arXiv:2610.08691v1 [cs.AI] 6 Oct 2026

ScienceClaw: Benchmarking Continual Self-Evolution of AI-for-Science Agents Across the Natural and Social Sciences

Mingda Zhang
The Chinese University of Hong Kong, Shenzhen
Shenzhen, China
mingdazhang@ieee.org

Wenjin Liu
Nanyang Technological University
Singapore, Singapore
wenjinliu23@outlook.com

Tiesunlong Shen*
Centre for Biomedical Data Science, Duke-NUS Medical School
Singapore, Singapore
tiesunlong@nus.edu.sg

Zikai Xiao
Zhejiang University
Hangzhou, China
zikai@zju.edu.cn

Zhenghong Lin
Nanyang Technological University
Singapore, Singapore
hongzhenglin970323@gmail.com

Qing Xu
Jurisprudence Research Association, China Law Society
Beijing, China
20180090@ynu.edu.cn

Erik Cambria
Nanyang Technological University
Singapore, Singapore
cambria@ntu.edu.sg

Xiaoying Tang
The Chinese University of Hong Kong, Shenzhen
Shenzhen, China
tangxiaoying@cuhk.edu.cn

Haoran Luo*
Nanyang Technological University
Singapore, Singapore
haoran.luo@ntu.edu.sg

Abstract

Large language model agents are accelerating scientific automation, yet verified executions rarely become persistent program-level improvements, and existing evaluations do not examine this process across sequential tasks in both the natural and social sciences. We formalize ScienceClaw as fixed-parameter program self-evolution that unifies task solving, scientific verification, and program updates. ScienceClaw-Eval spans 23 disciplines and measures scientific correctness, evolutionary gain, retention, cross-dataset transfer, and evolution cost through sequential streams and independent reset evaluation. Our framework repairs executable workflows through multi-turn interaction, converts re-execution-verified failure–success trajectories into linked Skill and Operator candidates, and retains an update only when source-task replay reproduces the repair and independent scientific tasks improve. Code is available at https://github.com/beita6969/ScienceClaw.

CCS Concepts

• **Computing methodologies** → **Intelligent agents**; • **Applied computing** → *Physical sciences and engineering*; *Law, social and behavioral sciences.*

Keywords

AI for Science, agent self-evolution, multi-turn interactive scientific workflows, Skill evolution, Operator evolution

1 Introduction

Large language model agents are increasingly automating scientific problem solving [4, 14]. As illustrated in Figure 1, they connect natural-language reasoning with data, domain tools, and executable computation to construct end-to-end workflows [26, 31].

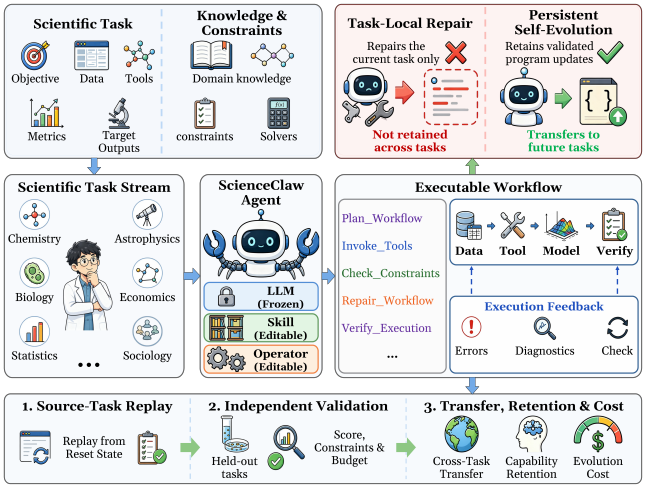


Figure 1: Task formulation of verifiable program-level self-evolution for AI-for-Science agents.

AI4Science tasks stress this capability through heterogeneous dependencies, long-horizon interactions, and stringent constraints, yet also expose runtime states, numerical diagnostics, and constraint checks that verify intermediate decisions. Scientific agents therefore offer a flexible cross-domain interface beyond fixed scripts and rules. Despite this promise, scientific agents remain limited in continual self-evolution, since repairs rarely persist.

Training-based methods improve workflow orchestration and error repair through supervised fine-tuning, reinforcement learning, or separate task optimization, but acquire most improvements before deployment [1, 20, 32, 34]. Training-free methods instead

*Corresponding authors.

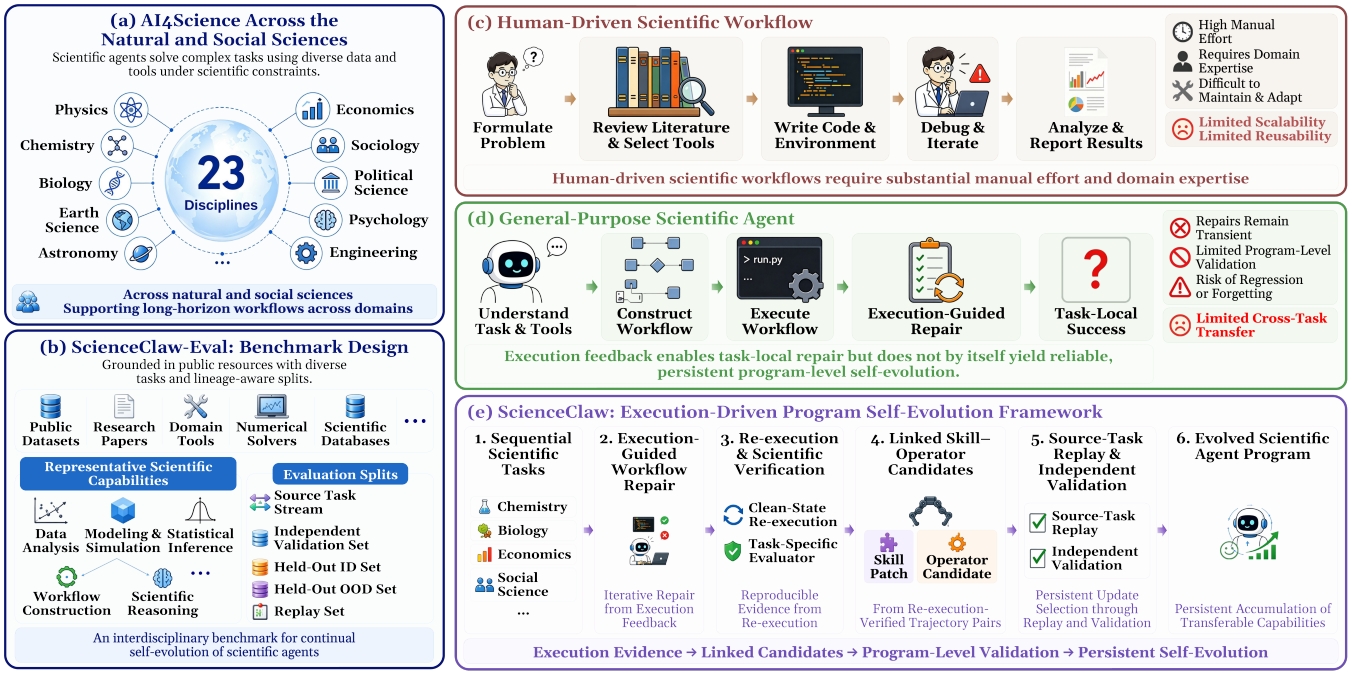


Figure 2: Overview of ScienceClaw. ScienceClaw-Eval spans 23 disciplines across the natural and social sciences, while ScienceClaw turns verified execution evidence into persistent Skill–Operator program updates.

update tools, Skills, control logic, or agent harnesses from execution trajectories [10, 11, 15, 18, 19, 21, 24, 30]. Scientific efforts, however, remain concentrated on individual tools or Skills [15, 22], while program-level evolution is evaluated mainly in general reasoning, coding, or system-operation environments. No unified setting therefore couples in-task verification of executability and scientific validity, linked Skill–Operator updates, and independent program-level evaluation. Scientific-agent and sequential self-evolution benchmarks do not cover constrained task streams spanning the natural and social sciences [2, 8, 12, 20, 29, 35]. Thus, task- or component-level gains do not establish reliable and transferable scientific self-evolution across disciplines and rounds.

This raises a central question: **How can verified scientific executions drive persistent and transferable program-level self-evolution without updating foundation-model parameters?** We answer it with ScienceClaw, a verifiable AI4Science framework connecting multi-turn workflow orchestration and program-level self-evolution. ScienceClaw represents the editable program as Skills and typed Operators, repairs workflows through execution feedback, and derives linked candidates from re-execution-verified failure–success trajectories. A candidate persists only if source-task replay reproduces the repair and independent scientific tasks improve. Figure 2 summarizes this process. In summary, our three main contributions are as follows:

- **Novel ScienceClaw Task.** We formalize program-level self-evolution over sequential AI4Science problems, turning verified executions into persistent and transferable updates without modifying foundation-model parameters.
- **ScienceClaw-Eval.** We construct a 23-discipline benchmark across the natural and social sciences with sequential task streams and independent reset evaluations.
- **ScienceClaw Framework.** We propose an execution-driven framework coupling multi-turn workflow repair with independently evaluated Skill–Operator updates, allowing scientific capabilities to accumulate across tasks and disciplines without model training.

2 Related Work

Scientific Agents and Evaluation. General-purpose LLM agents continue to exhibit limited reliability and generalization in specialized scientific tasks, motivating domain-specific tools, interactive execution environments, and scientific-agent benchmarks [2, 5, 20, 23]. Evaluation has progressed from static question answering to executable workflows that connect scientific data, domain software, and code execution [16, 20]. Interactive environments assess multi-step tool use and error recovery [25], while more recent benchmarks extend evaluation to the end-to-end research lifecycle, including idea generation, experiment analysis, and iterative refinement using real papers and data [3, 12, 16, 20, 29]. Discipline-specific benchmarks further examine the computational reproducibility of social-science findings and the physical fidelity of scientific programs [2, 33]. ScienceClaw-Eval instead assesses whether agents accumulate and retain capabilities across tasks.

Workflow Orchestration and Agent Self-Evolution. Early LLM agents mainly relied on hand-crafted prompts, static toolsets,

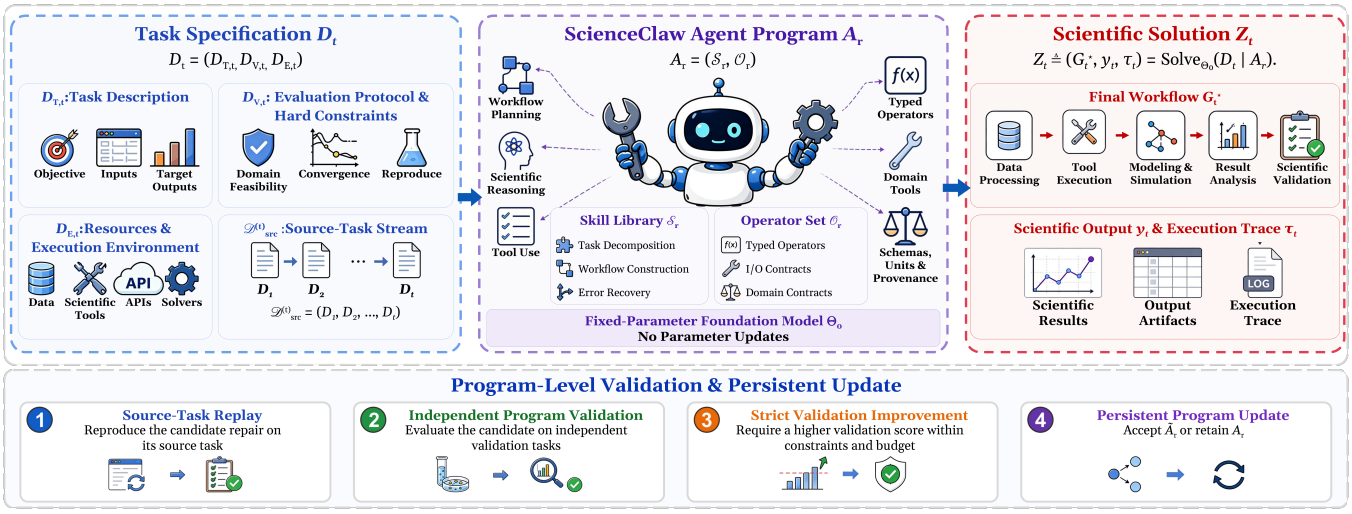


Figure 3: Definition and overview of the ScienceClaw task. Given task specification D_t and agent program A_r, ScienceClaw produces scientific solution Z_t and retains a candidate update only after source-task replay and independent program validation.

or predefined workflows, with tool interfaces and control flow typically fixed before deployment [7, 28, 31]. Automated workflow optimization subsequently introduced execution-guided search and progressive graph editing, while later methods abstract reusable Operators or distill Skills from execution trajectories [6, 19, 30, 32, 34]. More recent work evolves tools, Skills, control logic, and agent harnesses without updating foundation-model parameters [10, 11, 15, 18, 21, 22, 24]. In parallel, sequential evaluation frameworks measure evolutionary gain, cross-task transfer, retention, and forgetting [8, 13, 35]. Although scientific agents begin to evolve tools or Skills, program-level self-evolution has not been evaluated on executable task streams spanning the natural and social sciences, the gap ScienceClaw-Eval targets.

3 Task: ScienceClaw

ScienceClaw formalizes continual program-level self-evolution over sequential AI4Science problems while keeping foundation-model parameters fixed. As shown in Figure 3, verified evidence from long-horizon tool interaction is converted from transient task context into persistent program updates.

Let $D_t = (D_{T,t}, D_{V,t}, D_{E,t})$ denote task t. Here, $D_{T,t}$ specifies its objective, inputs, initial conditions, and target outputs; $D_{V,t}$ defines the evaluation protocol and hard constraints, including unit consistency, domain feasibility, convergence, and reproducibility; and $D_{E,t}$ provides data, scientific tools, solvers, simulators, and the execution environment. At round r, the editable program is $A_r = (\mathcal{S}_r, \mathcal{O}_r)$, where $\mathcal{S}_r$ contains Skills for decomposition, workflow construction, and recovery, and $\mathcal{O}_r$ contains typed executable Operators with explicit input, output, and domain contracts. Conditioned on D_t and A_r, a fixed-parameter foundation model constructs and executes

$$Z_t \triangleq (G_t^\star, y_t, \tau_t) = \text{Solve}_{\Theta_0}(D_t \mid A_r)\,. \tag{1}$$

The scientific solution Z_t comprises the final workflow $G_t^\star$, replayed from a reset task state, its output y_t, and execution trace τ_t. A task-specific evaluator checks both y_t and τ_t against $D_{V,t}$, making validity depend on the result and its computation.

Let $\mathcal{E}_r$ be the verified evidence accumulated through round r and $\mathcal{P}(\mathcal{E}_r)$ its induced candidate programs. Feasibility and selection are defined by

$$\begin{aligned} \mathcal{F}_r &= \left\{ A \in \mathcal{P}(\mathcal{E}_r) \;\middle|\; \begin{array}{l} R_{\text{src}}(A) = 1, \quad \mathbf{H}_{\text{val}}(A) = \mathbf{1}, \\ \qquad\qquad \mathbf{C}_{\text{val}}(A) \preceq \mathbf{B} \end{array} \right\}, \\ \widetilde{A}_r &\in \arg\max_{A \in \mathcal{F}_r} Q_{\text{val}}(A), \qquad \mathcal{F}_r \neq \emptyset. \end{aligned} \tag{2}$$

The persistent program update is then

$$\begin{aligned} A_{r+1} &= \begin{cases} \widetilde{A}_r, & \mathcal{F}_r \neq \emptyset \;\wedge\; Q_{\text{val}}(\widetilde{A}_r) > Q_{\text{val}}(A_r), \\ A_r, & \text{otherwise}, \end{cases} \\ \Theta_{r+1} &= \Theta_r = \Theta_0. \end{aligned} \tag{3}$$

Here, $\mathcal{F}_r$ is the feasible set; $R_{\text{src}}(A) = 1$ requires source-task reproduction of every new repair; and $Q_{\text{val}}(A)$ is the aggregate scientific score on independent validation tasks. $\mathbf{H}_{\text{val}}(A)$ collects non-negotiable execution and scientific integrity checks, with $\mathbf{1}$ the all-ones vector. Unsolved tasks lower Q_{val}, while hard-constraint violations make $\mathbf{H}_{\text{val}}(A) \neq \mathbf{1}$. $\mathbf{C}_{\text{val}}(A)$ and $\mathbf{B}$ are the resource-cost vector and component-wise budget; $\preceq$ denotes component-wise comparison. Equation (3) keeps Θ_0 fixed and retains the incumbent unless a feasible candidate strictly improves validation performance.

4 Benchmark: ScienceClaw-Eval

ScienceClaw-Eval benchmarks continual scientific-agent self-evolution with 23 research tasks across 23 natural- and social-science disciplines. It combines sequential task processing with independent reset evaluation. Systems share the orchestration model, initial program A_0, source order, tools, and budget. We describe its construction, protocol, and metrics (Appendix C).

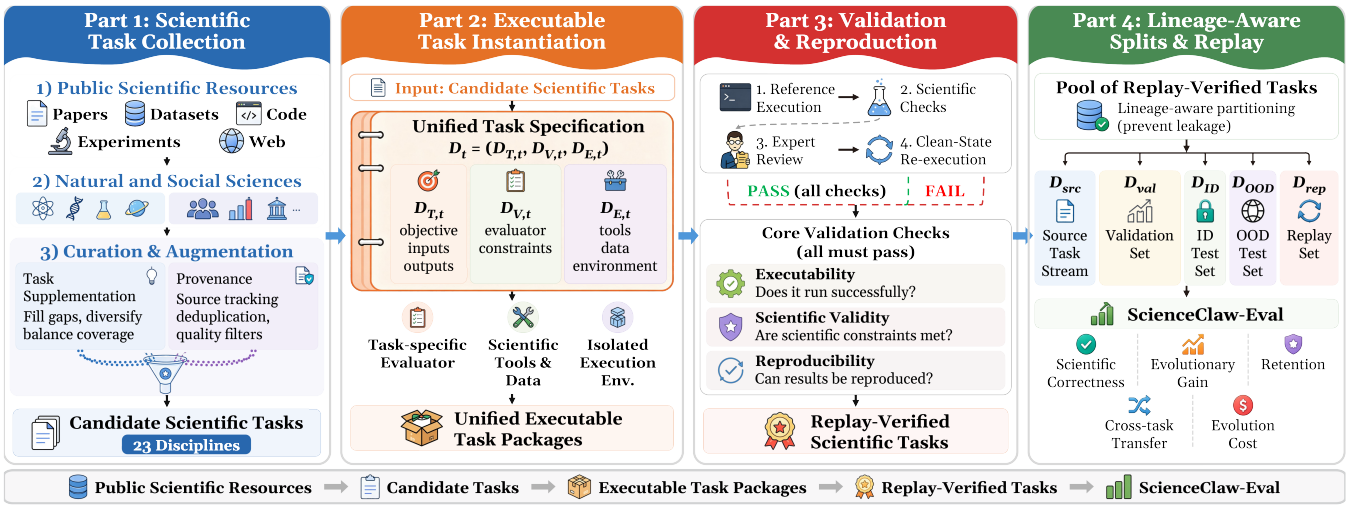


Figure 4: Construction of ScienceClaw-Eval, from scientific-task collection and executable instantiation to scientific validation, reproducibility checking, and lineage-aware evaluation splits.

4.1 ScienceClaw-Eval Construction

Figure 4 shows four stages: task collection, executable instantiation, validation, and evaluation partitioning. We collect tasks with explicit objectives, accessible inputs, and verifiable outcomes from public scientific resources, and supplement them to improve disciplinary coverage, workflow diversity, and horizon length while preserving provenance. Each task is instantiated as D_t with an isolated environment and a task-specific evaluator. Domain reviewers inspect evaluator logic, constraints, and reference executions; reset replay then verifies executability, scientific validity, and reproducibility. Provenance and construction lineage [27] determine the source stream $\mathcal{D}_{\text{src}}$, independent validation set $\mathcal{D}_{\text{val}}$, held-out ID set $\mathcal{D}_{\text{ID}}$, and same-discipline cross-dataset OOD set $\mathcal{D}_{\text{OOD}}$ [9]. $\mathcal{D}_{\text{rep}}$ contains frozen source-task instances encountered before each evaluation point, measuring retention without supplying new evolution evidence. Each task record fixes its source, partition rule, evaluator, and statistics before experiments begin.

4.2 ScienceClaw-Eval: Evaluation Design

Only source trajectories and evaluator feedback supply evolution evidence; $\mathcal{D}_{\text{val}}$ serves candidate selection. At each evaluation point, A_r is frozen, transient state is cleared, and the snapshot is evaluated on $\mathcal{D}_{\text{ID}}$, $\mathcal{D}_{\text{OOD}}$, and $\mathcal{D}_{\text{rep}}$.

For instance i, the evaluator jointly examines output and trace. $z_i(A_r) = 1$ only when execution completes within budget, task-native metrics meet their acceptance rule (Appendix C.3), and every scientific hard constraint holds; otherwise $z_i(A_r) = 0$. We macro-average discipline-level success:

$$\text{MacroSR}(A_r) = \frac{1}{23} \sum_{d=1}^{23} \frac{1}{|\mathcal{I}_d|} \sum_{i \in \mathcal{I}_d} z_i(A_r), \tag{4}$$

where $\mathcal{I}_d$ contains instances in discipline d. Q_{val} in Eq. (2) is MacroSR on $\mathcal{D}_{\text{val}}$. Changes from A_0 on $\mathcal{D}_{\text{ID}}$ and $\mathcal{D}_{\text{OOD}}$ measure in-benchmark gain and same-discipline cross-dataset transfer; $\mathcal{D}_{\text{rep}}$ measures retention against the snapshot immediately after each source task. We also report natural- and social-science success, task-native metrics, multi-turn repair, long-horizon completion, and execution-cost diagnostics. Evaluators, partitions, and executable configurations remain frozen across methods.

5 Methodology: ScienceClaw

Scientific workflow failures often surface only when heterogeneous data, domain software, and numerical procedures execute. ScienceClaw turns reproducible repairs into linked Skill–Operator updates through execution-guided orchestration (Section 5.1) and program self-evolution (Section 5.2), with foundation-model parameters fixed. A repair persists only after it reproduces on its source task and improves the complete program on independent tasks, rather than on a single execution trace (Appendix B).

5.1 Execution-Guided Scientific Workflow Orchestration

Typed scientific workflow representation. ScienceClaw represents the workflow of task t after interaction step k as a directed graph with port-level data dependencies:

$$\begin{aligned} G_{t,k} &= (N_{t,k}, E_{t,k}, \psi_{t,k}), \\ \psi_{t,k}(v) &= (o_v, \eta_v), \quad v \in N_{t,k}, \\ ((u,p),(v,q)) \in E_{t,k} &\Longrightarrow \text{Compat}_{D_t}\left(\Gamma^{\text{out}}_{u,p}, \Gamma^{\text{in}}_{v,q}\right) = 1. \end{aligned} \tag{5}$$

$N_{t,k}$ contains Operators and task tools; $\psi_{t,k}$ maps node v to executable o_v and configuration η_v; and an edge links output port p of u to input port q of v. Schema $\Gamma = (\text{type}, \text{shape}, \text{unit}, \text{provenance})$ records port metadata. Compat_{D_t} requires matching type and shape and explicit, provenance-recorded unit conversion; the same schemas define Operator boundaries below. Units are required when applicable, and provenance records upstream data and the transformations applied along every edge.

Execution-guided orchestration. ScienceClaw constructs $G_{t,k}$ through atomic edits after tool execution [26, 32]. Let $\mathcal{R}^{\mathcal{S}}_{t,r} = \text{Retrieve}(D_t; \mathcal{S}_r)$ and $\mathcal{R}^{\mathcal{O}}_{t,r} = \text{Retrieve}(D_t; \mathcal{O}_r)$ be retrieved Skills and Operators, $\mathcal{T}_t$ the tools in $D_{E,t}$, and $\mathcal{R}_{t,r} = (\mathcal{R}^{\mathcal{S}}_{t,r}, \mathcal{R}^{\mathcal{O}}_{t,r} \cup \mathcal{T}_t)$ their execution context. Given history $H_{t,k}$ and fixed Θ_0, the policy samples

$$(a_{t,k}, v_{t,k}) \sim \pi_{\Theta_0}\big(\cdot \mid D_t, G_{t,k}, H_{t,k}, \mathcal{R}_{t,r}\big). \tag{6}$$

Here, π_{Θ_0} is fixed; $a_{t,k}$ adds, removes, or modifies a node, dependency, or configuration, and $v_{t,k}$ records contributing Skill and Operator identifiers. Checkpoint $\chi_{t,k}$ stores completed outputs and runtime state; execution proceeds as

$$\big(G_{t,k+1}, \chi_{t,k+1}, f_{t,k}\big) = \text{Execute}_{D_{E,t}}\big(G_{t,k}, \chi_{t,k}, a_{t,k}\big). \tag{7}$$

$H_{t,k+1}$ appends the edit and execution record. The executor checks schemas and reruns only affected descendants; $f_{t,k}$ records runtime and numerical diagnostics, $v_{t,k}$ attributes reproduced repairs, and checkpointing limits long-horizon recomputation. Scientific validity is assessed by reset replay below.

Reproducible execution evidence. Because checkpoints may retain transient state, candidate evidence is regenerated in reset environment $D_{E,t}$ under the replication protocol and evaluator in $D_{V,t}$:

$$\begin{aligned}(y_{t,k}, \tau_{t,k}) &= \text{Replay}_{D_{E,t}}(G_{t,k}),\\ (q_{t,k}, \mathbf{h}_{t,k}, \mathbf{c}_{t,k}) &= \text{Eval}_{D_{V,t}}(y_{t,k}, \tau_{t,k}).\end{aligned} \tag{8}$$

With D_t and $G_{t,k}$, these outputs define $e_{t,k}$: $y_{t,k}$ is the scientific output; $\tau_{t,k}$, the executed path and diagnostics; $q_{t,k}$, the task score; $\mathbf{h}_{t,k}$, hard-constraint indicators; and $\mathbf{c}_{t,k}$, resource use. $\text{Pass}_t(e_{t,k}) = 1$ iff applicable convergence, physical or statistical validity, and reproducibility requirements hold within budget. Replay alone does not authorize persistence, which is decided by the strict-improvement rule on independent validation in Section 5.2.

5.2 Execution-Driven Program Self-Evolution

Evolution trigger and repair attribution. Evolution instance i on $D_i = D_t$ ends at replay-verified success $e_i^+ = e_{t,k_i^+}$ with $\text{Pass}_i(e_i^+) = 1$. Its preceding endpoint $e_i^- = e_{t,k_i^-}$ is the latest replay-verified failure, or $\emptyset$ if none; the endpoints define the shortest repair:

$$\delta_i = \begin{cases}\langle a_{t,k}\rangle_{k_i^- \le k < k_i^+}, & e_i^- \neq \emptyset,\\ \emptyset, & e_i^- = \emptyset.\end{cases} \tag{9}$$

Let $G_i^+ = G_{t,k_i^+}$ and $\tau_i^+ = \tau_{t,k_i^+}$. Edit types and graph topology separate strategy changes from executable capability:

$$\begin{aligned}\Delta_i^{\mathcal{S}} &= \Pi_{\text{ctrl}}(\delta_i),\\ \Delta_i^{\mathcal{O}} &= \text{CC}_\Gamma\big(G_i^+, \Pi_{\text{exec}}(G_i^+, \delta_i, \tau_i^+; \mathcal{O}_r)\big).\end{aligned} \tag{10}$$

Π_{ctrl} selects topology, routing, and configuration edits that preserve executable payloads; Π_{exec} selects generated, repaired, or unregistered executable nodes; and CC_Γ groups them into maximal connected subgraphs with compatible boundaries. Thus a reproduced repair can yield linked Skill and Operator candidates; without e_i^-, only the latter is formed. No auxiliary LLM judge is required to rank or select candidates.

Linked Skill–Operator abstraction. Let $v_i^{\mathcal{S}}$ collect the Skill identifiers in δ_i. For $\Delta_i^{\mathcal{S}} \neq \emptyset$, its control edits define

$$\Delta\mathcal{S}_i = \text{Patch}_{\Theta_0}\big(\mathcal{S}_r[v_i^{\mathcal{S}}]; \Delta_i^{\mathcal{S}}, e_i^-, e_i^+\big). \tag{11}$$

where $\mathcal{S}_r[v_i^{\mathcal{S}}]$ selects attributed entries: an empty identifier set creates a Skill, otherwise only selected instructions change; $\Delta_i^{\mathcal{S}} = \emptyset$ yields no Skill candidate. Each executable component $U \in \Delta_i^{\mathcal{O}}$ has

$$\hat{o}_{i,U} = \big(G_i^+[U], \Gamma_{\partial U}, \kappa_{i,U}, \rho_{i,U}\big). \tag{12}$$

Here, $G_i^+[U]$ is the executable subgraph; $\partial U = (\partial^- U, \partial^+ U)$ its external input and output ports; $\Gamma_{\partial U}$ their schemas; $\kappa_{i,U}$ the preconditions, postconditions, and applicability conditions derived from D_i and the boundary slice of τ_i^+; and $\rho_{i,U}$ the component provenance. Boundary replay restores recorded inputs at $\partial^- U$ and checks isolated outputs at $\partial^+ U$ against τ_i^+ under $D_{V,i}$, including numerical tolerances and stochastic replication. $\hat{o}_{i,U}$ enters $\Delta\mathcal{O}_i$ only if $\text{BReplay}_{D_{V,i}}(\hat{o}_{i,U}; \tau_i^+) = 1$. Bundle $B_i = (\Delta\mathcal{S}_i, \Delta\mathcal{O}_i)$ couples the strategy change with its executable capability.

Program-level validation and update. Atomic application $\widetilde{A}_i = \text{Apply}(A_r; B_i)$ assigns version identifiers ω_i and requires

$$\begin{aligned}(H_i^{\text{src}}, e_i^{\text{src}}, \tau_i^{\text{src}}) &= \text{Run}_{D_i}(\widetilde{A}_i),\\ R_{\text{src}}(\widetilde{A}_i) &= \text{Pass}_i(e_i^{\text{src}}) \wedge \text{Use}_i(\omega_i; H_i^{\text{src}}, \tau_i^{\text{src}}).\end{aligned} \tag{13}$$

Here, Run_{D_i} returns reset-run history H_i^{src}, evidence e_i^{src}, and trace τ_i^{src} as in Eq. (8), while Use_i checks every ω_i. Only $R_{\text{src}} = 1$ candidates enter $\mathcal{P}(\mathcal{E}_r)$ and $\mathcal{D}_{\text{val}}$; Equation (3) then retains strict budget-feasible improvements with $\mathcal{E}_r$ and foundation-model parameters fixed at Θ_0 throughout every round (Appendix A).

6 Experimental Evaluation

Setup. All systems process the common 23-discipline source streams. Evolving methods update their programs, whereas the frozen control remains at A_0. We freeze the snapshot after each of seven rounds, clear transient state, and evaluate it on the held-out same-discipline target datasets in $\mathcal{D}_{\text{OOD}}$; OOD results never generate or select updates. Qwen3.5-27B orchestrates every task for all systems, GPT-5.4 performs the evolution of ScienceClaw and every baseline, and each discipline contributes 64 IID and 64 OOD instances. We compare ScienceClaw with an evolution-disabled frozen agent and methods that evolve Skills, scientific tools, agent harnesses, or persistent scientific-agent programs and memory: SkillOpt [30], test-time tool evolution (TTE) [15], Agentic Harness Engineering (AHE) [10], EvoMaster [36], and EvoScientist [17]. Each baseline runs its best released configuration through fixed ScienceClaw-Eval adapters under the same models, tools, source stream, validation data, and update budget (Appendix B.8); adapters and initial assets are fixed without access to ID/OOD instances or evaluators. Two ScienceClaw variants complete the comparison: RuleEvo evolves by deterministic trace projection alone, and Qwen-only evolution replaces GPT-5.4 with Qwen. Three PhD reviewers per discipline and a natural- or social-science expert audited each evaluator and its reference executions. Table 1 reports task-native scores, compared only within each discipline because metrics differ in units and direction; round-wise analyses use MacroSR on $\mathcal{D}_{\text{OOD}}$ (Eq. (4)) as the OOD macro score.

Table 1: IID and cross-dataset OOD results across 23 disciplines. Discipline and metric names are abbreviated. The best score in each discipline and split is in bold and the second best is underlined.

	Frozen (A_0)		SkillOpt		TTE		AHE		EvoMaster		EvoScientist		ScienceClaw	
Discipline (metric)	IID	OOD	IID	OOD	IID	OOD	IID	OOD	IID	OOD	IID	OOD	IID	OOD
Higher is better (↑)														
Agricultural sci. (PQ+)	69.4761	64.6829	69.9837	66.7649	73.5549	72.0626	73.1602	72.0490	75.1756	73.1862	74.7019	73.2184	**76.9612**	**75.7629**
Biological sci. (Spearman)	0.5057	0.3745	0.5358	0.3659	0.5599	0.3975	0.5642	0.3968	0.5674	0.4103	0.5683	0.4089	**0.5816**	**0.4234**
Biomedical sci. (DSC)	0.7816	0.7088	0.7794	0.7521	0.8169	0.7784	0.8214	0.7811	0.8348	0.7877	0.8387	0.7798	**0.8556**	**0.8226**
Chemical sci. (ROC-AUC)	0.6518	0.6205	0.6830	0.6518	0.7232	0.7054	0.7299	0.7143	0.7344	0.7165	0.7299	0.7188	**0.7545**	**0.7478**
Creative arts (SDR (dB))	8.9208	8.0993	9.4971	8.3085	9.8649	9.0812	9.7900	9.0046	9.9453	9.1835	9.9112	9.2653	**10.1867**	**9.5619**
Education (10-mask acc.)	0.5965	0.6202	0.6603	0.6626	0.6561	0.6467	0.6616	0.6567	0.6720	0.6731	0.6579	0.6594	**0.6878**	**0.7087**
Engineering (DCASE score)	0.5709	0.4663	0.6138	0.4659	0.6243	0.4943	0.6317	0.4902	0.6380	0.4992	0.6310	0.5017	**0.6535**	**0.5269**
Health sci. (Clin. utility)	0.4971	0.6834	0.5170	0.7132	0.5278	0.7380	0.5234	0.7182	0.5383	0.7417	0.5367	0.7428	**0.5514**	**0.7796**
Indigenous (chrF++)	15.4451	13.7205	16.5641	14.8441	16.3493	14.1870	16.7301	14.8969	16.9074	15.0442	16.6453	14.7130	**17.4351**	**15.9778**
Computing sci. (pass@1)	0.7656	0.3750	0.7969	0.4063	0.8438	0.4219	0.8438	0.4219	0.8594	0.4375	0.8438	0.4375	**0.8750**	**0.4531**
Language & culture (LAS)	0.7288	0.5599	0.7623	0.5713	0.7786	0.6042	0.7761	0.6004	0.7845	0.6267	0.7759	0.5879	**0.8138**	**0.6497**
Law (mAP)	0.7513	0.6918	0.8158	0.7652	0.8074	0.7706	0.8211	0.7720	0.8292	0.7863	0.8170	0.7642	**0.8554**	**0.8289**
Mathematics (Oracle acc.)	0.8438	0.7344	0.9063	0.7969	0.9063	0.7656	0.9063	0.8125	0.9219	0.8438	0.9063	0.8125	**0.9531**	**0.8750**
Philosophy (F1)	0.4178	0.4166	0.4599	0.4428	0.4699	0.4846	0.4636	0.4868	0.4717	0.4943	0.4737	0.4925	**0.4866**	**0.5138**
Psychology (Micro acc.)	0.6088	0.5615	0.6364	0.6064	0.6269	0.6084	0.6355	0.6149	0.6467	0.6343	0.6476	0.6284	**0.6618**	**0.6586**
Lower is better (↓)														
Built env. (NRMSE %)	51.0997	53.5009	47.7143	49.6083	46.7210	49.5570	46.7362	49.0341	46.2726	46.7640	47.2762	49.4801	**45.0065**	**45.0970**
Commerce (MASE)	1.6009	1.8740	1.5268	1.6867	1.5671	1.7502	1.5227	1.7066	1.4984	1.6930	1.5281	1.6846	**1.4653**	**1.5986**
Earth sci. (RMSE (K))	1.0562	1.1364	1.0476	1.0379	1.0157	1.0005	1.0094	1.0114	1.0010	1.0042	1.0008	1.0079	**0.9721**	**0.9580**
Economics (sMAPE (%))	12.7394	18.1343	11.8721	16.8216	12.1577	16.9410	11.7390	16.2028	11.6350	16.1758	11.7621	16.7195	**11.3320**	**15.5025**
Environmental sci. (CRPS)	0.8104	0.6504	0.7786	0.6456	0.7517	0.6325	0.7565	0.6118	0.7304	0.5944	0.7326	0.5979	**0.7164**	**0.5751**
History (cMER-micro)	0.0185	0.0296	0.0173	0.0271	0.0177	0.0268	0.0172	0.0264	0.0172	0.0261	0.0172	0.0268	**0.0166**	**0.0251**
Human society (nRMSE)	0.0113	0.0213	0.0107	0.0201	0.0104	0.0192	0.0103	0.0191	0.0103	0.0192	0.0104	0.0191	**0.0100**	**0.0181**
Physical sci. (MAE)	36.6271	44.6757	35.1699	41.9077	33.4027	39.0907	33.5864	39.7923	33.0932	38.2995	32.8654	39.0137	**32.1581**	**37.1306**

RQ1: Does persistent program evolution improve cross-dataset performance? Yes. ScienceClaw is best in all 23 disciplines on both splits (Table 1), and its mean relative margin over the strongest baseline in each discipline grows from 2.58% on IID to 4.19% on OOD (12.25% and 16.45% over the frozen agent), and it is positive in every discipline, from 1.82% to 3.73% on IID and from 3.05% to 6.21% on OOD (Appendix G.1). The largest OOD margins appear in Indigenous studies (6.21%) and Law (5.42%), the smallest in Physical sciences (3.05%). The wider OOD margin is consistent with the gate: a candidate persists only after it reproduces on its source task and improves independent validation tasks, which filters out updates fitted to one source, whereas single-asset methods can fall below the frozen agent (SkillOpt in three of 46 cells). The baselines order by how much of the program they evolve: their mean OOD gain over the frozen agent rises from 5.74% for SkillOpt (Skills) through 8.82% for TTE (tools) and 9.73% for AHE (harness) to 10.55% for EvoScientist and 11.95% for EvoMaster (programs and memory), against 16.45% for ScienceClaw, above all of them. ScienceClaw also beats the frozen agent in every discipline, by 11.57% to 23.34% on OOD, and its margin is similar on higher-is-better (4.26%) and lower-is-better metrics (4.06%). Its OOD score falls 11.73% below its IID score on average, against 16.06% for the frozen agent and 13.46% to 15.45% for the baselines, so the evolved program degrades least of all systems when the dataset changes.

RQ2: Does evolution keep improving OOD performance over rounds? Yes. ScienceClaw's OOD macro score rises from 77.72 at A_0 to 91.30 after round 7 (+13.59 pp), against +9.78 pp for EvoMaster, the strongest baseline, and its lead over the best other system grows from 0.41 pp after round 1 to 3.80 pp (Figure 6a,d,e). Round 3 is the only round in which another system leads, by 0.41 pp. From round 3 to round 7 ScienceClaw adds 8.83 pp, nearly twice the 4.6 pp of EvoMaster. The curve keeps rising because ScienceClaw promotes more candidates than any other system, 70 of the 161 submitted (43.48% against 22.98% to 39.13%; Figure 6i), so the gate is selective without stalling. RuleEvo and Qwen-only, which change only how candidates are produced, reach +3.60 and +5.50 pp, so candidate quality, not the gate alone, drives the gain. None of the five evolving systems dips by more than 0.34 pp in any round.

RQ3: Does evolution transfer across disciplines and retain earlier repairs? Yes. Evolving on one discipline family lifts the OOD score of the others: 18 of the 20 cross-family pairs are positive, with a mean gain of 1.58 pp against 14.69 pp within a family (Figure 6b), so most of the evolved capability is discipline specific and the part that transfers rarely harms. Transfer is strongest between related families: Social/behavioural and Humanities/law (+4.27, +3.61), and Engineering/computing and Physical/Earth (+3.54, +3.47). The two negative pairs, both between Physical/Earth and Humanities/law, stay below 0.6 pp in magnitude. Retention shows the same pattern (Figure 6c; Appendix G.3): ScienceClaw reaches 4.20 pp forward (all 25 pairs) and 3.66 pp backward transfer with 0.13 pp average and 0.51 pp maximum forgetting and 11.89% negative transfer, against 0.33 pp, 1.05 pp, and 14.33% for EvoMaster. Source replay protects earlier repairs, and independent validation guards the other families against regressions.

RQ4: Which mechanisms produce the gain? The linked

Table 2: Mean of IID and cross-dataset OOD results of the ablation variants across 23 disciplines. Discipline and metric names are abbreviated. The best score in each discipline is in bold and the second best is underlined.

Discipline (metric)	Frozen	RuleEvo	Qwen-only	w/o Skill evol.	w/o Oper. evol.	w/o Work. evol.	w/o link. evol.	w/o src. replay	w/o sci. constr.	w/o ind. valid.	w/o cross-task mem.	w/o IID select.	w/o OOD transfer	ScienceClaw (full)
Higher is better (↑)														
Agricultural sci. (PQ+)	67.0795	71.5023	72.0207	74.1946	71.3347	72.0963	72.5891	70.7277	69.4636	71.4427	73.5959	68.5843	72.1931	**76.3621**
Biological sci. (Spearman)	0.4401	0.4617	0.4692	0.4897	0.4728	0.4871	0.4793	0.4841	0.4799	0.4819	0.4707	0.4482	0.4622	**0.5025**
Biomedical sci. (DSC)	0.7452	0.7724	0.7953	0.8154	0.7929	0.8005	0.8016	0.7841	0.7726	0.7770	0.8054	0.7488	0.8072	**0.8391**
Chemical sci. (ROC-AUC)	0.6362	0.6875	0.7087	0.7232	0.7076	0.7254	0.7176	0.6964	0.6908	0.7121	0.7232	0.6920	0.7098	**0.7511**
Creative arts (SDR (dB))	8.5101	9.2885	9.1862	9.5970	9.2316	9.4530	9.3943	9.4257	9.3184	9.5349	9.5109	9.0730	9.4250	**9.8743**
Education (10-mask acc.)	0.6083	0.6382	0.6579	0.6654	0.6761	0.6766	0.6722	0.6728	0.6627	0.6690	0.6652	0.6254	0.6331	**0.6982**
Engineering (DCASE)	0.5186	0.4976	0.5114	0.5703	0.5398	0.5568	0.5356	0.5368	0.5240	0.5371	0.5671	0.4736	0.5516	**0.5902**
Health sci. (Clin. utility)	0.5903	0.6240	0.6186	0.6460	0.6163	0.6246	0.6380	0.6299	0.6034	0.6281	0.6428	0.6041	0.6358	**0.6655**
Indigenous (chrF++)	14.5828	15.4084	15.9032	15.5616	16.0139	16.0826	15.9824	16.1154	15.8247	16.1141	15.7938	15.4115	15.6417	**16.7064**
Computing sci. (pass@1)	0.5703	0.5859	0.6016	0.6484	0.6172	0.6328	0.6250	0.6172	0.6094	0.6250	0.6406	0.5625	0.6406	**0.6641**
Language & culture (LAS)	0.6443	0.6441	0.6687	0.7097	0.6830	0.6937	0.6852	0.7009	0.6905	0.7028	0.6956	0.6229	0.6632	**0.7317**
Law (mAP)	0.7215	0.7791	0.7863	0.7881	0.8102	0.7967	0.8068	0.7959	0.7769	0.7813	0.7971	0.7743	0.7898	**0.8422**
Mathematics (Oracle acc.)	0.7891	0.8594	0.8516	0.8594	0.8750	0.8906	0.8750	0.8594	0.8438	0.8438	0.8828	0.8359	0.8828	**0.9141**
Philosophy (F1)	0.4172	0.4581	0.4715	0.4672	0.4786	0.4773	0.4781	0.4555	0.4553	0.4680	0.4666	0.4383	0.4454	**0.5002**
Psychology (Micro acc.)	0.5851	0.6177	0.6144	0.6194	0.6346	0.6402	0.6301	0.6138	0.6092	0.6242	0.6298	0.6122	0.6209	**0.6602**
Lower is better (↓)														
Built env. (NRMSE %)	52.3003	48.2400	48.0417	46.8999	47.4011	46.3133	46.8281	47.1144	47.7366	46.9883	46.8659	48.7799	47.1814	**45.0517**
Commerce (MASE)	1.7375	1.6348	1.6234	1.6220	1.5815	1.5819	1.6117	1.5874	1.6189	1.5968	1.6018	1.6564	1.6396	**1.5320**
Earth sci. (RMSE (K))	1.0963	1.0506	1.0268	0.9952	1.0220	0.9925	0.9982	1.0033	1.0035	0.9987	0.9978	1.0409	1.0221	**0.9650**
Economics (sMAPE (%))	15.4369	15.1645	14.9715	14.4654	14.0826	14.1269	14.9184	14.1704	14.2476	14.1622	14.5381	16.2718	15.4171	**13.4172**
Environmental sci. (CRPS)	0.7304	0.6994	0.6845	0.6659	0.6889	0.6723	0.6707	0.6756	0.6752	0.6656	0.6837	0.7144	0.6870	**0.6458**
History (cMER-micro)	0.0241	0.0243	0.0230	0.0227	0.0217	0.0220	0.0227	0.0230	0.0234	0.0228	0.0220	0.0242	0.0238	**0.0209**
Human society (nRMSE)	0.0163	0.0158	0.0162	0.0145	0.0154	0.0151	0.0156	0.0158	0.0154	0.0154	0.0149	0.0166	0.0153	**0.0141**
Physical sci. (MAE)	40.6514	39.0216	38.3645	35.8828	37.1914	36.4845	36.7402	37.4532	37.3627	36.6232	35.8646	40.0907	37.0317	**34.6443**

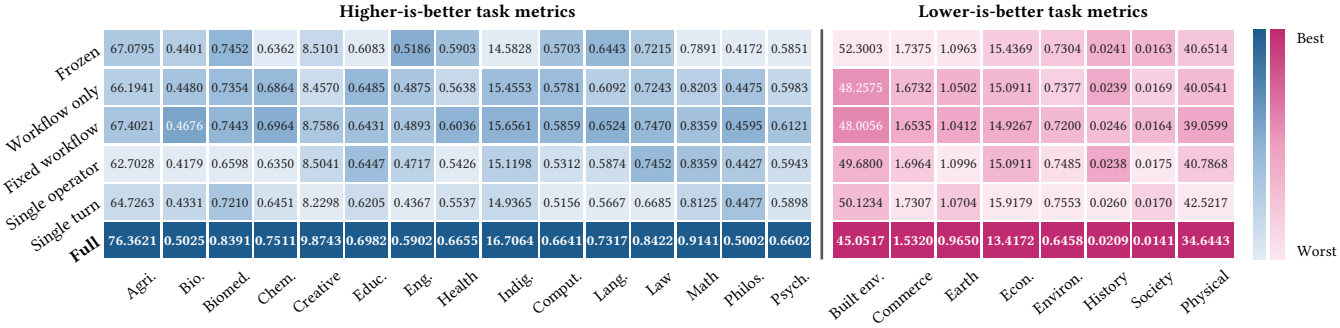


Figure 5: Mean of IID and OOD task-native scores for the single-component and execution-structure ablation variants across 23 disciplines (variants not listed in Table 2, with the frozen agent and the full system for reference). Colours are normalised within each discipline column (darker is better), and the two groups follow higher-is-better and lower-is-better metrics.

evolution and the replay gate. The full system is best in all 23 disciplines (Table 2; Appendices B.7 and G.2), and every ablation keeps only part of its gain over the frozen agent, with gain measured as the average relative drop below the full system (13.68% for Frozen; Appendix F.3). Updating Skills and Operators without committing them as one bundle keeps 60% of the gain, a loss comparable to removing Operator evolution (59%); on Commerce and Law the linked system gains 14.69% and 19.82% on OOD over Frozen, against 9.27% and 13.12% unlinked (Figure 6f). The gate matters even more: without IID selection only 15% of the gain remains, and without scientific constraints, source replay, or the independent validator, 46%, 55%, and 59%, with losses 2.4 to 2.8 times larger on OOD than on IID. RuleEvo and Qwen-only keep only about a third and 44%. In 20 of 23 disciplines the runner-up to the full system removes a single evolution component, so no component carries the gain alone. Execution structure is the base on which evolution acts: single-turn and single-operator execution fall below the frozen agent (Figure 5), and Linked Full runs 4.84 planner rounds and 4.24 distinct Operators per task, repairs 71% of failures from feedback, recovers 89% after interruption, and replays 96% cleanly, against 0, 31%, and 78% for a single-turn agent (Figure 6g; case studies of Commerce and Law in Appendix E).

RQ5: Is the gain reliable and efficient? Yes. ScienceClaw passes the hard constraints on 98.23% of instances, 2.58 points above EvoMaster, makes the fewest erroneous promotions in Figure 6c (1.23% against 3.80% to 7.25%), and costs less per OOD

(a) Continual self-evolution

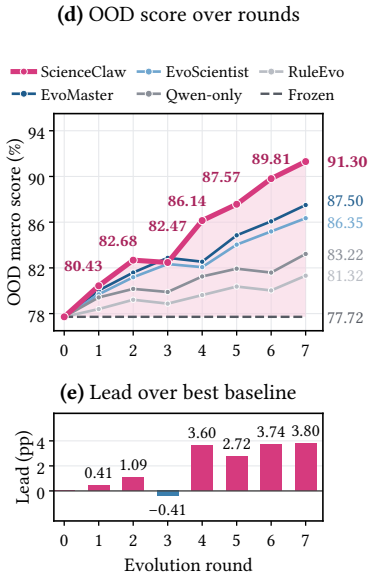
Method	OOD macro score (%) after round 0	1	2	3	4	5	6	7	Round-7 summary Gain (pp)	Prom.	Rej.	Rate (%)
Frozen	77.72	77.72	77.72	77.72	77.72	77.72	77.72	77.72	0.00	0	0	–
RuleEvo	77.72	78.40	79.21	78.87	79.62	80.37	80.03	81.32	3.60	37	124	22.98
Qwen-only	77.72	79.42	80.16	79.89	81.25	81.93	81.59	83.22	5.50	49	112	30.43
EvoScientist	77.72	79.69	81.18	82.34	82.07	84.04	85.19	86.35	8.63	56	105	34.78
EvoMaster	77.72	80.03	81.59	**82.88**	82.54	84.85	86.07	87.50	9.78	63	98	39.13
ScienceClaw	77.72	**80.43**	**82.68**	82.47	**86.14**	**87.57**	**89.81**	**91.30**	**13.59**	**70**	91	**43.48**

(b) Cross-family forward transfer (pp)

Source family	Life/ health	Phys./ Earth	Eng./ comp.	Social/ behav.	Hum./ law
Life/health	**13.08**	2.90	1.77	0.62	0.33
Phys./Earth	2.20	**15.32**	3.47	0.36	−0.33
Eng./comp.	1.92	3.54	**15.04**	1.84	1.64
Social/behav.	1.10	0.32	0.95	**14.37**	4.27
Hum./law	0.27	−0.51	1.26	3.61	**15.66**

(c) Transfer, forgetting and reliability

Method	FWT/BWT ↑	Neg. ↓	Forg. ↓ avg/max	Attain. ↑	Err. ↓
SkillOpt	1.42/0.26	31.39	0.88/2.45	92.67	6.52
TTE	1.75/0.41	26.97	0.76/2.17	93.63	7.25
EvoScientist	2.74/1.23	17.32	0.42/1.23	94.43	4.58
EvoMaster	3.19/1.53	14.33	0.33/1.05	94.99	3.80
ScienceClaw	**4.20/3.66**	**11.89**	**0.13/0.51**	**96.42**	**1.23**



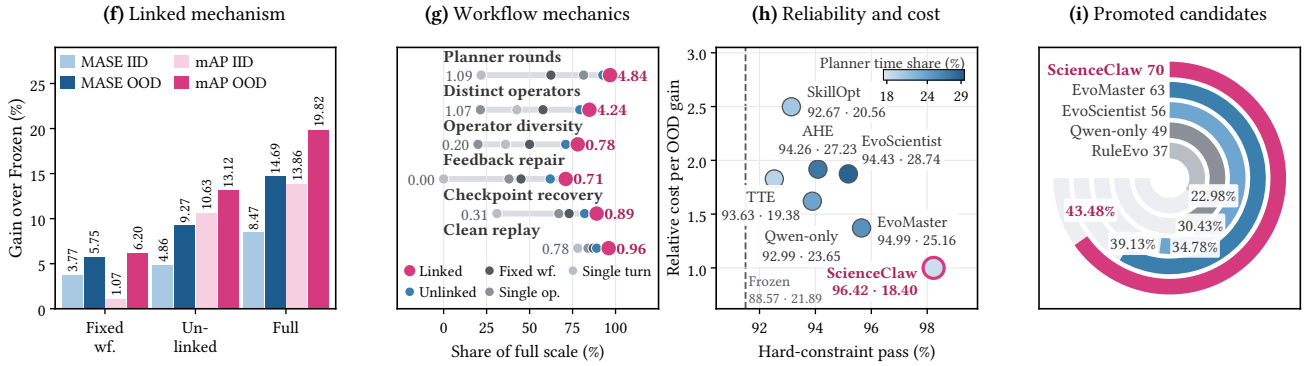


Figure 6: Continual evolution, transfer, and mechanisms. (a) OOD macro score (MacroSR on $\mathcal{D}_{\text{OOD}}$, %) of each snapshot after round r, with gain over A_0 in pp and promoted (Prom.) and rejected (Rej.) candidates out of 161. (b) Forward transfer (pp): OOD gain on the target family after evolving on the source family. (c) Forward and backward transfer (FWT/BWT, pp), negative-transfer share (Neg., %), average and maximum forgetting on $\mathcal{D}_{\text{rep}}$ (Forg., pp), continual-target attainment (Attain., %), and erroneous promotions (Err., %). (d) OOD macro score by round. (e) Lead of ScienceClaw over the best other system (pp). (f) Gain over Frozen (%) of the linked-mechanism variants on Commerce (MASE) and Law (mAP). (g) Workflow mechanics: planner rounds, distinct Operators, Operator diversity, feedback repair, checkpoint recovery, and clean replay. (h) Hard-constraint pass rate against cost per OOD gain (ScienceClaw = 1), coloured by planner wall-time share, with Attain. and planner share annotated. (i) Promoted candidates and promotion rate. Bold and underline mark the best and second best in (a) and (c); in (b), bold marks the within-family gain.

point than every other method (1.37 to 2.50 times as much, 27.1% less than EvoMaster), while its planner takes the smallest share of wall-clock time (18.40%; Figure 6h; Appendix G.3). Every compared evolving method passes more constraints than the frozen agent (91.51%), and ScienceClaw adds the most (+6.73 points). Its target attainment of 96.42% is 1.43 points above EvoMaster and 7.85 above the frozen agent (88.57%). The replay and validation that block invalid updates therefore do not trade against cost. Appendix F.8 gives the cost accounting behind these ratios, and Appendix G.3 defines the reliability metrics.

7 Conclusion

Across 23 disciplines, ScienceClaw's replay-gated linked Skill–Operator evolution yields sustained OOD gains, positive cross-family transfer, and stronger retention at a lower cost per gain, without updating any model parameters. These results support evaluating scientific-agent evolution by persistence, transfer, validity, and reproducibility, rather than isolated task scores. ScienceClaw-Eval measures scientific correctness, evolutionary gain, retention, cross-dataset transfer, and evolution cost through sequential streams and independent reset evaluation.

8 Limitations and Ethical Considerations

Automated evaluators may miss domain errors (Appendix D), and updates may transfer incorrectly. ScienceClaw should support, not replace, experts; high-stakes use requires provenance, licensing, privacy safeguards, and independent review. The 23-discipline scope and cross-dataset splits do not eliminate shared conventions or within-field variation. Reported trajectories are final snapshots, so broader claims need repeated runs and discipline-level intervals. Persistent executable updates can reuse errors and expand the attack surface, requiring versioned program snapshots, replay and rollback, sandboxing, least-privilege credentials, auditable provenance, and human approval for irreversible actions.

9 Generative AI Usage

Generative AI assisted language and figures; the authors verified and take responsibility for the paper.

References

[1] Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. 2026. Gepa: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, Vol. 2026. 8479–8565.

[2] Meysam Alizadeh, Mohsen Mosleh, Fabrizio Gilardi, Atoosa Kasirzadeh, and Joshua Tucker. 2026. AI Coding Agents Can Reproduce Social Science Findings. *arXiv preprint arXiv:2606.11447* (2026).

[3] Jonathan Bragg, Mike D'Arcy, Nishant Balepur, Dan Bareket, Bhavana Dalvi Mishra, Sergey Feldman, Dany Haddad, Jena Hwang, Peter Jansen, Varsha Kishore, et al. 2026. Astabench: Rigorous benchmarking of ai agents with a scientific research suite. In *International conference on learning representations*, Vol. 2026. 110136–110223.

[4] Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. 2023. Chemcrow: Augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376* (2023).

[5] Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, et al. 2025. Scienceagentbench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *International Conference on Learning Representations*, Vol. 2025. 96934–96990.

[6] Haipeng Ding, Yuexiang Xie, Zhewei Wei, Yaliang Li, and Bolin Ding. 2026. From Atomic Actions to Standard Operating Procedures: Iterative Tool Optimization for Self-Evolving LLM Agents. *arXiv preprint arXiv:2607.07321* (2026).

[7] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, et al. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, Vol. 2024. 23247–23275.

[8] Sihang Jiang, Lipeng Ma, Zhonghua Hong, Keyi Wang, Zhiyu Lu, Tengfei Wang, Shisong Chen, Jinghao Zhang, Tianjun Pan, Weijia Li, et al. 2026. SEA-Eval: A Benchmark for Evaluating Self-Evolving Agents Beyond Episodic Assessment. *arXiv preprint arXiv:2604.08988* (2026).

[9] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. 2021. Wilds: A benchmark of in-the-wild distribution shifts. In *International conference on machine learning*. PMLR, 5637–5664.

[10] Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, et al. 2026. Agentic harness engineering: Observability-driven automatic evolution of coding-agent harnesses. *arXiv preprint arXiv:2604.25850* (2026).

[11] Chen Ling, Pei Chen, Albert Guan, Jiaming Qu, Shayan Ali Akbar, Madhu Gopinathan, and Erwin Cornejo. 2026. PACE: Two-Timescale Self-Evolution for Small Language Model Agents. *arXiv preprint arXiv:2605.23019* (2026).

[12] Tianyu Liu, Allen Xin Wang, Antonia Panescu, Lisa Xinyi Chen, Wenxin Long, Xinyu Wei, Yueqian Jing, Ziyao Zeng, Jihang Chen, Sihan Jiang, et al. 2026. Benchmarking AI Agents for Addressing Scientific Challenges Across Scales. *arXiv preprint arXiv:2606.12736* (2026).

[13] David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems* 30 (2017).

[14] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. 2024. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292* (2024).

[15] Jiaxuan Lu, Ziyu Kong, Yemin Wang, Rong Fu, Haiyuan Wan, Cheng Yang, Wenjie Lou, Haoran Sun, Lilong Wang, Yankai Jiang, et al. 2026. Beyond static tools: Test-time tool evolution for scientific reasoning. *arXiv preprint arXiv:2601.07641* (2026).

[16] Alisia Lupidi, Bhavul Gauri, Thomas Simon Foster, Bassel Al Omari, Despoina Magka, Alberto Pepe, Alexis Audran-Reiss, Muna Aghamelu, Nicolas Baldwin, Lucia Cipolina-Kun, et al. 2026. Airs-bench: a suite of tasks for frontier ai research science agents. *arXiv preprint arXiv:2602.06855* (2026).

[17] Yougang Lyu, Xi Zhang, Xinhao Yi, Yuyue Zhao, Shuyu Guo, Wenxiang Hu, Jan Piotrowski, Jakub Kaliski, Jacopo Urbani, Zaiqiao Meng, et al. 2026. Evoscientist: Towards multi-agent evolving ai scientists for end-to-end scientific discovery. *arXiv preprint arXiv:2603.08127* (2026).

[18] Darwin Godel Machine. 2025. Open-Ended Evolution of Self-Improving Agents.

[19] Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. 2026. Trace2skill: Distill trajectory-local lessons into transferable agent skills. *arXiv preprint arXiv:2603.25158* (2026).

[20] Yujiong Shen, Yajie Yang, Zhiheng Xi, Binze Hu, Huayu Sha, Jiazheng Zhang, Qiyuan Peng, Junlin Shang, Jixuan Huang, Yutao Fan, et al. 2026. Sciagentgym: Benchmarking multi-step scientific tool-use in llm agents. *arXiv preprint arXiv:2602.12984* (2026).

[21] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems* 36 (2023), 8634–8652.

[22] NexusAgent Team. 2026. S1-NexusAgent: a Self-Evolving Agent Framework for Multidisciplinary Scientific Research. *arXiv preprint arXiv:2602.01550* (2026).

[23] Minyang Tian, Luyu Gao, Shizhuo D Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon, Yao Li, et al. 2024. Scicode: A research coding benchmark curated by scientists. *Advances in Neural Information Processing Systems* 37 (2024), 30624–30650.

[24] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291* (2023).

[25] Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. Scienceworld: Is your agent smarter than a 5th grader?. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 11279–11298.

[26] Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024. Executable code actions elicit better llm agents. *arXiv preprint arXiv:2402.01030* (2024).

[27] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. 2016. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific data* 3, 1 (2016), 160018.

[28] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155* (2023).

[29] Wanghan Xu, Shuo Li, Tianlin Ye, Qinglong Cao, Yixin Chen, Hengjian Gao, Yiheng Wang, Qi Li, Kun Li, Sheng Xu, et al. 2026. ResearchClawBench: A benchmark for end-to-end autonomous scientific research. *arXiv preprint arXiv:2606.07591* (2026).

[30] Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, et al. 2026. Skillopt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904* (2026).

[31] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629* (2022).

[32] Mingda Zhang, Haoran Luo, Tiesunlong Shen, Qika Lin, Xiaoying Tang, Rui Mao, and Erik Cambria. 2026. Flowsteer: Interactive agentic workflow orchestration via end-to-end reinforcement learning. *arXiv e-prints* (2026), arXiv–2602.

[33] Shi-Xin Zhang and Yu-Qin Chen. 2026. ORBIT-Q: Dual-axis benchmarking of autonomous agents in scientific quantum programming. *arXiv preprint arXiv:2607.03105* (2026).

[34] Mingming Zhao, Xiaokang Wei, Yuanqi Shao, Kaiwen Zhou, Lin Yang, Siwei Rao, Junhui Zhan, and Zhitang Chen. 2026. A^2Flow: Automating Agentic Workflow Generation via Self-Adaptive Abstraction Operators. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 40. 29930–29938.

[35] Congjie Zheng, Chuanyi Xue, Bin Liang, Jun Yang, and Changshui Zhang. 2026. SEAGym: An Evaluation Environment for Self-Evolving LLM Agents. *arXiv preprint arXiv:2606.17546* (2026).

[36] Xinyu Zhu, Yuzhu Cai, Zexi Liu, Cheng Wang, Fengyang Li, Wenkai Jin, Wanxu Liu, Zehao Bing, Bingyang Zheng, Jingyi Chai, et al. 2026. EvoMaster: A Foundational Evolving Agent Framework for Agentic Science at Scale. *arXiv preprint arXiv:2604.17406* (2026).

A Formal Properties of the Replay-Gated Update

This appendix states what the update rule of Section 3 guarantees. The results concern the rule as implemented (Appendix B) and mark where the implementation is stricter than the literal Eq. (3).

A.1 Setting and Conventions

Validation outcomes. For a program A and a validation episode $j \in \mathcal{D}_{\text{val}}$, let $u_j(A) = (z_j(A), s_j(A), \mathbf{h}_j(A), \mathbf{c}_j(A))$ collect the success indicator, a bounded normalized task score, the hard-constraint indicators, and the resource use of the reset solve of j under A. Write $M(A) = \text{MacroSR}(A)$ on $\mathcal{D}_{\text{val}}$ (Eq. (4)) and $S(A)$ for the mean normalized score. The implementation compares programs lexicographically,

$$A \succ A' \iff M(A) > M(A') \text{ or } \big(M(A) = M(A') \wedge S(A) - S(A') \geq \varepsilon\big), \tag{A.1}$$

with a score margin $\varepsilon > 0$ that exceeds the re-solve noise of a single episode. The literal rule $Q_{\text{val}} = M$ is the limit $\varepsilon = \infty$, where ties are never broken.

Lemma A.1 ($\succ$ is a strict order). *The relation $\succ$ is irreflexive and transitive.*

Proof. Irreflexivity: $M(A) > M(A)$ is false and $S(A) - S(A) = 0 < \varepsilon$. Transitivity: let $A \succ A'$ and $A' \succ A''$. Then $M(A) \geq M(A') \geq M(A'')$. If either inequality is strict, $M(A) > M(A'')$. Otherwise both are equalities and $S(A) - S(A'') \geq 2\varepsilon \geq \varepsilon$. □

Assumption A.2 (Fixed, lineage-disjoint partitions). $\mathcal{D}_{\text{src}}$, $\mathcal{D}_{\text{val}}$, $\mathcal{D}_{\text{ID}}$, and $\mathcal{D}_{\text{OOD}}$ are fixed before the first round and are pairwise disjoint in items and in lineage groups. $\mathcal{D}_{\text{rep}}$ consists of frozen copies of source episodes. Hidden labels of $\mathcal{D}_{\text{val}}$, $\mathcal{D}_{\text{ID}}$, and $\mathcal{D}_{\text{OOD}}$ stay inside the evaluator and are never part of the execution environment $D_{E,t}$.

Assumption A.3 (Recorded values). Eq. (3) compares the recorded validation value of a candidate with the recorded value of the incumbent. A candidate is solved once per validation episode whose retrieved program slice it changes; other episodes reuse the incumbent's recorded result. An episode cut short by an infrastructure error is retried and, if it still fails, counted as an error entry that blocks admission rather than as a failure ($z = 0$).

A.2 Safety Invariants of the Persisted Program

Proposition A.4 (Monotone validation value). *Let $V_r = (M(A_r), S(A_r))$ be the recorded value of the round-r program. For every r, either $A_{r+1} = A_r$ or $A_{r+1} \succ A_r$. Hence V_r is nondecreasing in lexicographic order and strictly increases at every promotion. Because M takes values in a finite set and S increases by at least ε among programs with equal M, the number of promotions against a fixed $\mathcal{D}_{\text{val}}$ is finite.*

Proof. By Eq. (3), $A_{r+1} \neq A_r$ only if the selected candidate strictly improves the recorded value, i.e. $A_{r+1} \succ A_r$ (Eq. (A.1)). By Lemma A.1 the relation composes along the chain. If M takes K values and $\bar{S}$ bounds S, at most $K - 1$ promotions raise M and at most $K\lfloor \bar{S}/\varepsilon \rfloor$ promotions raise S at constant M. □

Proposition A.5 (Feasibility is invariant). *Every persisted program $A_{r+1} \neq A_r$ satisfies, at promotion: (i) $R_{\text{src}}(A_{r+1}) = 1$; (ii) $\mathbf{H}_{\text{val}}(A_{r+1}) \succeq \mathbf{H}_{\text{val}}(A_r)$, that is, no validation episode shows a hard-constraint violation that the incumbent did not show; and (iii)*

$$\mathbf{C}_{\text{val}}(A_{r+1}) \preceq \mathbf{B}_{\text{abs}} \quad \textit{and} \quad \mathbf{C}_{\text{val}}(A_{r+1}) \preceq (1+\beta)\,\mathbf{C}_{\text{val}}(A_r), \tag{A.2}$$

where $\mathbf{B}_{\text{abs}}$ scales with $|\mathcal{D}_{\text{val}}|$ and $\beta \geq 0$. Consequently, the set of hard-constraint violations on any completed validation episode is nonincreasing along the chain, and the validation cost of every persisted program is at most $\mathbf{B}_{\text{abs}}$ whenever $\mathbf{C}_{\text{val}}(A_0) \preceq \mathbf{B}_{\text{abs}}$.

Proof. (i) and (iii) are conditions of the candidate set in Eq. (2), with the budget vector instantiated as Eq. (A.2), and (ii) is its hard-constraint condition in no-regression form; a candidate that violates any of them is never selected. By (ii), $\mathbf{h}_j(A_{r+1}) \geq \mathbf{h}_j(A_r)$ component-wise on every completed validation episode, so the violation set cannot grow. The cost bound follows from the first inequality of Eq. (A.2) and induction from A_0. □

Remark A.6 (Where the implementation is stricter than Eq. (3)). Four additions follow: the lexicographic order Eq. (A.1) instead of M alone; the relative budget cap in Eq. (A.2) in addition to an absolute cap; the noise guard of Proposition A.9; and the no-regression form $\succeq$ of the hard-constraint condition. The relative cap and the noise guard make admission harder, whereas the lexicographic order also admits a candidate that ties on M and raises S by at least ε, and the no-regression form admits a candidate that keeps a violation of the incumbent. Proposition A.4 holds for the literal rule because each literal promotion strictly raises M, and the consequences of Proposition A.5 hold for it because they use only (i), (ii), and the absolute cap in (iii). Appendix B.5 lists the parameter values.

Proposition A.7 (Information-flow isolation). *Under Assumption A.2, the program A_{r+1} is a function of A_r, the round-r source episode with its evaluator feedback, the validation evaluations of the candidate and the incumbent, and the solver's randomness. No quantity computed on $\mathcal{D}_{\text{ID}}$, $\mathcal{D}_{\text{OOD}}$, or $\mathcal{D}_{\text{rep}}$ is an input to the update, and $\Theta_{r+1} = \Theta_r = \Theta_0$.*

Proof. Candidates are built from the repair interval δ_i of a source episode (Eqs. (9)–(12)). Admission reads R_{src} (a reset run of the same source episode) and the validation values u_j in Eqs. (2) and (3). By Assumption A.2 no held-out label reaches the run environment, so held-out episodes can influence neither the solver's feedback nor the evaluator values that these equations consume. The parameters Θ_0 appear only as the fixed policy of Eq. (6). □

Corollary A.8 (No adaptive selection on held-out splits). *Snapshot scores on $\mathcal{D}_{\text{ID}}$ and $\mathcal{D}_{\text{OOD}}$ are not maximized over, or conditioned on, by any step of the procedure. They are single evaluations of a program chosen without them.*

A.3 The Noise Guard: What It Buys and What It Costs

The literal rule admits a candidate on any strict gain in Q_{val}. With one stochastic solve per validation episode, a single lucky episode suffices. The implementation therefore requires the gain to be supported by at least m individually improved episodes and by at most

v regressed ones.

Proposition A.9 (False promotion under the noise guard). *Consider a candidate whose effect on each of n affected validation episodes of one discipline is independent, with the episode improving with probability a, regressing with probability b, and unchanged otherwise. Let U and V be the numbers of improved and regressed episodes. The literal rule admits the candidate iff $U > V$. The guarded rule admits it iff $U > V$, $U \geq m$, and $V \leq v$. Then*

$$\Pr[\textit{guard admits}] \;\leq\; \Pr[U \geq m] \;\leq\; \binom{n}{m} a^m, \tag{A.3}$$

and for $n = m = 2$, $v = 1$,

$$\Pr[\textit{literal admits}] = a^2 + 2a(1-a-b), \qquad \Pr[\textit{guard admits}] = a^2. \tag{A.4}$$

In particular, when the candidate has no effect and noise is symmetric ($a = b = p$), the literal rule admits it with probability $2p - 3p^2$ and the guard with probability p^2, a ratio of $(2-3p)/p$.

Proof. The guard adds conditions to $U > V$, so its admission event is contained in $\{U \geq m\}$; the second inequality in Eq. (A.3) is the union bound over the $\binom{n}{m}$ subsets of m episodes that must all improve. For $n = 2$, $U > V$ holds in exactly two disjoint ways: both episodes improve (probability a^2), or exactly one improves and the other is unchanged (probability $2a(1-a-b)$). With $m = 2$ only the first survives, and then $V = 0 \leq v$. □

Remark A.10 (Illustration and price). For $p = 0.1$, a no-effect candidate passes the literal rule with probability 0.17 and the guard with probability 0.01. The price is power. A candidate that helps each episode with probability $a = 0.5$ and harms with $b = 0.1$ passes with probability 0.65 under the literal rule and 0.25 under the guard, so the guard also rejects some genuine one-episode improvements. These values come from the model above; the measured promotion counts are in Section 6 and Appendix G.

Corollary A.11 (Stagnation condition). *If $m > n$, the guard admits no candidate whatever its effect, because $\Pr[U \geq m] = 0$. A configuration with $m > n_{\text{val}}$ therefore never promotes a single-discipline candidate.*

Remark A.12 (When the program stalls). Corollary A.11 gives the configuration in which the gate stalls by construction. In any other configuration the program stalls at round r exactly when $\mathcal{F}_r$ contains no member that passes the guard and strictly improves validation value, so progress is set by the quality of the candidates and the validation set. Table 24 reports the measured promotion counts per method.

A.4 Operator Abstraction

An Operator candidate is a component U of the final graph G_i^+ replaced by one node with the boundary ∂U (Eq. (12)). Two properties make this replacement well defined.

Definition A.13 (Convex node set). $U \subseteq N$ is *convex* in a DAG G if every directed path in G between two nodes of U stays inside U.

Lemma A.14 (Contraction preserves acyclicity iff U is convex). *Let G be a DAG and U a nonempty node set. The graph G/U, obtained by contracting U into one node, is acyclic iff U is convex.*

Proof. ($\Leftarrow$) A cycle in G/U cannot avoid the contracted node, since G is acyclic. It lifts to a path in G that leaves U at some $v \in U$ through nodes outside U and returns to some $w \in U$. If $v \neq w$ this contradicts convexity; if $v = w$ it is a cycle in G. ($\Rightarrow$) If U is not convex, some path between nodes of U contains a maximal outside segment $x_1, \ldots, x_k$ entered from $a \in U$ and left toward $b \in U$. Then $U \to x_1 \to \ldots \to x_k \to U$ is a cycle in G/U. □

Corollary A.15 (A valid partition always exists). *Every singleton is convex, and the maximal weakly connected components of Eq. (10) can be refined along a topological order into convex pieces. A component whose boundary would still create a cycle is dropped and logged rather than registered.*

Proposition A.16 (Boundary replay is sound). *Let $\hat{o}_{i,U}$ pass boundary replay: executed on the recorded inputs at $\partial^- U$, its outputs at $\partial^+ U$ equal the recorded outputs in τ_i^+ exactly (deterministic nodes) or within δ in a norm. Then, with the nodes downstream of U unchanged, (i) the final output equals y_i^+ in the exact case; and (ii) if the map from the boundary outputs to y is L-Lipschitz, the final output differs from y_i^+ by at most $L\delta$.*

Proof. Evaluate the downstream nodes in topological order. In case (i) each receives identical inputs and, being deterministic, returns identical outputs, so identical y follows by induction. Case (ii) is the Lipschitz bound applied to the boundary difference. □

Remark A.17 (Boundary replay and the source check). Proposition A.16 shows that an Operator reproduces the component it abstracts on the recorded boundary. A candidate then reproduces its source task from a reset state with its component versions used ($R_{\text{src}} = 1$, Eq. (13)) and improves independent validation tasks, which extends the evidence from the recorded boundary to the full program and to other tasks.

A.5 Metrics: Invariance and Sampling Variability

Definition A.18 (Relative margin). For a discipline with task-native score s, a reference system with score $s_{\text{ref}} > 0$, and a method with score s_{M}, the relative margin is $(s_{\text{M}} - s_{\text{ref}})/s_{\text{ref}}$ if higher is better and $(s_{\text{ref}} - s_{\text{M}})/s_{\text{ref}}$ if lower is better.

Proposition A.19 (Unit invariance and sign). *The relative margin is unchanged under $s \mapsto cs$ for every $c > 0$ (for example, percent versus fraction), and it is positive iff the method is better than the reference in the metric's own direction. It is not invariant under translation $s \mapsto s + c$.*

Proof. Scaling multiplies numerator and denominator by c. The sign follows from the definition, and translation changes the denominator without changing the numerator. □

This is why Section 6 compares systems only within a discipline and averages relative margins, never raw scores, across disciplines.

Proposition A.20 (Sampling variability of MacroSR). *Suppose, for a fixed program, the outcomes z_i in discipline d are independent Bernoulli(p_d) over n_d instances. Then*

$$\mathrm{Var}\big[\mathrm{MacroSR}\big] = \frac{1}{23^2}\sum_{d=1}^{23}\frac{p_d(1-p_d)}{n_d} \;\le\; \frac{1}{4\cdot 23\,n} \quad \text{if } n_d = n \;\; \forall d. \tag{A.5}$$

For $n = 64$ the standard error is at most 1.30 percentage points, and for the difference of two programs evaluated on independent draws at most 1.84 points.

Proof. Independence across disciplines and instances gives the variance of the average. Each term is at most $1/(4n_d)$, with equality at $p_d = 1/2$; summing 23 terms and dividing by 23^2 gives $1/(92n)$, whose square root at $n = 64$ is 0.0130. The difference has variance at most twice this. □

Remark A.21 (Scale of sampling noise). The bound is the binomial sampling error of MacroSR over $23 \times 64 = 1{,}472$ instances per split. The +13.59 point OOD gain of ScienceClaw over A_0 is more than seven times the 1.84 point bound.

Remark A.22 (Summary of the guarantees). The persisted program is monotone in recorded validation value, feasible at every promotion, selected without access to the held-out splits, and built from components that reproduce the evidence they came from. The noise guard lowers false promotion at a cost in power, and operator replacement preserves acyclicity.

B Implementation and Configuration

This appendix specifies how the formalism of Sections 3 and 5 is operationalized: the objects of the paper and the modules that realize them, the execution runtime, the policy interface, the evolution procedure, the gate parameters, the models and budgets, and the configuration of every compared variant and baseline. The benchmark adapters, solver, runtime, and evolution engine are in the project repository (https://github.com/beita6969/ScienceClaw).

B.1 From Formalism to Modules

Table 3 maps each object to its module. Every run writes receipts: the resolved configuration, each program snapshot A_r, and, per episode, the trajectory, the final graph, the output, and the evaluation record, together with a ledger of tokens and wall time per phase.

B.2 Execution Runtime

Node kinds and schemas. A node is a *tool* call into the task's visible tools $\mathcal{T}_t$, a *code* node holding a Python function of its inputs and configuration, an *llm* node that maps a prompt template over a list of items, an *operator* node that expands a registered Operator, or the single *submit* node whose input is the required output y_t. A port schema Γ has a type (any, number, text, array, table, list, dict, series), a shape (integers or named symbols), a unit, and provenance. An edge is valid when the types agree or one is any, the shapes unify with symbols bound consistently, and the units are equal, unspecified on either side, or bridged by an explicit conversion (a factor and an offset) attached to the edge, which is recorded in provenance. A graph is valid when it is acyclic, all ports exist, every edge is compatible, and it has at most one submit node.

Table 3: Paper objects and the modules that implement them.

Paper object	Implementation
Task $D_t = (D_T, D_V, D_E)$	episode: objective and output schema, constraints and evaluator, tools and data
Program $A_r = (\mathcal{S}_r, \mathcal{O}_r)$	program of Skills and Operators with version identifiers
Fixed model Θ_0	policy role of the language-model client; never trained
Solve, Eq. (1)	solver: one episode, one program, one mode
Port schema Γ, Compat	schema type with a compatibility rule
Graph $G_{t,k}$, Eq. (5)	workflow graph with node and edge validation
Policy π_{Θ_0}, Eq. (6)	policy proposing one atomic action per turn
Retrieve	lexical retriever over Skills and Operators
Execute, Eq. (7)	executor with checkpoint and fingerprints
Replay, Eval, Eq. (8)	clean replay and the episode evaluator
δ_i, Eq. (9)	attribution of evolution instances
$\Pi_{\mathrm{ctrl}}, \Pi_{\mathrm{exec}}, \mathrm{CC}_\Gamma$	edit split with convex components
Skill candidate, Eq. (11)	skill patch
Operator candidate, BReplay	operator abstraction and boundary replay
R_{src}, Eq. (13)	source replay check with the use test
$\mathcal{F}_r$, Eq. (2)	validation gate
Update, Eq. (3)	evolver
MacroSR, Eq. (4)	benchmark metrics
$\mathcal{D}_{\mathrm{src}}, \mathcal{D}_{\mathrm{val}}, \mathcal{D}_{\mathrm{ID}}, \mathcal{D}_{\mathrm{OOD}}, \mathcal{D}_{\mathrm{rep}}$	split plan with frozen manifests

Checkpoints and fingerprints. Each node has a fingerprint, the hash of its specification, its upstream fingerprints, and its edge wiring. The checkpoint χ maps fingerprints to recorded outputs. After an edit only nodes whose fingerprint is absent from χ run again, in topological order, so an edit to one node recomputes its descendants and nothing else. A node whose outcome is not reproducible, such as a failed language-model call or a first timeout, is marked transient, and neither it nor any descendant enters the checkpoint.

Sandbox and time limits. Code nodes run in a separate process with their own working directory and a time limit, and tool nodes run in trusted adapter code under one lock. A node has a limit of max_node_s seconds, and an operator node as a whole has twice that limit. Hidden labels never enter the run environment: adapters keep them inside the evaluator. A static scan of code nodes rejects absolute paths and directory traversal, references to the dataset root and to the evaluator package, network, download, and process-spawning modules, dynamic evaluation and imports, and environment or credential access; its violations feed the integrity check in $\mathbf{H}_{\mathrm{val}}$, and the worker runs with a scrubbed environment.

Diagnostics. After each execution the executor returns per-node status and error text, output summaries (type, shape, unit, fraction of finite values, range, head), violated contract conditions, the visible constraint checks on the submit output, and the visible development score when the task provides one. These fields form $f_{t,k}$ in Eq. (7).

Reset replay. Replay executes the complete graph in a fresh run directory with an empty checkpoint. A result is reproducible when the replayed output matches the executor's output within the episode tolerance. A clean replay runs whenever a submit output differs in fingerprint from the last replayed one, and the sequence of replays gives the evidence $e_{t,k}$.

B.3 Policy Interface

Actions. At each turn the policy emits one JSON object with a short rationale, one action, and the identifiers $v_{t,k}$ of the Skills and Operators it used. The actions are `add_node`, `remove_node`, `modify_node` (with a patch to code, a find-and-replace code edit, prompt, configuration, ports, or wiring), `add_edge`, `remove_edge`, and `finish`. A rejected action leaves the graph unchanged and returns the reason as feedback. Edits that change only configuration, edges, node removal, or add tool, operator, or submit nodes are control edits (Π_{ctrl}), and edits that add or modify code or prompt nodes are executable edits (Π_{exec}).

What the policy sees. The system prompt describes what each action, node kind, and interface rule does, the deliverable format, and a uniform acceptance statement: a deliverable counts as solved when the task metric beats the task's reference by a fixed margin, all constraints hold, and the final workflow replays reproducibly within the budget. It names no metric value, reference method, or task-specific recipe. The initial program A_0 is empty; strategies reach the prompt only as retrieved Skills, and the prompt is scanned for strategy phrases to keep it so. Each turn adds the objective, the tool signatures with port descriptions, the retrieved Skills and Operators, the remaining budget, the feedback of the last step, and a compact history of earlier steps. The policy never sees hidden-label scores.

Retrieval. Skills and Operators are ranked against the episode query (objective, task type, tags, discipline, and required-output type) by a lexical Okapi BM25 over their text, plus bonuses for matching discipline, task type, tags, and, for Operators, input port types. A component is retrievable only if it matches the episode by metadata or covers at least 15% of the query's inverse-document frequency mass; ranking is deterministic, and the top 4 Skills and top 6 Operators enter the context. A Skill or Operator learned in one discipline stays retrievable in another when it is relevant to the episode, which is how cross-discipline transfer arises. The hash of the retrieved version identifiers and content digests is the *retrieval slice* of an episode, and a candidate is validated on an episode only when it changes that slice (Section B.5).

B.4 Evolution Procedure

One round processes one source episode per discipline in a fixed source order, so seven rounds give $7 \times 23 = 161$ source episodes and 161 candidates per method. The default schedule is per candidate: each candidate is validated immediately against the current incumbent, and the next source episode already uses the updated program. Snapshots $A_0, \dots, A_7$ are taken at round ends.

(1) **Solve the source episode** with A_r in source mode, which stops at the first replay-verified pass. The final workflow is the replay-verified passing graph with the best visible development score.
(2) **Extract the evolution instance.** The pass is e_i^+, the latest replay-verified failure before it is e_i^- (or none), and the actions between them form δ_i (Eq. (9)). Without e_i^-, only an Operator candidate is formed.
(3) **Split the repair.** Control edits give $\Delta_i^{\mathcal{S}}$. Executable nodes created or modified in δ_i, together with the executable nodes weakly connected to them, are grouped into maximal connected components that are made convex by splitting along a topological cut (Lemma A.14). Nodes expanded from a registered Operator are not candidates unless δ_i touched them.
(4) **Form candidates.** The Skill candidate patches the attributed Skills, or creates one if none was attributed, from the control edits and the failure and success evidence, without instance-specific values or answers. Each component becomes an Operator candidate: its body is the subgraph, concrete integer dimensions at its boundary are replaced by symbols so that instance sizes do not make it incompatible with other episodes, its contract κ records the types, shapes, units, finiteness, and ranges observed at the boundary together with episode tags, and its provenance records episode, steps, and parent program version. Names and descriptions are documentation only.
(5) **Boundary replay.** Recorded inputs at $\partial^- U$ are loaded, the Operator runs in isolation in a fresh directory, and its outputs at $\partial^+ U$ are compared with τ_i^+ within tolerance. Only passing Operators enter the bundle B_i.
(6) **Apply and check the source.** The bundle is applied atomically, assigning version identifiers ω_i, and the source episode is solved again from a reset state. $R_{\text{src}} = 1$ requires a passing replay-verified result and that every new version was used: an Operator version is used if an operator node with that reference is in the evidence graph and ran successfully, and a Skill version is used if it is in v of an applied step up to the evidence step. Citations in rejected or failed actions do not count.
(7) **Gate.** The candidate is evaluated on the validation episodes and admitted under the rules of Section B.5; otherwise the incumbent is kept.

B.5 Gate Parameters and Implementation Choices

Table 4 lists the parameters of the gate and of the benchmark protocol. The following choices fix details that Eqs. (2) and (3) leave open.

Choice of Q_{val} and $\mathbf{H}_{\text{val}}$. Q_{val} is MacroSR on $\mathcal{D}_{\text{val}}$ with ties broken by the mean normalized score, so a candidate that solves the same episodes but solves them measurably better can be admitted (Eq. (A.1)). $\mathbf{H}_{\text{val}}$ collects (a) no leakage or sandbox violation, (b) a valid output schema, and (c) no hard-constraint violation that the incumbent did not show. An episode that produced no output is already penalized by MacroSR and has no output to violate a constraint with, so it cannot veto a candidate through $\mathbf{H}_{\text{val}}$; a solver or gateway error does block admission.

Table 4: Gate and protocol parameters of all reported runs.

Parameter	Value
Rounds R	7
Items per episode	16
Validation episodes per discipline	2
ID and OOD episodes per discipline	4 each (64 instances)
Update schedule	per candidate
Q_{val}	MacroSR, then mean normalized score
Score margin ε	0.02
Hard-constraint mode $\mathbf{H}_{\text{val}}$	no new violation against the incumbent
Absolute token budget	250,000 per validation episode
Relative cost cap β	0.5
Wall-time budget	1,800 s per validation episode
Lazy re-validation	on
Noise guard (m, ν)	at least 2 improved, at most 1 regressed episode
Boundary-replay repeats	1
Evolution instances per source episode	1
Infrastructure retries	2, backoff from 30 s
Partition and sampling seed	20260928

Budget. Eq. (A.2) makes **B** concrete: tokens are logical (spent plus cached) so that a run served from the response cache is judged like the original run, and wall time is checked against its own budget. The violated condition is recorded with each rejected candidate.

Validation cost control. A validation episode is solved again only if the candidate changes its retrieval slice, and otherwise the incumbent's recorded result is reused. The number of re-solved and reused episodes is recorded per candidate, so the validation cost grows with the candidate rather than with the library.

Noise guard. With one stochastic solve per validation episode and two validation episodes per discipline, one lucky episode would admit a candidate under the literal rule. The implementation therefore requires the gain to be supported by at least two episodes that individually improved (the episode solves where it failed, or solves with a normalized-score gain of at least ε) and by at most one that regressed, comparing the entries already recorded and drawing no extra samples (Proposition A.9). Setting the first threshold to one restores the literal rule.

Infrastructure outages. A solve cut short by a gateway outage is retried with doubling backoff, then becomes an error entry that blocks admission and is never reused. A source solve is retried in the same way and, if it still fails, deferred once to the end of its round. Outages are recorded and are never scored as task failures.

B.6 Models, Sampling, and Budgets

Table 5 lists the models. The orchestrator is the only model that executes tasks, and its parameters never change. Evolution operates on source trajectories and evaluator feedback and has no access to validation, ID, or OOD labels.

Solver and episode budgets. A source or evaluation episode has at most 24 planner steps, 200,000 policy tokens, and 1,800 to 3,600 seconds of wall time depending on the task, with per-node limits of 180 to 900 seconds (Appendix C lists the per-discipline values). The six most recent steps are shown in full and earlier steps are summarized. Evaluation episodes run in evaluation mode: the final workflow is the latest valid, fully wired workflow, and no hidden verification happens per step. Table 4 lists the caps of a validation episode at the gate, and Table 11 the budget of a source or evaluation episode.

Table 5: Models and roles. All systems share the orchestrator.

Role	Setting
Orchestrator (policy, executor, patch)	Qwen3.5-27B, open weights, self-hosted OpenAI-compatible server, thinking disabled, temperature 0
Evolution	GPT-5.4 in isolated sessions
Qwen-only variant	Qwen3.5-27B in evolution
RuleEvo variant	no model in evolution; deterministic trace projection and compiler
Output limit per call	policy 8,000, executor 2,000, patch 4,000 tokens
Policy output format	JSON object
Response cache	on; identical contexts return identical outputs
Parameter updates	none, $\Theta_r = \Theta_0$ for all r

Compute. Orchestrator calls are served by self-hosted vLLM servers on H800 GPUs; the evaluator code of each task runs on CPU, and tasks that need a GPU, such as segmentation or source separation, call a GPU tool broker.

B.7 Configuration of the Ablation Variants

Table 6 lists the variants in four groups. The first group removes parts of the update scope, the second changes how candidates are produced, the third removes one component of the gate, and the last group restricts execution itself. Variants in the same group share everything except the named factor, so each difference in Table 2 and Figure 5 is attributable to that factor. The execution variants are enforced by the solver: in single-operator orchestration the graph may hold only tool nodes, one code node, and the submit node; in fixed-workflow orchestration only edits to the code node and its configuration are accepted; in single-turn orchestration the policy must emit the complete graph in one batch action.

B.8 Baseline Configuration and Fairness

Table 7 states the protocol. Each baseline runs its best released configuration through fixed ScienceClaw-Eval adapters, and adapters and initial assets are fixed without access to ID or OOD instances or evaluators. No method is tuned on $\mathcal{D}_{\text{ID}}$ or $\mathcal{D}_{\text{OOD}}$. Every baseline runs its own evolution calls on GPT-5.4, the evolution model of ScienceClaw, for as many evolution rounds as ScienceClaw.

C ScienceClaw-Eval Specification

This appendix documents how ScienceClaw-Eval is built and scored: task sources, partition rules, evaluator definitions, acceptance criteria, hard constraints, budgets, and the checks that keep evaluation honest. Per-discipline values are in Tables 10 and 11.

C.1 Construction Pipeline

Construction follows the four stages of Figure 4.

Task collection. Each of the 23 disciplines is one division of the ANZSRC 2020 Fields of Research (divisions 30 to 52), and its task is built on a public scientific resource with explicit inputs, a verifiable

Table 6: Configuration of the eighteen variants. Each variant changes one factor of the full system and leaves the runtime, budgets, source stream, and validation data unchanged.

Variant	Factor	Single change
ScienceClaw (full)	none	Dynamic workflow orchestration and linked Skill–Operator self-evolution.
Frozen	evolution	All persistent updates are off; runtime and budgets are unchanged.
Workflow only	scope of updates	Only the structured workflow-template and routing fields inside Skills are updated; other Skill text and Operators stay frozen.
w/o Skill evol.	scope of updates	Only ordinary Skill strategy text is frozen; the structured workflow fields and Operators may still change.
w/o Oper. evol.	scope of updates	Operator creation and fusion are disabled in the full system.
w/o Work. evol.	scope of updates	Only the workflow-template and routing fields inside Skills are frozen; ordinary Skill text and Operators may still change.
w/o link. evol.	linkage	Skills and Operators are updated separately, and not committed together as one atomic bundle.
RuleEvo	candidate source	Generative evolution is off; candidates come only from deterministic trace projection and a compiler.
Qwen-only	candidate source	Qwen replaces GPT-5.4 in evolution.
w/o src. replay	gate	Source replay still runs, but its result does not enter the admission rule.
w/o sci. constr.	gate	The agent and the admission rule do not see the scientific hard constraints; the final official scorer still checks them.
w/o ind. valid.	gate	The evolving agent scores its own candidates and freezes; the official scorer is applied afterward only for blind evaluation.
w/o cross-task mem.	memory	Reuse of Skills, Operators, and failure memory across tasks is off.
w/o IID select.	gate	No paired IID selection; the latest valid candidate is frozen.
w/o OOD transfer	transfer	The preregistered robustness pack is off; the entry is not available when no independent callable pack exists.
Fixed workflow	orchestration	Dynamic canvas orchestration is off; a fixed tool chain executes.
Single operator	orchestration	Each sample may call exactly one task Operator.
Single turn	orchestration	One model generation per sample, with no feedback, rollback, or replanning.

Table 7: Baseline protocol. Every baseline runs on the same orchestrator, evolution model (GPT-5.4), tools, source stream, validation data, and update budget as ScienceClaw.

Method	What it evolves	Fixed across methods	Method-specific setup
Frozen (A_0)	nothing	orchestrator, tools, budgets	none
SkillOpt	Skills	orchestrator, evolution model, tools, source stream, validation data, update budget	best released configuration, same evolution rounds as ScienceClaw
TTE	tools at test time	same	same
AHE	agent harness	same	same
EvoMaster	programs and memory	same	same
EvoScientist	programs and memory	same	same
ScienceClaw	linked Skill–Operator program	same	default configuration of Appendix B.5

target, and a recorded license. Every resource is pinned to a version, release, or commit, and its files are checked against recorded checksums when the benchmark loads.

Executable instantiation. A task adapter turns the resource into an episode $D_t = (D_T, D_V, D_E)$. D_T is the objective and the required output with its schema. D_E is the execution environment: visible training and development data, the scientific tools of the task, and, where the discipline has one, a domain library of documented functions. D_V is the evaluator, its hard constraints, and its acceptance rule. An episode contains 16 evaluation items, so a split of four episodes holds 64 instances.

Validation. Domain reviewers inspect each evaluator, its constraints, and its reference execution (Appendix D). Reset replay then confirms that a reference solution executes, scores as expected, and reproduces.

Evaluation partitioning. Partitions are fixed before any run, drawn with a seeded permutation that is stable in the number of episodes, and written to manifests that are hashed and verified on every load. Items inside an episode are drawn without replacement, and no item is reused across episodes of a held-out split.

Table 8: Splits of one discipline at the default plan.

Split	Episodes	Items	Role
$\mathcal{D}_{\text{src}}$	7 (one per round)	16 each	evolution evidence
$\mathcal{D}_{\text{val}}$	2	16 each	candidate selection
$\mathcal{D}_{\text{ID}}$	4	64 instances	held-out, same source
$\mathcal{D}_{\text{OOD}}$	4	64 instances	held-out source
$\mathcal{D}_{\text{rep}}$	frozen copies of source episodes seen	16 each	retention

C.2 Splits and Source Stream

Table 8 gives the split sizes. Across 23 disciplines the source stream has $7 \times 23 = 161$ episodes, the same for every method. Round r presents one source episode per discipline in a fixed discipline order, and a source pool smaller than seven episodes is recycled with

its items marked as reused. Snapshots $A_0, \ldots, A_7$ are evaluated after each round. The replay set $\mathcal{D}_{\text{rep}}$ at an evaluation point consists of frozen copies of the source episodes encountered so far, so retention is measured on tasks that supplied evolution evidence while no new evidence is drawn from them.

Disjointness. Splits are disjoint in items and in a discipline-specific lineage unit, such as the capture in FoR30, the track in FoR36, the series in FoR35, the student in FoR39, the simulated dataset in FoR44, and the contract in FoR48. The loader re-verifies that no item, capture, or lineage identifier occurs in two roles whenever the roles are loaded, and a violation prevents the benchmark from running.

Held-out sources. $\mathcal{D}_{\text{OOD}}$ of every discipline is held out by source, by year in FoR37 and by economy in FoR38 (Table 9). FoR30 OOD is scored by the leaf-instance panoptic quality at IoU ≥ 0.5.

C.3 Evaluator Definition

Success indicator. For instance i, $z_i(A_r) = 1$ iff the run completes within budget, the task-native metric meets the acceptance rule, every hard constraint holds, and the workflow replays reproducibly (Section 4.2). The task-native metric is evaluated for each instance, and the acceptance rule compares it with a reference predictor.

Reference predictors and margins. Every task has a deterministic reference predictor computed from the visible data of the same episode (Table 10). The acceptance rule is *the metric beats the reference by a fixed margin*, either additive or multiplicative, and the margin is fixed per task before any run. The reference itself is never accepted. The policy is told only that the task metric has to beat the task's reference by a fixed margin; the reference method, the margin, and metric values are not part of any prompt, objective, or tool description, and automated tests scan the episodes of the forecasting tasks to confirm it.

Official scorers. Where a task has an official scorer, the evaluator calls or reproduces it: FoR30 uses the PhenoBench evaluator at a pinned commit, FoR42 the official sepsis-score script, FoR47 the CoNLL 2018 rules for valid trees, FoR48 the ContractNLI scorer (the implementation of precision at recall 80% agrees exactly with the official function in its tests), and FoR35 reproduces the published seasonal-naive MASE of the Monash archive (1.631) in its tests. FoR44 is cross-checked against the data team's R outputs (the OLS coefficient equals R's `lm` within 10^{-8}, and the true effect and response standard deviation are identical).

Failure semantics. A held-out episode never drops out of a pooled metric. An episode whose solve produced no output, an invalid output, or a crashed evaluator contributes the adapter's reference payload, so it counts as reference level, never better, with $z = 0$, and it is flagged in the records. Coverage is checked against the episodes the split promises, and an incomplete discipline is excluded from a comparison rather than compared on different episode sets. An infrastructure failure, such as a gateway outage, is not a task failure: it is logged and the episode is retried at the next evaluation run.

C.4 Per-Discipline Specification

C.5 Task Cards

Table 12 states, for each discipline, the data a policy may inspect, the extra tools of the episode, the domain toolkit that the library can draw on, and the form of the required output y. Every episode exposes the same four generic tools: `load_eval_inputs` returns the 16 evaluation items without labels, `load_train` returns labelled examples drawn from the source role of the episode, `load_dev_inputs` returns further unlabelled items, and `score_dev` returns the visible development signal and the score of the reference predictor on it. The columns of the table list only what is specific to a discipline. The toolkit entries are modules of the shared library whose description is appended to the episode objective.

Two design choices apply across the cards. First, the data a policy sees are those of the source role: the held-out evaluation items never appear in `load_train`. Second, tool outputs that would reveal the answer are removed or anonymized, for example the series identity of FoR38 and the target answers of FoR39, which the query tool never reveals.

C.6 Leakage Controls

Visible data. The policy sees an opaque handle and the inputs of each item, never a label, date, or lineage identifier. Training and development data come from the source roles only and are disjoint from every evaluated item; the visible development score is computed on a development slice carved from visible data.

Items scored together. Items of one episode are scored together and all of their inputs are visible, so an item's input must not be a near-future observation of another item's label. Forecasting adapters enforce a minimum spacing when an episode is drawn: FoR37 episodes are lanes of 16 initializations at least 120 hours apart, and FoR33 windows of one building are at least 144 hours apart.

Causality and queries. FoR42 checks causality with a hidden probe on prefixes of the stay in addition to a visible check against positional-only outputs, and FoR39 takes the query budget from signed receipts of the whole solve, failing closed when they are missing. In FoR38 economy identifiers, series kinds, and periods are opaque.

Sandbox. Generated code is scanned and runs in a separate process with a scrubbed environment (Appendix B.2). Hidden labels live inside the evaluator and never in the run directory.

D Human Audit of Evaluators

Each discipline was assigned three PhD reviewers, and one expert from the natural or social sciences audited every evaluator and its reference executions across disciplines. The reviewers read the task card and the evaluator, ran the reference predictor, and checked the metric direction and unit, the acceptance margin, the hard constraints, and the leakage controls (Appendix C.6). When two reviewers disagreed, the cross-field expert read both reasons and the code involved and decided. A defect was fixed in the adapter or in the acceptance rule, and an unfixable property of a source benchmark was recorded in the task card.

Table 9: OOD source of each discipline.

FoR	OOD source	FoR	OOD source	FoR	OOD source
30	CropAndWeedAndLeaf	38	WDI, other economies, origin 2015	46	SWE-bench Verified
31	TAPE fluorescence	39	EdNet-KT1	47	UD Indonesian, Swedish Sign Language, French, Hindi
32	Dryad hippocampus	40	ToyNscale	48	ACORD
33	EULP	41	USGS river metabolism	49	SMT-LIB 2024 QF_NIA
34	BACE	42	SepsisExp	50	ETHICS
35	Monash Tourism Quarterly	43	ICDAR2019 POCR	51	PhononDB
36	MoisesDB	44	IHDP	52	Psych-101
37	WeatherBench 2 (ERA5, 2020)	45	BLOOM Vist MaMa		

Table 10: Task sources, metrics, references, and acceptance rules. ↑ and ↓ give the direction of the metric, s the score of an instance, and s_{ref} that of its reference.

FoR	Discipline	IID source (version)	Item	Metric	Reference predictor	Acceptance
30	Agricultural, veterinary and food	PhenoBench v1.1.0	image	PQ+ ↑	excess-green index with Otsu threshold	$s > s_{\text{ref}} + 2.0$
31	Biological	ProteinGym v1.3	assay	Spearman ↑	mean training score of the mutated position	$s > s_{\text{ref}} + 0.03$
32	Biomedical and clinical	MSD Task04 Hippocampus	volume	DSC ↑	location-only probabilistic atlas	$s > s_{\text{ref}} + 0.03$
33	Built environment	BuildingsBench v1.0.0	building window	NRMSE (%) ↓	previous-day persistence	$s \leq 0.90\, s_{\text{ref}}$
34	Chemical	OGB `ogbg-molhiv`	molecule	ROC-AUC ↑	logistic regression on 11 graph-count features	$s \geq s_{\text{ref}} + 0.05$
35	Commerce and services	Monash Tourism Monthly	series	MASE ↓	seasonal naive	$s \leq 0.95\, s_{\text{ref}}$
36	Creative arts and writing	MUSDB18 v1.0.0	6-s excerpt	SDR (dB) ↑	mixture as the estimate of every stem	$s \geq s_{\text{ref}} + 3$ dB
37	Earth	WeatherBench 2 (ERA5, 2019)	initialization	RMSE (K) ↓	24-h persistence	$s \leq 0.96\, s_{\text{ref}}$
38	Economics	World Bank WDI, origin 2012	economy series	sMAPE (%) ↓	no-change forecast	$s \leq 0.97\, s_{\text{ref}}$
39	Education	Eedi NeurIPS 2020 Task 4	student	10-mask acc. ↑	most common training correctness per question	$s \geq s_{\text{ref}} + 0.02$
40	Engineering	DCASE 2024 Task 2	audio clip	DCASE ↑	nearest-neighbour distance on log-mel statistics	$s \geq s_{\text{ref}} + 0.02$
41	Environmental	NEON aquatics (neon4cast)	site and date	CRPS ↓	day-of-year climatology	$s \leq 0.95\, s_{\text{ref}}$
42	Health	PhysioNet/CinC 2019 sepsis	patient stay	Clin. utility ↑	balanced logistic regression on trivial causal features	$s > \max(s_{\text{ref}}, 0) + 0.05$
43	History and archaeology	HIPE-OCRepair 2026 (v0.9.5)	OCR text	cMER-micro ↓	copy of the OCR input	$s \leq 0.95\, s_{\text{ref}}$
44	Human society	ACIC 2016	simulated dataset	nRMSE ↓	OLS coefficient of the treatment	$s \leq 0.90\, s_{\text{ref}}$
45	Indigenous studies	AmericasNLP 2026	image caption	chrF++ ↑	per-language medoid caption	$s \geq s_{\text{ref}} + 1.0$
46	Computing	HumanEval	problem	pass@1 ↑	memorizer of visible examples	$s \geq s_{\text{ref}} + 0.5$
47	Language and culture	UD 2.2 Marathi, Italian, Arabic, Croatian	sentence	LAS ↑	right-branching chain with frequent relations	$s \geq s_{\text{ref}} + 0.15$
48	Law	ContractNLI	contract–hypothesis pair	mAP ↑	lexical overlap scores with majority label	$s \geq s_{\text{ref}} + 0.10$
49	Mathematical	SMT-LIB 2025 QF_NIA, QF_NIRA	formula	Oracle acc. ↑	majority status of training labels	$s \geq s_{\text{ref}} + 0.2$
50	Philosophy and religion	Touché23-ValueEval	argument	F1 ↑	predict all 20 categories	$s \geq s_{\text{ref}} + 0.05$
51	Physical	Matbench phonons v0.1	crystal	MAE (cm^{-1}) ↓	5-nearest-neighbour regression	$s \leq 0.8\, s_{\text{ref}}$
52	Psychology	Psych-201 (discrete)	session	Micro acc. ↑	most frequent earlier response	$s \geq s_{\text{ref}} + 0.05$

Verdict agreement. In four disciplines (FoR31, 38, 46, and 48), the PhD reviewers also judged ten run records on source tasks each (five evaluator passes and five fails), and Table 13 compares their verdicts with those of the evaluator. The two agree on 37 of the 40 records (92.5%), with no missed detection (an evaluator pass that the reviewers reject) and three false rejections (an evaluator fail that the reviewers accept), and each false rejection lies at the acceptance boundary of its metric. Tables 15 and 16 list every verdict with its reason.

Readings fixed in advance. Several task statements admit more than one reading. The hard constraints of Table 11 fix such readings before any run, so that an output that depends on a silent choice is rejected instead of scored. Table 14 lists typical cases.

E Case Studies

This appendix reports two disciplines of different families and data types: Commerce and services (FoR35, time series, MASE) and Law (FoR48, text, mAP). The task cards, reference predictors, acceptance rules, and hard constraints are in Appendix C.

Table 11: Hard constraints and budgets. Every task also fixes 24 planner steps and 200,000 policy tokens per episode. The family column gives the group used in the transfer matrix (Figure 6b).

FoR	Family	Hard constraints	Wall (s)	Node (s)
30	Life/health	output structure; integer labels in range; declared unit	3600	900
31	Life/health	output shape; finite values; declared unit	1800	300
32	Life/health	output structure; labels in $\{0, 1, 2\}$; declared unit	2400	600
33	Eng./computing	output shape; finite; $0 \leq y \leq 1300$ kWh and a scale guard against unit errors; declared unit	1800	300
34	Phys./Earth	output shape; finite; probabilities in $[0, 1]$	1800	300
35	Social/behav.	output shape; finite; $y \geq 0$ with a scale guard; declared unit	1800	300
36	Hum./law	output shape; finite; no silent estimate over a whole 1-s window	2400	600
37	Phys./Earth	output shape; finite; 150 to 350 K (catches degrees Celsius); unit K	1800	300
38	Social/behav.	output shape; finite; range by series kind; declared unit	1800	300
39	Social/behav.	values in $\{-1, 0, 1\}$ on every target cell; query budget from signed receipts	1800	300
40	Eng./computing	score vector of the right length; finite	1800	300
41	Phys./Earth	output structure; finite; positive standard deviations; physical ranges (catches kelvin)	1800	300
42	Life/health	output structure; binary values; not positional only; hidden causal-prefix probe; declared unit	1800	300
43	Hum./law	output length; strings; length within 0.5 to 2 times the OCR input	3600	900
44	Social/behav.	output shape; finite; $\lvert\hat{r}\rvert \leq 5\,\mathrm{sd}(y)$	1800	600
45	Hum./law	output length; non-empty captions of at most 1000 characters	1800	600
46	Eng./computing	list of 16 strings; every problem implemented beyond a docstring	1800	180
47	Hum./law	output length; one head and one relation per word; valid tree; 37 UD relations	2400	600
48	Hum./law	output length; one finite score in $[0, 1]$ per span; valid labels	2400	600
49	Eng./computing	list of 16 strings; labels in {sat, unsat, unknown}	2400	900
50	Hum./law	output shape 16×20; binary entries	1800	600
51	Phys./Earth	output shape; finite; frequencies in 10 to 5000 cm^{-1}; unit conversion edge required	1800	300
52	Social/behav.	list of 16 strings; each label among the item's options	1800	300

E.1 Commerce and Services (FoR35)

The policy forecasts the next 24 months of 16 tourism series in the native unit of each series. The reference predictor is the seasonal-naive forecast, and an instance is accepted if its MASE is at most 0.95 times that of the reference. The hard constraints require a finite, non-negative forecast of the right shape, with a scale guard against unit errors. Table 17 lists the held-out MASE of every system, and ScienceClaw is lowest on both splits.

E.2 Law (FoR48)

The policy scores every span of a contract against a hypothesis. The reference predictor combines lexical overlap with the majority label, and an instance is accepted if its mAP exceeds the reference by 0.10. The hard constraints require one finite score in $[0, 1]$ per span and valid labels. Table 17 lists the held-out mAP of every system, and ScienceClaw is highest on both splits.

Table 18 gives the gain over the frozen agent of the fixed-workflow and unlinked variants. On both disciplines and both splits the linked system gains more than either.

F Scope, Limits, and Further Analyses

This appendix adds notes on the benchmark, the gate, the foundation model, the instance counts, the notation, and the compute.

F.1 Scope of ScienceClaw-Eval

A task of ScienceClaw-Eval is a computational task whose output can be scored by an executable evaluator inside a sandbox: a structured prediction, a forecast, a segmentation, a program, a parse, or a set of labels, together with the trace that produced it. The 23 tasks span the 23 disciplines of the field-of-research classification used in the main text, and each is built from a published benchmark with an official or reconstructed scorer (Table 10). A task enters the framework through $D_t = (D_{T,t}, D_{V,t}, D_{E,t})$ and requires an evaluator that checks a replayed output against $D_{V,t}$.

F.2 What the Evaluators Check

An evaluator decides acceptance from the metric of an instance against a reference predictor, from hard constraints on the output, and from the failure semantics of the run. Three layers narrow the gap between acceptance and validity. The hard constraints catch the silent errors that a metric hides, such as a unit error, a forecast in the wrong scale, or a positional answer (Table 11). The leakage controls remove tool outputs that would reveal an answer (Appendix C.6). The acceptance rules require a policy to exceed a reference predictor by a stated margin (Table 10). The human audit of the evaluators is described in Appendix D.

F.3 Selectivity and Progress of the Gate

The gate is strict by design (Appendix A.3). Of the 161 candidates of ScienceClaw, 70 were promoted (43.48%) and 91 were rejected. Table 19 lists the change of the OOD macro score per round. The score rises in six of seven rounds and falls only in round 3, by 0.20 pp.

In the seven rounds of the benchmark, each of the last three rounds still adds between 1.43 and 2.24 pp. The components of the gate matter for the final score. Removing source replay, the scientific constraints, or the independent validator lowers the score of the full system by 6.18, 7.33, and 5.68 percent on average over

Table 12: Task cards. The generic tools are listed in the text. The toolkit column names the module of the shared scientific library available to the task. The last column gives the structure of the required output y for the 16 items of an episode (k is the number of items, n the length of an item).

FoR	Discipline	Extra tools	Toolkit	Required output y
30	Agricultural, veterinary and food	none	`phenoseg`	dict of three integer arrays $(k, 512, 512)$: semantic labels $\{0, 1, 2\}$, plant instances, leaf instances
31	Biological	`substitution_features`	`proteinfit`	one float per query row, higher means fitter
32	Biomedical and clinical	none	`hippo`	list of integer arrays with the shape of each volume, values in $\{0, 1, 2\}$
33	Built environment	`load_history` (own building)	`loadforecast`, `tsfm`	array $(k, 24)$ of hourly energy in kWh
34	Chemical	`featurize_molecules`	`molecules`	vector of k probabilities
35	Commerce and services	none	`forecast`, `tsfm`	array $(k, 24)$ in the unit of the series
36	Creative arts and writing	none	`audiosep`	array $(k, 4, 132300, 2)$ of stem estimates
37	Earth	none	`weather`	array $(k, 64, 32)$ of temperatures in K on a longitude–latitude grid
38	Economics	none (anonymized series)	`macro`	array $(k, 4)$ of levels at offsets 1 to 4
39	Education	`query_answers` (budgeted, signed receipts)	`adaptive`	integer array $(k, 948)$ with values $\{-1, 0, 1\}$ on every target cell
40	Engineering	`log_mel_spectrogram`	`anomsound`, `audioenc`	one anomaly score per clip
41	Environmental	none	`aquatics`, `tsfm`	dict with oxygen and temperature means and standard deviations, each $(k, 30)$
42	Health	none	`sepsis`	list of binary arrays, one value per hour of each stay
43	History and archaeology	`text_mer`	`ocrfix`	list of k corrected transcriptions
44	Human society	`load_covariates`	`causal`	one effect estimate per dataset
45	Indigenous studies	`image_features`, `image_knn_reference`	`captions`, `clip_retrieval`	list of captions in the language of each image (16 items per episode)
46	Computing	`run_visible_tests`	`codegen`	list of k program strings
47	Language and culture	`read_conllu`	`udparse`	per sentence, a head and a relation for each word
48	Law	`load_hypotheses`	`contracts`, `textenc`	per pair, an entailment label and one score in $[0, 1]$ per candidate span
49	Mathematical	`z3_check` (timeouts, status)	`logic`	list of k labels in {sat, unsat, unknown}
50	Philosophy and religion	`load_value_taxonomy`	`valueeval`	binary matrix $(k, 20)$
51	Physical	`featurize_structures`	`matphonon`	one frequency in cm^{-1} per crystal
52	Psychology	none	`psych`	list of k labels, each among the options of the item

Table 13: Agreement between the evaluator and the PhD reviewers on ten run records on source tasks per discipline (five evaluator passes and five fails).

Discipline	FoR	Records	Agree	Rate	Missed	False rej.
Biological	31	10	9	90%	0	1
Computing	46	10	10	100%	0	0
Law	48	10	9	90%	0	1
Economics	38	10	9	90%	0	1
Total		40	37	92.5%	0	3

the two splits, and removing the IID selection by 11.64 percent, against 13.68 percent for the frozen agent (Table 22).

F.4 Foundation Model and Zero-Shot Performance

Every system of the benchmark uses the same foundation model for orchestration: Qwen3.5-27B with open weights, served by a self-hosted inference server with thinking disabled and temperature 0 (Table 5). The zero-shot reference is the Frozen agent, which runs the initial program A_0 with this model. Its OOD macro score is 77.72% (Table 24), and ScienceClaw reaches 91.30% after seven rounds.

Evolution uses GPT-5.4 in the main configuration (Table 5). The

Table 14: Typical ambiguities and the hard constraint that resolves each.

FoR	Ambiguity	Resolution
37	kelvin or degrees Celsius	values must lie in 150 to 350 K
41	kelvin or degrees Celsius	physical ranges with positive standard deviations
33	kWh or another energy unit	$0 \leq y \leq 1300$ kWh with a scale guard
35	unit and scale of a series	$y \geq 0$ with a scale guard
36	whether a silent segment is estimated	no silent estimate over a whole 1-s window
42	whether an answer may follow position only	positional-only answers are rejected
46	whether a stub counts as an implementation	every problem implemented beyond a docstring
51	wavenumber unit	frequencies in 10 to 5000 cm^{-1}

Qwen-only variant runs evolution on the orchestrator model instead. It retains 43.8% of the gain of the full system (Table 22), and it is better than the frozen agent in 44 of the 46 discipline–split cells; the exceptions are FoR40 and FoR44 on OOD. A stronger evolution model therefore raises the quality of the candidates, while the gate and the replay set are the same for both models. Every baseline also runs its evolution calls on GPT-5.4 (Appendix B.8), so the Qwen-only variant is the one comparison in which the evolution model differs from that of the other evolving methods.

Table 15: Verdicts of the evaluator and the PhD reviewers on the run records on source tasks of FoR31 and FoR46.

Record	Evaluator	Reviewer	Agree	Type	Reason
Biological (FoR31)					
FoR31-01	Pass	Pass	yes	–	Output shape, finiteness, and units are valid, and the metric reaches the threshold.
FoR31-02	Pass	Pass	yes	–	Uses the visible training data, with a sound procedure and no hidden labels.
FoR31-03	Pass	Pass	yes	–	Predictions are complete, and the method follows protein-prediction practice.
FoR31-04	Pass	Pass	yes	–	Output length is correct and the values are stable.
FoR31-05	Pass	Pass	yes	–	Legal tool calls, and the Spearman metric meets the requirement.
FoR31-06	Fail	Fail	yes	–	The metric does not reach the reference plus the threshold.
FoR31-07	Fail	Fail	yes	–	The output contains NaN or Inf.
FoR31-08	Fail	Pass	no	false rejection	Sound output and procedure; the metric is at the acceptance boundary, where the evaluator is stricter.
FoR31-09	Fail	Fail	yes	–	Output length differs from the number of targets.
FoR31-10	Fail	Fail	yes	–	Constant prediction, with insufficient metric and domain plausibility.
Computing (FoR46)					
FoR46-01	Pass	Pass	yes	–	All 16 functions are implemented and the visible tests pass.
FoR46-02	Pass	Pass	yes	–	Function signatures and return formats are correct.
FoR46-03	Pass	Pass	yes	–	Boundary conditions are handled sensibly, and no hidden answers are used.
FoR46-04	Pass	Pass	yes	–	The code logic is complete and the result is reproducible.
FoR46-05	Pass	Pass	yes	–	The visible tests and the task requirements are both satisfied.
FoR46-06	Fail	Fail	yes	–	A function is left unimplemented.
FoR46-07	Fail	Fail	yes	–	The return type is wrong, so the task interface cannot be passed.
FoR46-08	Fail	Fail	yes	–	Some function signatures do not match.
FoR46-09	Fail	Fail	yes	–	A clear semantic error.
FoR46-10	Fail	Fail	yes	–	Depends on the external network, so independent reproduction cannot be guaranteed.

Table 16: Verdicts of the evaluator and the PhD reviewers on the run records on source tasks of FoR48 and FoR38.

Record	Evaluator	Reviewer	Agree	Type	Reason
Law (FoR48)					
FoR48-01	Pass	Pass	yes	–	Score range, length, and labels are all valid.
FoR48-02	Pass	Pass	yes	–	Evidence spans correspond to the contract text.
FoR48-03	Pass	Pass	yes	–	The legal interpretation agrees with the evidence.
FoR48-04	Pass	Pass	yes	–	The mAP reaches the acceptance threshold.
FoR48-05	Pass	Pass	yes	–	The output format is complete and no hidden information is used.
FoR48-06	Fail	Fail	yes	–	Some contract spans have no score.
FoR48-07	Fail	Fail	yes	–	A score falls outside $[0, 1]$.
FoR48-08	Fail	Fail	yes	–	The evidence ranking and the legal conclusion do not match.
FoR48-09	Fail	Pass	no	false rejection	Sound binary output and evidence; the mAP misses the threshold only through a boundary difference.
FoR48-10	Fail	Fail	yes	–	The legal polarity is wrong.
Economics (FoR38)					
FoR38-01	Pass	Pass	yes	–	Output shape, units, and value range are correct.
FoR38-02	Pass	Pass	yes	–	Only data before the forecast origin is used.
FoR38-03	Pass	Pass	yes	–	The macroeconomic variables relate sensibly and the metric is met.
FoR38-04	Pass	Pass	yes	–	The procedure is complete and the result is reproducible.
FoR38-05	Pass	Pass	yes	–	No prediction is clearly out of range.
FoR38-06	Fail	Fail	yes	–	The sMAPE does not meet the requirement.
FoR38-07	Fail	Fail	yes	–	The GDP or unemployment forecast is out of range.
FoR38-08	Fail	Pass	no	false rejection	The output is valid, and the metric exceeds the threshold only slightly because of rounding.
FoR38-09	Fail	Fail	yes	–	The number of forecast steps or variables is wrong.
FoR38-10	Fail	Fail	yes	–	Uses future information or a physically implausible negative value.

F.5 Instance Counts and Sampling Error

Every discipline contributes 64 instances to each of the ID and OOD splits (16 items in each of four episodes; Table 8). The MacroSR over the 23 disciplines is a mean over these instances, and Proposition A.20 bounds its standard error by 1.30 pp at $n = 64$ per discipline, and the standard error of a difference of two methods on independent draws by 1.84 pp. The margins of ScienceClaw over the strongest baseline in a discipline range from 1.82 to 3.73 percent (ID) and from 3.05 to 6.21 percent (OOD) on relative scale (Table 23), and the margin over the frozen agent ranges from 7.96 to 16.45 (ID) and from 11.57 to 23.34 (OOD).

F.6 Reading the Notation

Table 20 gives, for the equations of Sections 3 and 5, the plain-language reading and the object that realizes the symbol in the implementation (Table 3).

Table 17: Held-out scores of the two case disciplines. MASE is lower-is-better and mAP is higher-is-better.

	FoR35 (MASE ↓)		FoR48 (mAP ↑)	
System	IID	OOD	IID	OOD
Frozen	1.6009	1.8740	0.7513	0.6918
SkillOpt	1.5268	1.6867	0.8158	0.7652
TTE	1.5671	1.7502	0.8074	0.7706
AHE	1.5227	1.7066	0.8211	0.7720
EvoMaster	1.4984	1.6930	0.8292	0.7863
EvoScientist	1.5281	1.6846	0.8170	0.7642
ScienceClaw	1.4653	1.5986	0.8554	0.8289

Table 18: Gain over the frozen agent (%) of the linkage variants on the two case disciplines.

	FoR35		FoR48	
Variant	IID	OOD	IID	OOD
Fixed workflow	3.77	5.75	1.07	6.20
w/o linked evolution	4.86	9.27	10.63	13.12
ScienceClaw (full)	8.47	14.69	13.86	19.82

Table 19: Change of the OOD macro score of ScienceClaw per round (pp) with the cumulative score (%).

Round	1	2	3	4	5	6	7
Change (pp)	+2.72	+2.24	−0.20	+3.67	+1.43	+2.24	+1.49
Score (%)	80.43	82.68	82.47	86.14	87.57	89.81	91.30

F.7 Program-Level Self-Evolution and Genetic Programming

ScienceClaw evolves the persistent program of an agent, and genetic programming evolves a population of programs, so the two share the idea of improving executable artifacts by selection. Table 21 lists the differences that matter for the claims of the paper.

F.8 Compute and Cost

A full run serves Qwen3.5-27B from self-hosted servers on H800 GPUs, and no model is trained. Cost per OOD gain in Table 26 is normalized to ScienceClaw under the protocol, which fixes the budgets of all methods (Appendix B.5). Per task, averaged over the IID and OOD instances, ScienceClaw makes 6.54 model API calls, Operator calls included (8.98 counting cache hits), and uses 3,155 uncached tokens, the newly processed tokens that normally count toward API usage (4,378 with cached tokens, the full logical context the model processes), 10.2 s of wall time, and 0.0113 GPU-hours on four GPUs.

G Complete Results

This appendix reports the per-discipline data behind the figures and tables of Section 6: relative margins, all eighteen variants on both splits, and the continual-evolution, transfer, reliability, and workflow measurements. Each discipline contributes 64 IID and 64 OOD instances to every number in this appendix.

G.1 Relative Margins per Discipline

Table 23 gives the relative margin of ScienceClaw over the frozen agent and over the strongest baseline in every discipline and split, computed from the scores shown in Table 1. ScienceClaw leads the frozen agent in all 23 disciplines on both splits, and its margin over the strongest baseline is positive in every discipline: it ranges from 1.82% to 3.73% on IID and from 3.05% to 6.21% on OOD. EvoMaster or EvoScientist is the strongest baseline in most rows, which matches the order of the baselines by how much of the program they evolve (Section 6).

G.2 Ablation Results on Both Splits

Table 22 summarizes every variant by its relative drop below the full system and by the share of the full system's gain over the frozen agent that it retains. Figures 7 and 8 show the task-native scores of all eighteen variants in each discipline, shaded within each discipline from the worst to the best variant, and Tables 28–30 list the same scores with the best and second best marked. The full system is the best variant in all 23 disciplines on both IID and OOD. The loss of every ablated variant is larger on OOD than on IID, in particular for the gate components: removing source replay, the scientific constraints, or the independent validator increases the drop 2.4 to 2.8 times from IID to OOD, and removing OOD transfer, which targets the OOD split, leaves IID nearly unchanged (a drop of 0.91%) while raising the OOD drop to 13.09%.

G.3 Continual Evolution, Transfer, and Reliability

Definitions. The OOD macro score after round r is MacroSR of the frozen snapshot A_r on $\mathcal{D}_{\text{OOD}}$ (Eq. (4)), and the round-0 entry is the shared starting program. The gain is the round-7 score minus the round-0 score. Of the 161 candidates a method submits (7 rounds × 23 disciplines), *promoted* candidates pass the gate and *rejected* candidates do not, and the rate is the promoted share. Forward transfer is the OOD gain on a target family after evolving on a source family, and the diagonal is the within-family gain. For ScienceClaw, FWT is the mean of the 25 entries of Figure 6b. Forgetting is measured on $\mathcal{D}_{\text{rep}}$ against the snapshot immediately after the source task. The five families are Life/health (FoR30, 31, 32, 42), Physical/Earth (FoR34, 37, 41, 51), Engineering/computing (FoR33, 40, 46, 49), Social/behavioural (FoR35, 38, 39, 44, 52), and Humanities/law (FoR36, 43, 45, 47, 48, 50). Backward transfer (BWT) is the mean change of the score on $\mathcal{D}_{\text{rep}}$ after later rounds, and forgetting is the average and maximum decrease of that score. Negative transfer is the share of transfer measurements with a negative gain. Erroneous promotion is the rate of promotions that lower the retained score or violate a hard constraint, and Attain. is the continual-target attainment score.

Table 20: Reading guide to the equations of the method.

Equation	Reading	Realized by
Eq. (1)	The fixed model solves task t with program A_r and returns the final workflow, the output, and the trace	one episode of the solver; the three results are saved with the run
Eq. (5)	A workflow is a graph of Operators and tools whose connected ports must agree in type, shape, and unit	the workflow graph and its schema check at every edge
Eq. (6)	At each step the policy proposes one atomic edit and names the Skills and Operators that contributed	the policy call that returns a JSON edit
Eq. (7)	Applying an edit reruns only the affected nodes and records diagnostics	the executor with checkpoints
Eq. (8)	Evidence is regenerated from a reset environment and scored by the evaluator	the replay and the evaluator of the task
Eqs. (9) and (10)	A repair is the sequence of edits between the last failure and the first success, split into strategy edits and executable edits	the evolution engine
Eqs. (11) and (12)	A Skill candidate rewrites attributed instructions, and an Operator candidate packages a connected executable subgraph with its interface and conditions	the evolution engine and the boundary replay
Eq. (13)	A candidate must reproduce its source repair and be used where expected before it enters the pool	the source reproduction check
Eqs. (2) and (3)	Among the feasible candidates choose the one with the best validation score, and update only if it strictly beats the current program	the gate with its noise guard and budgets (Appendix B.5)

Table 21: Program-level self-evolution compared with genetic programming.

	Genetic programming	ScienceClaw
Unit of evolution	a population of whole programs	one lineage $A_r = (\mathcal{S}_r, \mathcal{O}_r)$ of Skills and Operators
Variation	random crossover and mutation	repairs reproduced by replay and abstracted into candidates
Source of change	the search operators	the execution traces of solved tasks
Selection	fitness on the training cases	strict gate on validation tasks, replay set, and hard constraints
Persistence	the best individual of the run	a versioned program with rollback
Parameters of the model	not applicable	fixed, $\Theta_r = \Theta_0$

Table 22: Relative drop below the full system (%, mean over 23 disciplines) and retained share of the gain. The drop is $(s_{\text{full}} - s)/s_{\text{full}}$ for higher-is-better and $(s - s_{\text{full}})/s_{\text{full}}$ for lower-is-better metrics. The retained share is $1 - \text{drop}/\text{drop}_{\text{Frozen}}$ of the mean of the two splits; negative values mean that the variant is worse than the frozen agent.

Variant	IID	OOD	Mean	Retained (%)
Update scope				
Workflow only	9.42	15.16	12.29	10.2
w/o Skill evolution	3.46	5.41	4.44	67.6
w/o Operator evolution	4.43	6.87	5.65	58.7
w/o Workflow evolution	2.95	5.82	4.39	67.9
Linkage and candidate source				
w/o linked evolution	3.99	7.01	5.50	59.8
RuleEvo	6.92	11.62	9.27	32.3
Qwen-only	6.01	9.39	7.70	43.8
Gate and memory				
w/o source replay	3.25	9.10	6.18	54.9
w/o scientific constraints	4.19	10.48	7.33	46.4
w/o independent validator	3.33	8.03	5.68	58.5
w/o cross-task memory	2.30	7.14	4.72	65.5
w/o IID selection	9.27	14.00	11.64	14.9
w/o OOD transfer	0.91	13.09	7.00	48.9
Execution				
Fixed workflow	8.15	12.67	10.41	23.9
Single operator	12.52	17.05	14.78	-8.1
Single turn	13.84	18.82	16.33	-19.3
Reference				
Frozen	11.69	15.68	13.68	0.0

Table 23: Relative margin of ScienceClaw (%) in each discipline and split. The margin over the frozen agent is $(s_{SC} - s_0)/s_0$ for higher-is-better metrics and $(s_0 - s_{SC})/s_0$ for lower-is-better metrics. The strongest baseline is the best of SkillOpt, TTE, AHE, EvoMaster, and EvoScientist in that discipline and split.

	IID			OOD		
Discipline (metric)	vs. Frozen	vs. strongest	Strongest	vs. Frozen	vs. strongest	Strongest
Higher is better (↑)						
Agricultural sci. (PQ+)	10.77	2.38	EvoMaster	17.13	3.48	EvoScientist
Biological sci. (Spearman)	15.00	2.33	EvoScientist	13.07	3.19	EvoMaster
Biomedical sci. (DSC)	9.47	2.01	EvoScientist	16.05	4.43	EvoMaster
Chemical sci. (ROC-AUC)	15.75	2.74	EvoMaster	20.50	4.04	EvoScientist
Creative arts (SDR (dB))	14.19	2.43	EvoMaster	18.06	3.20	EvoScientist
Education (10-mask acc.)	15.31	2.34	EvoMaster	14.27	5.28	EvoMaster
Engineering (DCASE score)	14.48	2.43	EvoMaster	12.99	5.01	EvoScientist
Health sci. (Clin. utility)	10.92	2.44	EvoMaster	14.07	4.96	EvoScientist
Indigenous (chrF++)	12.88	3.12	EvoMaster	16.45	6.21	EvoMaster
Computing sci. (pass@1)	14.29	1.82	EvoMaster	20.83	3.57	EvoMaster
Language & culture (LAS)	11.67	3.73	EvoMaster	16.04	3.66	EvoMaster
Law (mAP)	13.86	3.16	EvoMaster	19.82	5.42	EvoMaster
Mathematics (Oracle acc.)	12.96	3.39	EvoMaster	19.15	3.70	EvoMaster
Philosophy (F1)	16.45	2.72	EvoScientist	23.34	3.95	EvoMaster
Psychology (Micro acc.)	8.70	2.20	EvoScientist	17.31	3.83	EvoMaster
Lower is better (↓)						
Built env. (NRMSE %)	11.92	2.74	EvoMaster	15.71	3.56	EvoMaster
Commerce (MASE)	8.47	2.21	EvoMaster	14.69	5.10	EvoScientist
Earth sci. (RMSE (K))	7.96	2.87	EvoScientist	15.70	4.24	TTE
Economics (sMAPE (%))	11.05	2.60	EvoMaster	14.51	4.16	EvoMaster
Environmental sci. (CRPS)	11.60	1.91	EvoMaster	11.57	3.25	EvoMaster
History (cMER-micro)	10.17	3.05	AHE	15.22	3.84	EvoMaster
Human society (nRMSE)	11.68	2.58	EvoMaster	15.04	5.27	EvoScientist
Physical sci. (MAE)	12.20	2.15	EvoScientist	16.89	3.05	EvoMaster
Mean over 23 disciplines	12.25	2.58		16.45	4.19	
Minimum	7.96	1.82		11.57	3.05	
Maximum	16.45	3.73		23.34	6.21	

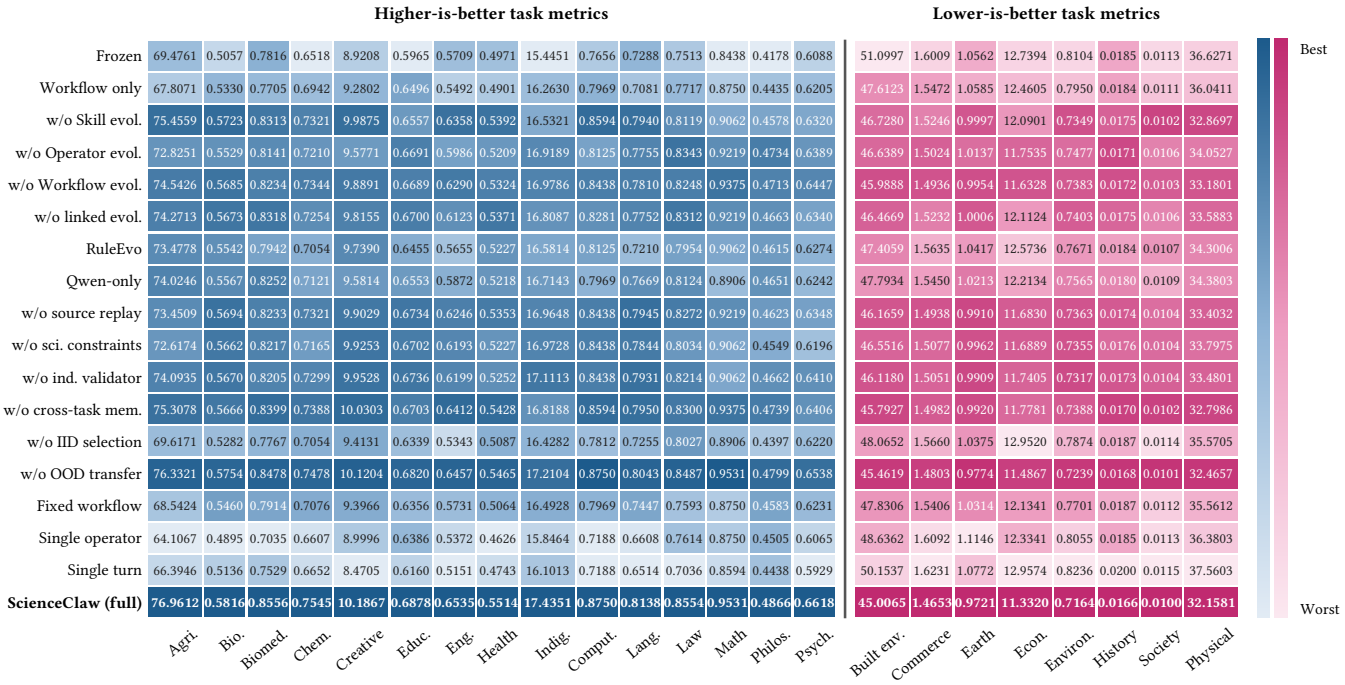


Figure 7: IID scores of all eighteen variants in each discipline. Shading runs from the worst (light) to the best (dark) variant within a discipline, and the full system is in bold.

Table 24: Continual evolution of the OOD macro score (%) over seven rounds with candidate counts over the 161 submitted candidates. The best entry in each column is in bold and the second best is underlined.

	OOD macro score (%) after round								Round-7 summary			
Method	0	1	2	3	4	5	6	7	Gain (pp)	Prom.	Rej.	Rate (%)
Frozen	77.72	77.72	77.72	77.72	77.72	77.72	77.72	77.72	0.00	0	0	–
RuleEvo	77.72	78.40	79.21	78.87	79.62	80.37	80.03	81.32	3.60	37	124	22.98
Qwen-only	77.72	79.42	80.16	79.89	81.25	81.93	81.59	83.22	5.50	49	112	30.43
EvoScientist	77.72	79.69	81.18	82.34	82.07	84.04	85.19	86.35	8.63	56	105	34.78
EvoMaster	77.72	80.03	81.59	**82.88**	82.54	84.85	86.07	87.50	9.78	63	98	39.13
ScienceClaw	77.72	**80.43**	**82.68**	82.47	**86.14**	**87.57**	**89.81**	**91.30**	**13.59**	**70**	91	**43.48**

Table 25: Workflow mechanics of the linked system and of the ablated executions. Planner rounds and distinct operators are per task, and the other columns are rates.

System	Plan. rounds	Dist. oper.	Oper. divers.	Feedb. repair (%)	Ckpt. recov. (%)	Clean replay (%)
Linked Full	4.84	4.24	**0.78**	**71**	**89**	**96**
Unlinked	4.63	3.96	0.71	62	82	89
Fixed Workflow	3.12	2.89	0.50	45	73	87
Single Operator	4.07	1.07	0.20	38	67	84
Single Turn	1.09	2.13	0.36	0	31	78

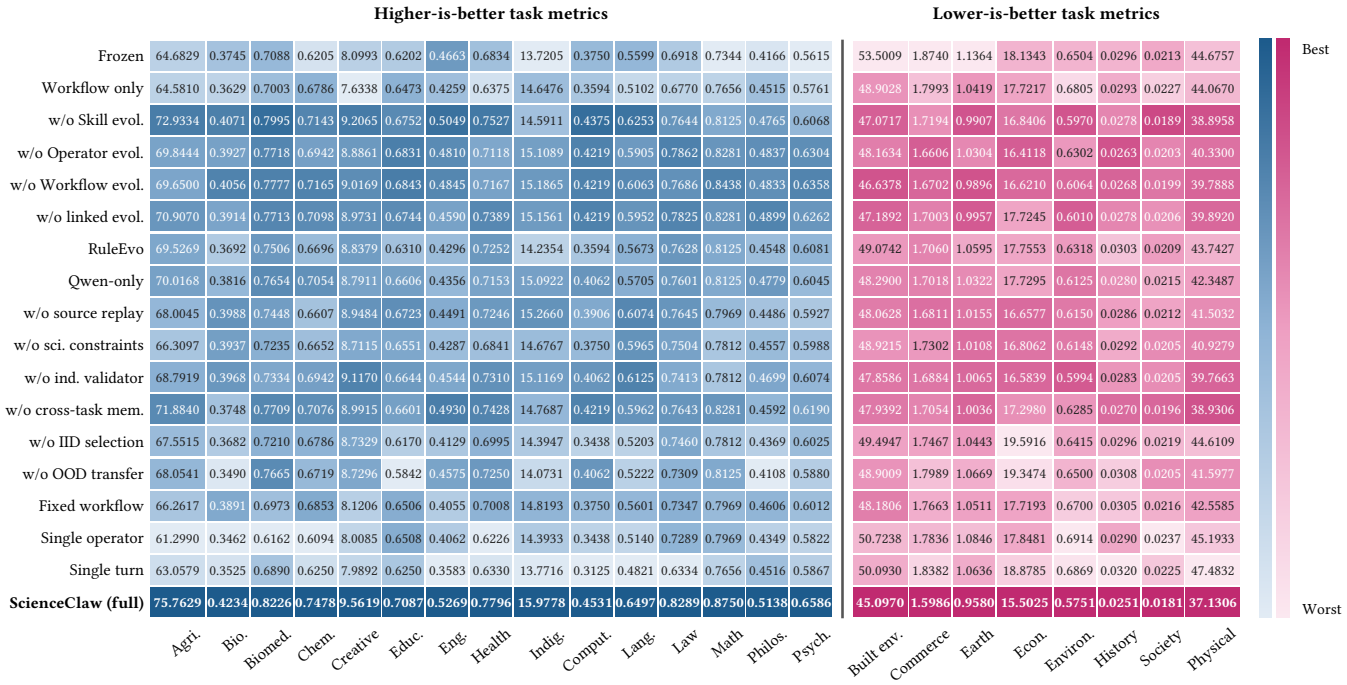


Figure 8: OOD scores of all eighteen variants in each discipline, shaded as in Figure 7.

Table 26: Reliability and efficiency. Cost per OOD gain is normalized to ScienceClaw (= 1) and is undefined for the frozen agent. The best entry in each column is in bold and the second best is underlined.

Method	Target attain. ↑	Hard-constr. pass (%) ↑	Erroneous prom. (%) ↓	Cost per OOD gain ↓	Planner time (%) ↓
Frozen	88.57	91.51	–	–	21.89
Qwen-only	92.99	93.89	7.41	1.62	23.65
SkillOpt	92.67	93.14	6.52	2.50	20.56
TTE	93.63	92.53	7.25	1.83	<u>19.38</u>
AHE	94.26	94.09	5.60	1.92	27.23
EvoMaster	<u>94.99</u>	<u>95.65</u>	<u>3.80</u>	<u>1.37</u>	25.16
EvoScientist	94.43	95.18	4.58	1.87	28.74
ScienceClaw	**96.42**	**98.23**	**1.23**	**1.00**	**18.40**

Table 27: Transfer, forgetting, and reliability of the evolving methods. FWT and BWT are forward and backward transfer (pp), Neg. the share of negative transfer (%), Forg. the average and maximum forgetting (pp), Attain. the target-attainment score, and Err. the erroneous-promotion rate (%). The best entry in each column is in bold and the second best is underlined.

Method	FWT/BWT ↑	Neg. ↓	Forg. ↓ avg/max	Attain. ↑	Err. ↓
SkillOpt	1.42/0.26	31.39	0.88/2.45	92.67	6.52
TTE	1.75/0.41	26.97	0.76/2.17	93.63	7.25
EvoScientist	2.74/1.23	17.32	0.42/1.23	94.43	4.58
EvoMaster	<u>3.19/1.53</u>	<u>14.33</u>	<u>0.33/1.05</u>	<u>94.99</u>	<u>3.80</u>
ScienceClaw	**4.20/3.66**	**11.89**	**0.13/0.51**	**96.42**	**1.23**

Table 28: Per-discipline ablation results (1/3): update-scope variants. Each row gives the task-native score of one discipline on one split. The best score in each row is in bold and the second best is underlined, taken over all 18 variants.

Discipline (metric)	Split	Frozen	Workflow only	w/o Skill evol.	w/o Oper. evol.	w/o Work. evol.	ScienceClaw (full)
Higher is better (↑)							
Agricultural sci. (PQ+)	IID	69.4761	67.8071	75.4559	72.8251	74.5426	**76.9612**
	OOD	64.6829	64.5810	72.9334	69.8444	69.6500	**75.7629**
Biological sci. (Spearman)	IID	0.5057	0.5330	0.5723	0.5529	0.5685	**0.5816**
	OOD	0.3745	0.3629	0.4071	0.3927	0.4056	**0.4234**
Biomedical sci. (DSC)	IID	0.7816	0.7705	0.8313	0.8141	0.8234	**0.8556**
	OOD	0.7088	0.7003	0.7995	0.7718	0.7777	**0.8226**
Chemical sci. (ROC-AUC)	IID	0.6518	0.6942	0.7321	0.7210	0.7344	**0.7545**
	OOD	0.6205	0.6786	0.7143	0.6942	0.7165	**0.7478**
Creative arts (SDR (dB))	IID	8.9208	9.2802	9.9875	9.5771	9.8891	**10.1867**
	OOD	8.0993	7.6338	9.2065	8.8861	9.0169	**9.5619**
Education (10-mask acc.)	IID	0.5965	0.6496	0.6557	0.6691	0.6689	**0.6878**
	OOD	0.6202	0.6473	0.6752	0.6831	0.6843	**0.7087**
Engineering (DCASE score)	IID	0.5709	0.5492	0.6358	0.5986	0.6290	**0.6535**
	OOD	0.4663	0.4259	0.5049	0.4810	0.4845	**0.5269**
Health sci. (Clin. utility)	IID	0.4971	0.4901	0.5392	0.5209	0.5324	**0.5514**
	OOD	0.6834	0.6375	0.7527	0.7118	0.7167	**0.7796**
Indigenous (chrF++)	IID	15.4451	16.2630	16.5321	16.9189	16.9786	**17.4351**
	OOD	13.7205	14.6476	14.5911	15.1089	15.1865	**15.9778**
Computing sci. (pass@1)	IID	0.7656	0.7969	0.8594	0.8125	0.8438	**0.8750**
	OOD	0.3750	0.3594	0.4375	0.4219	0.4219	**0.4531**
Language & culture (LAS)	IID	0.7288	0.7081	0.7940	0.7755	0.7810	**0.8138**
	OOD	0.5599	0.5102	0.6253	0.5905	0.6063	**0.6497**
Law (mAP)	IID	0.7513	0.7717	0.8119	0.8343	0.8248	**0.8554**
	OOD	0.6918	0.6770	0.7644	0.7862	0.7686	**0.8289**
Mathematics (Oracle acc.)	IID	0.8438	0.8750	0.9062	0.9219	0.9375	**0.9531**
	OOD	0.7344	0.7656	0.8125	0.8281	0.8438	**0.8750**
Philosophy (F1)	IID	0.4178	0.4435	0.4578	0.4734	0.4713	**0.4866**
	OOD	0.4166	0.4515	0.4765	0.4837	0.4833	**0.5138**
Psychology (Micro acc.)	IID	0.6088	0.6205	0.6320	0.6389	0.6447	**0.6618**
	OOD	0.5615	0.5761	0.6068	0.6304	0.6358	**0.6586**
Lower is better (↓)							
Built env. (NRMSE %)	IID	51.0997	47.6123	46.7280	46.6389	45.9888	**45.0065**
	OOD	53.5009	48.9028	47.0717	48.1634	46.6378	**45.0970**
Commerce (MASE)	IID	1.6009	1.5472	1.5246	1.5024	1.4936	**1.4653**
	OOD	1.8740	1.7993	1.7194	1.6606	1.6702	**1.5986**
Earth sci. (RMSE (K))	IID	1.0562	1.0585	0.9997	1.0137	0.9954	**0.9721**
	OOD	1.1364	1.0419	0.9907	1.0304	0.9896	**0.9580**
Economics (sMAPE (%))	IID	12.7394	12.4605	12.0901	11.7535	11.6328	**11.3320**
	OOD	18.1343	17.7217	16.8406	16.4118	16.6210	**15.5025**
Environmental sci. (CRPS)	IID	0.8104	0.7950	0.7349	0.7477	0.7383	**0.7164**
	OOD	0.6504	0.6805	0.5970	0.6302	0.6064	**0.5751**
History (cMER-micro)	IID	0.0185	0.0184	0.0175	0.0171	0.0172	**0.0166**
	OOD	0.0296	0.0293	0.0278	0.0263	0.0268	**0.0251**
Human society (nRMSE)	IID	0.0113	0.0111	0.0102	0.0106	0.0103	**0.0100**
	OOD	0.0213	0.0227	0.0189	0.0203	0.0199	**0.0181**
Physical sci. (MAE)	IID	36.6271	36.0411	32.8697	34.0527	33.1801	**32.1581**
	OOD	44.6757	44.0670	38.8958	40.3300	39.7888	**37.1306**

Table 29: Per-discipline ablation results (2/3): linkage, candidate-source, and gate variants. Each row gives the task-native score of one discipline on one split. The best score in each row is in bold and the second best is underlined, taken over all 18 variants.

Discipline (metric)	Split	Frozen	w/o link. evol.	RuleEvo	Qwen-only	w/o src. replay	w/o sci. constr.	w/o ind. valid.	ScienceClaw (full)
Higher is better (↑)									
Agricultural sci. (PQ+)	IID	69.4761	74.2713	73.4778	74.0246	73.4509	72.6174	74.0935	**76.9612**
	OOD	64.6829	70.9070	69.5269	70.0168	68.0045	66.3097	68.7919	**75.7629**
Biological sci. (Spearman)	IID	0.5057	0.5673	0.5542	0.5567	0.5694	0.5662	0.5670	**0.5816**
	OOD	0.3745	0.3914	0.3692	0.3816	0.3988	0.3937	0.3968	**0.4234**
Biomedical sci. (DSC)	IID	0.7816	0.8318	0.7942	0.8252	0.8233	0.8217	0.8205	**0.8556**
	OOD	0.7088	0.7713	0.7506	0.7654	0.7448	0.7235	0.7334	**0.8226**
Chemical sci. (ROC-AUC)	IID	0.6518	0.7254	0.7054	0.7121	0.7321	0.7165	0.7299	**0.7545**
	OOD	0.6205	0.7098	0.6696	0.7054	0.6607	0.6652	0.6942	**0.7478**
Creative arts (SDR (dB))	IID	8.9208	9.8155	9.7390	9.5814	9.9029	9.9253	9.9528	**10.1867**
	OOD	8.0993	8.9731	8.8379	8.7911	8.9484	8.7115	9.1170	**9.5619**
Education (10-mask acc.)	IID	0.5965	0.6700	0.6455	0.6553	0.6734	0.6702	0.6736	**0.6878**
	OOD	0.6202	0.6744	0.6310	0.6606	0.6723	0.6551	0.6644	**0.7087**
Engineering (DCASE score)	IID	0.5709	0.6123	0.5655	0.5872	0.6246	0.6193	0.6199	**0.6535**
	OOD	0.4663	0.4590	0.4296	0.4356	0.4491	0.4287	0.4544	**0.5269**
Health sci. (Clin. utility)	IID	0.4971	0.5371	0.5227	0.5218	0.5353	0.5227	0.5252	**0.5514**
	OOD	0.6834	0.7389	0.7252	0.7153	0.7246	0.6841	0.7310	**0.7796**
Indigenous (chrF++)	IID	15.4451	16.8087	16.5814	16.7143	16.9648	16.9728	17.1113	**17.4351**
	OOD	13.7205	15.1561	14.2354	15.0922	<u>15.2660</u>	14.6767	15.1169	**15.9778**
Computing sci. (pass@1)	IID	0.7656	0.8281	0.8125	0.7969	0.8438	0.8438	0.8438	**0.8750**
	OOD	0.3750	0.4219	0.3594	0.4062	0.3906	0.3750	0.4062	**0.4531**
Language & culture (LAS)	IID	0.7288	0.7752	0.7210	0.7669	0.7945	0.7844	0.7931	**0.8138**
	OOD	0.5599	0.5952	0.5673	0.5705	0.6074	0.5965	0.6125	**0.6497**
Law (mAP)	IID	0.7513	0.8312	0.7954	0.8124	0.8272	0.8034	0.8214	**0.8554**
	OOD	0.6918	0.7825	0.7628	0.7601	0.7645	0.7504	0.7413	**0.8289**
Mathematics (Oracle acc.)	IID	0.8438	0.9219	0.9062	0.8906	0.9219	0.9062	0.9062	**0.9531**
	OOD	0.7344	0.8281	0.8125	0.8125	0.7969	0.7812	0.7812	**0.8750**
Philosophy (F1)	IID	0.4178	0.4663	0.4615	0.4651	0.4623	0.4549	0.4662	**0.4866**
	OOD	0.4166	<u>0.4899</u>	0.4548	0.4779	0.4486	0.4557	0.4699	**0.5138**
Psychology (Micro acc.)	IID	0.6088	0.6340	0.6274	0.6242	0.6348	0.6196	0.6410	**0.6618**
	OOD	0.5615	0.6262	0.6081	0.6045	0.5927	0.5988	0.6074	**0.6586**
Lower is better (↓)									
Built env. (NRMSE %)	IID	51.0997	46.4669	47.4059	47.7934	46.1659	46.5516	46.1180	**45.0065**
	OOD	53.5009	47.1892	49.0742	48.2900	48.0628	48.9215	47.8586	**45.0970**
Commerce (MASE)	IID	1.6009	1.5232	1.5635	1.5450	1.4938	1.5077	1.5051	**1.4653**
	OOD	1.8740	1.7003	1.7060	1.7018	1.6811	1.7302	1.6884	**1.5986**
Earth sci. (RMSE (K))	IID	1.0562	1.0006	1.0417	1.0213	0.9910	0.9962	0.9909	**0.9721**
	OOD	1.1364	0.9957	1.0595	1.0322	1.0155	1.0108	1.0065	**0.9580**
Economics (sMAPE (%))	IID	12.7394	12.1124	12.5736	12.2134	11.6830	11.6889	11.7405	**11.3320**
	OOD	18.1343	17.7245	17.7553	17.7295	16.6577	16.8062	16.5839	**15.5025**
Environmental sci. (CRPS)	IID	0.8104	0.7403	0.7671	0.7565	0.7363	0.7355	0.7317	**0.7164**
	OOD	0.6504	0.6010	0.6318	0.6125	0.6150	0.6148	0.5994	**0.5751**
History (cMER-micro)	IID	0.0185	0.0175	0.0184	0.0180	0.0174	0.0176	0.0173	**0.0166**
	OOD	0.0296	0.0278	0.0303	0.0280	0.0286	0.0292	0.0283	**0.0251**
Human society (nRMSE)	IID	0.0113	0.0106	0.0107	0.0109	0.0104	0.0104	0.0104	**0.0100**
	OOD	0.0213	0.0206	0.0209	0.0215	0.0212	0.0205	0.0205	**0.0181**
Physical sci. (MAE)	IID	36.6271	33.5883	34.3006	34.3803	33.4032	33.7975	33.4801	**32.1581**
	OOD	44.6757	39.8920	43.7427	42.3487	41.5032	40.9279	39.7663	**37.1306**

Table 30: Per-discipline ablation results (3/3): memory, selection, transfer, and execution variants. Each row gives the task-native score of one discipline on one split. The best score in each row is in bold and the second best is underlined, taken over all 18 variants.

Discipline (metric)	Split	Frozen	w/o cross-task mem.	w/o IID select.	w/o OOD transfer	Fixed workflow	Single operator	Single turn	ScienceClaw (full)
Higher is better (↑)									
Agricultural sci. (PQ+)	IID	69.4761	75.3078	69.6171	76.3321	68.5424	64.1067	66.3946	**76.9612**
	OOD	64.6829	71.8840	67.5515	68.0541	66.2617	61.2990	63.0579	**75.7629**
Biological sci. (Spearman)	IID	0.5057	0.5666	0.5282	0.5754	0.5460	0.4895	0.5136	**0.5816**
	OOD	0.3745	0.3748	0.3682	0.3490	0.3891	0.3462	0.3525	**0.4234**
Biomedical sci. (DSC)	IID	0.7816	0.8399	0.7767	0.8478	0.7914	0.7035	0.7529	**0.8556**
	OOD	0.7088	0.7709	0.7210	0.7665	0.6973	0.6162	0.6890	**0.8226**
Chemical sci. (ROC-AUC)	IID	0.6518	0.7388	0.7054	0.7478	0.7076	0.6607	0.6652	**0.7545**
	OOD	0.6205	0.7076	0.6786	0.6719	0.6853	0.6094	0.6250	**0.7478**
Creative arts (SDR (dB))	IID	8.9208	10.0303	9.4131	10.1204	9.3966	8.9996	8.4705	**10.1867**
	OOD	8.0993	8.9915	8.7329	8.7296	8.1206	8.0085	7.9892	**9.5619**
Education (10-mask acc.)	IID	0.5965	0.6703	0.6339	0.6820	0.6356	0.6386	0.6160	**0.6878**
	OOD	0.6202	0.6601	0.6170	0.5842	0.6506	0.6508	0.6250	**0.7087**
Engineering (DCASE score)	IID	0.5709	0.6412	0.5343	0.6457	0.5731	0.5372	0.5151	**0.6535**
	OOD	0.4663	0.4930	0.4129	0.4575	0.4055	0.4062	0.3583	**0.5269**
Health sci. (Clin. utility)	IID	0.4971	0.5428	0.5087	0.5465	0.5064	0.4626	0.4743	**0.5514**
	OOD	0.6834	0.7428	0.6995	0.7250	0.7008	0.6226	0.6330	**0.7796**
Indigenous (chrF++)	IID	15.4451	16.8188	16.4282	17.2104	16.4928	15.8464	16.1013	**17.4351**
	OOD	13.7205	14.7687	14.3947	14.0731	14.8193	14.3933	13.7716	**15.9778**
Computing sci. (pass@1)	IID	0.7656	0.8594	0.7812	**0.8750**	0.7969	0.7188	0.7188	**0.8750**
	OOD	0.3750	0.4219	0.3438	0.4062	0.3750	0.3438	0.3125	**0.4531**
Language & culture (LAS)	IID	0.7288	0.7950	0.7255	0.8043	0.7447	0.6608	0.6514	**0.8138**
	OOD	0.5599	0.5962	0.5203	0.5222	0.5601	0.5140	0.4821	**0.6497**
Law (mAP)	IID	0.7513	0.8300	0.8027	0.8487	0.7593	0.7614	0.7036	**0.8554**
	OOD	0.6918	0.7643	0.7460	0.7309	0.7347	0.7289	0.6334	**0.8289**
Mathematics (Oracle acc.)	IID	0.8438	0.9375	0.8906	**0.9531**	0.8750	0.8750	0.8594	**0.9531**
	OOD	0.7344	0.8281	0.7812	0.8125	0.7969	0.7969	0.7656	**0.8750**
Philosophy (F1)	IID	0.4178	0.4739	0.4397	0.4799	0.4583	0.4505	0.4438	**0.4866**
	OOD	0.4166	0.4592	0.4369	0.4108	0.4606	0.4349	0.4516	**0.5138**
Psychology (Micro acc.)	IID	0.6088	0.6406	0.6220	0.6538	0.6231	0.6065	0.5929	**0.6618**
	OOD	0.5615	0.6190	0.6025	0.5880	0.6012	0.5822	0.5867	**0.6586**
Lower is better (↓)									
Built env. (NRMSE %)	IID	51.0997	45.7927	48.0652	45.4619	47.8306	48.6362	50.1537	**45.0065**
	OOD	53.5009	47.9392	49.4947	48.9009	48.1806	50.7238	50.0930	**45.0970**
Commerce (MASE)	IID	1.6009	1.4982	1.5660	1.4803	1.5406	1.6092	1.6231	**1.4653**
	OOD	1.8740	1.7054	1.7467	1.7989	1.7663	1.7836	1.8382	**1.5986**
Earth sci. (RMSE (K))	IID	1.0562	0.9920	1.0375	0.9774	1.0314	1.1146	1.0772	**0.9721**
	OOD	1.1364	1.0036	1.0443	1.0669	1.0511	1.0846	1.0636	**0.9580**
Economics (sMAPE (%))	IID	12.7394	11.7781	12.9520	11.4867	12.1341	12.3341	12.9574	**11.3320**
	OOD	18.1343	17.2980	19.5916	19.3474	17.7193	17.8481	18.8785	**15.5025**
Environmental sci. (CRPS)	IID	0.8104	0.7388	0.7874	0.7239	0.7701	0.8055	0.8236	**0.7164**
	OOD	0.6504	0.6285	0.6415	0.6500	0.6700	0.6914	0.6869	**0.5751**
History (cMER-micro)	IID	0.0185	0.0170	0.0187	0.0168	0.0187	0.0185	0.0200	**0.0166**
	OOD	0.0296	0.0270	0.0296	0.0308	0.0305	0.0290	0.0320	**0.0251**
Human society (nRMSE)	IID	0.0113	0.0102	0.0114	0.0101	0.0112	0.0113	0.0115	**0.0100**
	OOD	0.0213	0.0196	0.0219	0.0205	0.0216	0.0237	0.0225	**0.0181**
Physical sci. (MAE)	IID	36.6271	32.7986	35.5705	32.4657	35.5612	36.3803	37.5603	**32.1581**
	OOD	44.6757	38.9306	44.6109	41.5977	42.5585	45.1933	47.4832	**37.1306**