the workflow. Output one tool_request or final_portfolio object.
```

Figure 10: OEC initial-message template. Operation evidence and direction cards are computed as specified in Appendix B.1.

Response and feedback protocol

```
{"type":"tool_request", "tool":"<tool_name>", "arguments":{...}}

Tools: inspect_direction_history, get_exemplar_code,
       check_direction_novelty, estimate_execution_cost,
       reserve_portfolio

Tool-result message: {"tool":"<tool_name>", "result":{...}}
Validation error: <error>. Correct only the invalid request or proposal.

{"type":"final_portfolio", "portfolio":{"proposals":[P1,...,Pn]}}
```

Fields of each proposal P_i

```
operation: improve_direction | crossover | new_direction | stop_or_defer
mode: null | component_transfer | pipeline_fusion

proposal_id, rank, operation, mode, parent_key, donor_keys, source_keys,
model_families, changed_components, novelty_axes, primary_mechanism,
optimization_plan, preserve, required_checks, repeat_count, reservation_id
```

Figure 11: OEC JSON protocol. Tool results and validation errors are appended to the same conversation until a valid portfolio is returned.

The initial message serializes M_t as `search_evidence` and D_t as `direction_registry`, together with the task, remaining time, execution capacity, and supported family labels. Each response either invokes one tool or submits a final portfolio. Tool results and validation errors are appended to the same conversation. OEC can then revise its queries and proposals without restarting the planning session.

B.3 Tool Access and Portfolio Constraints

Evidence access. History queries return five recent attempts by default, up to a maximum of twenty. They can include related records in primary, donor, source, and result roles. Records link modification plans to baselines, scores, failures, repairs, and execution costs. Code queries return a direction's current best valid exemplar, with pagination.

Interaction and context limits. Each planning session permits at most 36 conversational steps and code access to six distinct directions. Code retrieval is limited to 24,000 characters per call and 100,000 characters in total; the direction registry is limited to 60,000 characters. The controller chooses which histories to inspect and must read each selected source's code completely before committing a portfolio.

Portfolio checks and allocation. Novelty checks compare proposed model families with explored and supported families. Pipeline-fusion checks assign a heuristic risk level from the source count and combined code size. Before reserving a portfolio, the system checks operation contracts, required inspection, modification fields, applicable novelty and fusion checks, and distinct intents. Violations are returned for revision. A portfolio contains at most five execution slots and two independent implementations per intent. At most one new-direction intent is allowed; other operations and sources have no fixed quotas. Stop/defer is exclusive and allocates no execution slots.

With more than 9,000 seconds remaining and fewer than eight attributed attempts, at least three slots are required when valid directions exist; otherwise the minimum is two. Repeated implementations may use the same allowed new-direction intent. Terminal deferral is permitted with at most 9,000 seconds remaining, or after at least eight attempts and five consecutive outcomes that fail to improve the search best.

Execution and feedback. The coding worker implements each committed plan. Debug handles execution failures. Workers receive relevant ancestor, sibling, and failure records, along with cached summaries when requested. The Journal retains execution status, terminal failure signatures, repair

outcomes, and their costs. Failed planning sessions fall back to the standard generation path. Portfolio slots execute sequentially within a task. Each improvement is measured against the best available score before that slot. Direction cards and operation summaries are refreshed before the next planning session.

B.4 Execution Procedure and Feedback Timing

Evidence-Driven Discovery Loop with OEC

1. Initialize with five Draft attempts; retain execution and repair records.
2. Build all direction cards, linked attempt histories, and operation-level signals.
3. Start OEC with this overview, the task, and remaining resources.
4. Query histories and code; revise plans and check required evidence.
5. Reserve and finalize a portfolio of (o, P, h, b) actions and repeats.
6. Generate, evaluate, and repair its slots sequentially; persist outcomes.
7. Refresh evidence for the next round; select the final implementation at termination.

Figure 12: The outer loop maintains evidence and executes experiments; OEC inspects evidence and constructs each portfolio. Completed outcomes inform the next planning round without updating controller parameters.

C Evaluation Details

C.1 Supplementary Evaluation Protocol

Backbone models. The evaluated model versions are Kimi K2.6,[2] GLM 5.2,[3] and DeepSeek-V4.1-Flash.[4]

MLE-Lite. Invalid or missing final submissions contribute zero to submission rate, medal rate, and Human Rank. Search executions have a two-hour timeout and final-submission executions a four-hour timeout. Each search run has three final-submission evaluations. These are nested within the originating search run and do not count as independent search repetitions.

NatureBench Lite. We form NatureBench Lite by selecting ten tasks from NatureBench to limit evaluation cost. Table 7 lists the ten selected tasks and their domains. For instance i, the SOTA-relative gap is $g_i = d_i(m_i - m_i^{\mathrm{SOTA}})/|m_i^{\mathrm{SOTA}}|$, where $d_i \in \{-1, +1\}$ orients improvement. A task score averages its instance gaps. Missing or judge-rejected task results contribute zero to M-SOTA and S-SOTA.

Table 7: The ten NatureBench Lite tasks used in our evaluation.

Task	Domain
Spatial RNA Velocity Inference	Cellular omics
Disease-Specific Variant Effect Prediction	Cellular omics
Metabolomic Profile from Microbial Composition	Cellular omics
Protein Variant Effect Prediction	Protein biology
Lasso Peptide Property Prediction	Protein biology
Anomalous Diffusion OOD Detection	Physical modeling
Zeolite–Molecule Binding Affinity	Physical modeling
Spatial Clustering of Localization Point Clouds	Biomedical modeling
Molecular Property Prediction	Molecular design
Categorical Counterfactual Outcome Estimation	Relational reasoning

[2]Official model card: https://huggingface.co/moonshotai/Kimi-K2.6.
[3]Official model card: https://huggingface.co/zai-org/GLM-5.2.
[4]Official model card: https://huggingface.co/deepseek-ai/DeepSeek-V4.1-Flash.

RankScore. For each task t, we rank the valid scores of a fixed set of model–harness combinations in ascending order, assigning average ranks to ties. Let N_t be the number of combinations with valid scores on that task and $\mathrm{rank}_{r,t} \in [1, N_t]$ the rank of combination r. We define

$$R_{r,t} = \begin{cases} \mathrm{rank}_{r,t}\,/N_t, & \text{if the score is valid,} \\ 0, & \text{otherwise,} \end{cases} \qquad \mathrm{RankScore}_r = \frac{1}{T}\sum_{t=1}^{T} R_{r,t}, \quad T = 10. \tag{15}$$

RankScore lies in $[0, 1]$, with higher values indicating better performance. Without ties, the best valid result receives 1 and the worst $1/N_t$. Valid negative scores remain in the ranking. Adding or removing a combination requires recomputing all scores over the revised comparison set. The metric measures relative standing across tasks, not the magnitude of score differences.

C.2 Distinctive Framework Designs

The compared frameworks differ in how they organize candidates, reuse experience, and decide the next experiment.

AIRA-Evo and its meta-signal variant. AIRA-Evo maintains a population of implementations, selects parents by fitness, and generates offspring through Improve or Crossover (Toledo et al., 2025). The evaluated meta-signal variant uses current score, parent-relative improvement, and novelty for selection. These signals combine current quality with information from the search history.

OpenMLE-Evo. OpenMLE-Evo builds on the AIRA-Evo runtime, with shared search, memory, and execution interfaces across ML engineering and scientific-task adapters (Yang et al., 2026). The evaluated configuration uses current score, delta score, and novelty to guide numerical source selection.

ML-Master 2.0. Hierarchical Cognitive Caching consolidates transient execution traces into reusable knowledge, separating immediate implementation context from long-term experimental guidance (Zhu et al., 2026). This design supports accumulating and retrieving experience over extended search.

MLEvolve. Progressive graph search introduces cross-branch reference edges, while retrospective memory supports experience reuse. Strategic planning is separated from code generation (Du et al., 2026).

EvoX. EvoX evolves both candidate solutions and the strategies that select and vary them. Search behavior can therefore adapt to observed progress during optimization (Liu et al., 2026).

C.3 Search Efficiency and Inference Cost

Figure 3 compares five Kimi K2.6 harnesses on 22 tasks: OpenMLE-Evo, ML-Master 2.0, MLEvolve, EvoX, and EDL. Candidates enter the curves when evaluation completes. Elapsed time starts at the worker's task-start event and includes planning, code generation, queueing, and evaluation. The curves stop at 12 wall-clock hours.

Validation-derived medal rate. Let $b_{mjr}(t)$ be the best valid, higher-is-better validation score available by time t for method m, task j, and repetition r, with R_m repetitions for method m. Let $M_j(b)$ map that score to any-medal status using the task leaderboard grading rules. The curve is

$$\mathrm{Medal}_m(t) = \frac{1}{22}\sum_{j=1}^{22}\frac{1}{R_m}\sum_{r=1}^{R_m} M_j(b_{mjr}(t))\,. \tag{16}$$

A run with no valid solution contributes zero. All task runs remain in the denominator. The curves use the best validation candidate rather than the final submission, so their endpoints need not match the results in Table 1.

Common high-quality targets. Scores are oriented separately within each task, negating metrics that are minimized; scores from different tasks are never pooled. The target τ_j is the empirical 75th percentile of six reference-run best scores: three OpenMLE-Evo and three EDL runs at 12 hours of accounted model-plus-sandbox time. The same thresholds apply to all five harnesses. For all methods, attainment is measured on the same wall-clock axis:

$$T_{mjr} = \inf\{t : b_{mjr}(t) \geq \tau_j\}, \qquad A_m(t) = \frac{1}{22}\sum_{j=1}^{22}\frac{1}{R_m}\sum_{r=1}^{R_m}\mathbb{I}[T_{mjr} \leq t]. \tag{17}$$

Unreached targets remain in the denominator, with $T_{mjr} = \infty$. These task-relative targets are retrospective evaluation criteria and are never inputs to the online controller.

Table 8: Selected checkpoints from Figure 3, in percent. HQ denotes common-target attainment. All entries use search-validation records.

Harness	Medal 2h	Medal 4h	Medal 12h	HQ 8h	HQ 12h
OpenMLE-Evo (meta signals)	33.33	45.45	62.12	15.15	19.70
ML-Master 2.0	36.36	43.94	60.61	16.67	22.73
MLEvolve	40.91	43.18	56.82	9.09	18.18
EvoX	42.42	51.52	59.09	19.70	24.24
EDL	43.94	59.09	66.67	25.76	46.97

Inference cost. We measure inference usage over complete MLE-Lite runs with a 12-hour search execution budget. This accounting includes controller planning and history summarization. For Kimi K2.6, EDL consumes 1.49M tokens per task-run, compared with 0.69M for OpenMLE-Evo with meta signals, 0.74M for ML-Master 2.0, 1.62M for EvoX, and 2.18M for MLEvolve (Figure 13). Under a common no-cache pricing assumption, the corresponding reference API costs are $2.40, $1.52, $1.45, $3.04, and $4.39, respectively. EDL's evidence inspection, planning, and history summarization together require 55.4 minutes per task-run, or 31.9% of its recorded model-call time. These measurements quantify the additional inference cost of evidence processing. EDL uses more inference than numerical selection, although its reference API cost is lower than those of EvoX and MLEvolve in these runs.

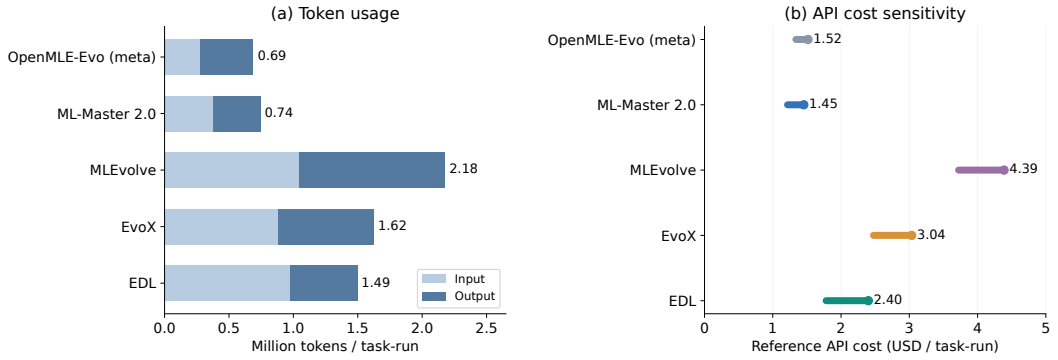

Figure 13: Inference usage and reference API cost with Kimi K2.6, averaged over 22 MLE-Lite tasks. Accounting covers complete runs with a 12-hour search execution budget. Cost segments span fully cached to uncached inputs; dots mark uncached estimates.

C.4 Action-Outcome Metrics

Figure 4 covers direct Improve and Crossover attempts from OpenMLE-Evo and EDL with Kimi K2.6 and GLM 5.2, using 22 tasks and three trajectories per model–harness condition. Unlike Figure 3, this analysis extends beyond 12 hours of wall-clock time. Draft, Debug, and code-free root nodes are excluded. Failures remain in every denominator. A successful Debug repair does not retroactively make the initial child valid.

For attempt k, let z_k indicate successful execution with a finite internal search reward y_k, oriented so that larger is better. Let p_k be the best reward among its recorded source nodes, and f_k the best reward available before execution. We distinguish

$$r_k^{\text{valid}} = z_k, \quad r_k^{\text{parent}} = \mathbb{I}[z_k = 1,\ y_k > p_k + \epsilon], \quad r_k^{\text{best}} = \mathbb{I}[z_k = 1,\ y_k > f_k + \epsilon]. \tag{18}$$

Here $\epsilon = 10^{-12}$. Comparing with the best solution so far requires a valid solution before the action. The first valid candidate therefore does not count as an improvement. These are internal search outcomes, not hidden-test improvements.

Table 9: Direct expansion outcomes. Each entry is OpenMLE-Evo (meta signals) → EDL; failures are included in attempt denominators.

Model	Operation	Attempts	Valid (%)	Improves best (%)
Kimi K2.6	Improve	391 → 778	68.29 → 87.53	10.49 → 19.79
	Crossover	149 → 131	57.05 → 90.84	8.72 → 28.24
GLM 5.2	Improve	442 → 876	75.57 → 92.47	14.03 → 22.83
	Crossover	180 → 104	76.67 → 89.42	10.56 → 21.15

C.5 Case Study: Evidence-Guided Jigsaw Ensemble

In Kimi K2.6 Jigsaw toxic-comment classification, repetition 0, the controller inspects a successful TF-IDF transfer (step 11), a regressed loss modification (step 12), and failed or regressed pseudo-label transfers (steps 13–15). It then reads the transformer–lexical implementation from step 11 and proposes checkpoint ensembling while preserving its TF-IDF branch and per-label fusion.

The resulting Improve child, step 17, uses step 11 as its recorded parent. Its code saves epoch checkpoints and combines their predictions. The logs record inference with epoch 1, 2, and 3 checkpoints, followed by blending with TF-IDF. Step 17 is selected for final submission and achieves Grade 0.0324 (Human Rank 0.9676). This case traces an observed outcome through the inspected evidence, proposed plan, and executed components. It does not establish whether the controller would have chosen the same action without those records.

C.6 OEC Ablations and Supporting Diagnostics

All variants retain five initial Draft attempts and up to five execution slots per round under the evaluation protocol in Appendix C.1.

No direction groups. Each candidate receives a separate entry. The initial view shows up to 40 candidate cards, with pagination for additional entries; code and history query limits are unchanged.

Fixed routing. A numerical router supplies five ordered operation/source assignments. The controller can inspect evidence, select a prefix and repetitions, and plan modifications. Crossover permits only component transfer. New-direction Draft is allowed only when no valid source is available. These restrictions also constrain the action mixture.

Fixed evidence. Each round receives up to six quality-ranked exemplars and five recent history records per direction, capped at 100,000 code characters and 60,000 history characters. Adaptive queries are disabled. Because sources must have been read completely, the bundle also limits source eligibility.

Direction organization. Figure 5a averages 66 task-runs per method, carrying completed runs forward to 12 wall-clock hours. At that point, OEC and No groups have 6.12 and 6.05 directions, versus 21.45 entries in the flat interface. Grouping thus reduces the overview size with similar direction coverage. Input tokens include all framework model usage.

Adaptive evidence retrieval. We compare retrieval on 292 recorded OEC states from 65 task-runs, using only candidates available before each round. OEC inspects 1.96 complete code exemplars and 3.00 detailed histories per round, versus 3.44 and 5.14 for fixed retrieval. In 59/292 rounds (20.2%), it accesses complete code or detailed history outside the fixed bundle. This comparison measures evidence access at matched states; it does not replay search performance.

In Leaf Classification, fixed retrieval supplies D01–D03; OEC reads D01 and ninth-ranked D09. The resulting transfer adds EfficientNet-B0 and horizontal-flip feature averaging to a ResNet pipeline while retaining fold-specific PCA and scaling. Validation log loss decreases from 0.0102228 to 0.00456734. This case illustrates useful evidence outside the fixed bundle. Full OEC uses 2.477M tokens per task-run versus 1.855M for Fixed evidence (33.5% more).