EVOLUTION FINE-TUNING: Learning to Discover Across 371 Optimization Tasks

Young-Jun Lee[1] **Seungone Kim**[2] **Minki Kang**[3]
Alistair Cheong[2] **Zerui Chen**[4]
Seungho Han[5] **Taehee Jung**[6] **Dongyeop Kang**[1]

University of Minnesota[1] Carnegie Mellon University[2] KAIST[3]
University of Cambridge[4] Hanyang University[5] Amazon[6]

Abstract

Would experience designing faster GPU kernels also help close in on a long-standing open mathematical conjecture? Large Language Models (LLMs) integrated into evolutionary search have recently produced state-of-the-art solutions on optimization tasks, including open mathematical conjectures, GPU kernel design, scientific law discovery, and combinatorial puzzles. To achieve this, prior work applied search scaffolds to one target task at a time, so every new problem is approached from scratch and the experience accumulated during search is discarded once the model finishes its attempt. This leaves the capability of iteratively evolving a solution (*e.g.*, knowing which part to mutate and how, deciding when to backtrack) entirely in the scaffold rather than in the model itself. Whether the model itself could acquire this capability and reuse it across different tasks has been largely unexamined. To address this, we introduce **EVOLUTION FINE-TUNING (EFT)**, a mid-training paradigm that teaches LLMs to evolve solutions across tasks by converting evolutionary search trajectories into supervision. We construct $\mathcal{F}$inch Collection, a 156K-trajectory dataset spanning 10 domains and 371 optimization tasks, and fine-tune open-source LLMs from 2B to 9B parameters. Empirically, EFT confers cross-task generalization: across 22 held-out tasks, our models surpass their base counterparts by 10.22% on average. Furthermore, when paired with test-time RL, our model match state-of-the-art performance on two circle-packing tasks and outperforms its base-model counterparts on the Erdős minimum-overlap problem. EFT thus serves as a "practice phase" for general-purpose discovery agents that doesn't solve new problems from scratch.

1 Introduction

Some of the most consequential problems in mathematics, algorithm engineering, and the natural sciences are optimization tasks: problems for which a candidate solution can be scored against an objective, but for which the optimal solution is not directly computable [1]. Concrete examples include open mathematical conjectures such as the Erdős minimum-overlap problem [2, 3], the design of high-performance GPU kernels [4], and the discovery of new scientific laws from data [5].

Recently, large language models (LLMs) combined with evolutionary search methods have begun to produce state-of-the-art solutions across such tasks: at each iteration, the LLM proposes new candidate solutions, a scaffold scores them and updates a population of high-scoring candidates, and the loop continues until a strong solution emerges [6, 7, 8]. Two methodological branches dominate this line of work: (1) *Test-time search* methods use a fixed, typically proprietary LLM as the mutation operator and rely on the scaffold's parent selection and prompting logic to drive

AI Agents for Discovery in the Wild (AID-Wild), Workshop at ACM CAIS 2026.

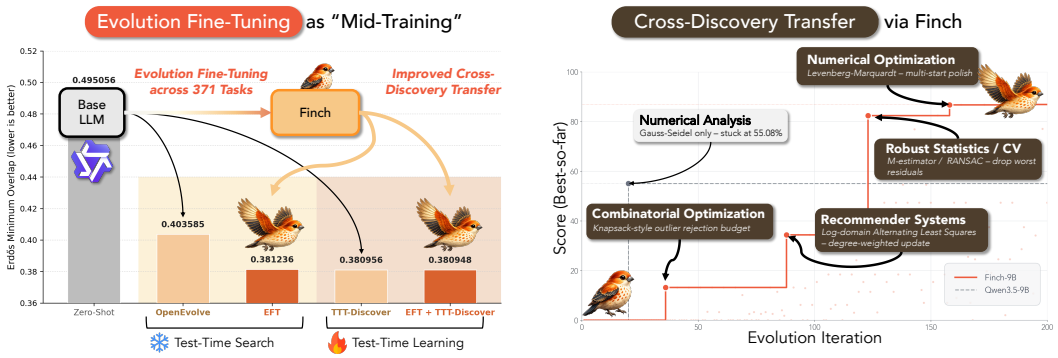


Figure 1: **(Left) EVOLUTION FINE-TUNING (EFT)** serves as "mid-training", boosting $\mathcal{F}$inch's discovery capability on the Erdős minimum overlap problem under both test-time search and learning. **(Right)** On NP-hard competitive programming (CALICO, UC Berkeley contest), EFT enables cross-discovery transfer: $\mathcal{F}$inch solves problems by combining strategies acquired from diverse domains, such as combinatorial optimization, recommender systems, robust statistics/computer vision, and numerical optimization. In contrast, the base model without EFT relies on a single, repetitive strategy.

improvement [9, 10, 11, 12]. (2) *Test-time learning* methods additionally update the LLM's weights during the search process, allowing the model to specialize to the target task as it explores [13, 14].

Despite empirical successes, both branches share a common limitation: the discovery capability (*i.e.*, the skill of iteratively improving a solution, knowing what to mutate, what to keep, and when to backtrack) is constructed during each search rather than internalized into the model itself. Specifically,

1. Test-time search methods often rely on proprietary, frontier-scale LLMs as their mutation operator, because the scaffold demands consistent, high-quality proposals at every iteration. In our experiments, we observe that open-source models smaller than 9B parameters fail to follow evolutionary trajectories within such scaffolds and yield substantially weaker performance (Figure 1, left).
2. Test-time learning methods alleviate this by allowing smaller LLMs to adapt their weights based on their own search experience, and have produced new best-known solutions on several mathematical problems [13, 14]. However, these updates are tailored to a single search loop and a single task; the strategies the model discovers are not consolidated into reusable capability, so the model cannot compose strategies from prior tasks when tackling a new one (Figure 1, right).
3. More fundamentally, in neither branch does the model itself acquire the evolving capability. Test-time search-based methods do not update the model at all, leaving the capability in the search procedure by design. Test-time learning-based methods update the model through test-time RL, but the updates serve to find a solution within a single search loop rather than to internalize the discovery capability itself, and they are discarded once the task is solved.

One promising direction to address these limitations is to have the LLM itself *meta-learn* the discovery capability (*i.e.*, learning how to evolve solutions across optimization tasks). The core challenge in doing so is that optimization tasks are NP-hard and lack ground truth optimal solutions, making the standard supervised learning recipe of collecting (problem, answer) pairs unavailable. To circumvent this, we propose **EVOLUTION FINE-TUNING (EFT)**, a mid-training paradigm that treats the *trajectories* of search runs as the supervision signal, thereby internalizing the discovery capability into the model itself. We construct $\mathcal{F}$inch Collection, a large-scale dataset of 156K such trajectories collected using a widely used search scaffold, i.e., OpenEvolve [15], and a recent large model,

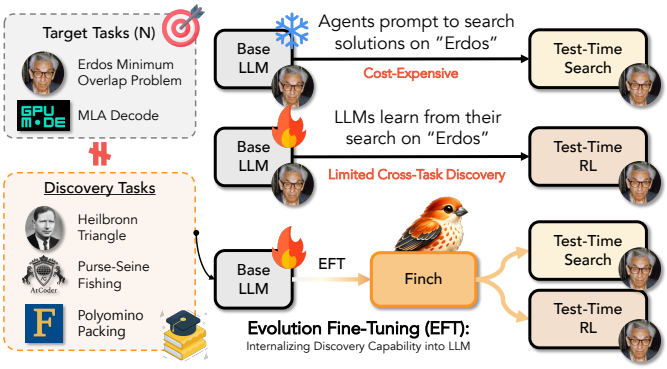


Figure 2: A Concept of Evolution Fine-Tuning (EFT)

Table 1: Comparison of evolutionary methods. Scaffold indicates whether the method provides search or learning support for evolution. Train and Test denote scaffolding at training or test time, respectively. OS denotes open sourcing.

Method	Main Contribution	Scaffold		Training				OS
		Search	Learn	Train	Test	Paradigm	#Tasks	
AlphaEvolve [7]	Search	✓	✗	✗	✗	–	–	✗
OpenEvolve [15]	Search	✓	✗	✗	✗	–	–	✓
ShinkaEvolve [8]	Search	✓	✗	✗	✗	–	–	✓
GEPA [18]	Search	✓	✗	✗	✗	–	–	✓
PAC-Evolve [10]	Search	✓	✗	✗	✗	–	–	✗
AdaEvolve [11]	Search	✓	✗	✗	✗	–	–	✓
EvoX [12]	Search	✓	✗	✗	✗	–	–	✓
DGM [19]	Search	✓	✗	✗	✗	–	–	✓
HyperAgent [20]	Search	✓	✗	✗	✗	–	–	✓
CORAL [21]	Search	✓	✗	✗	✗	–	–	✓
ThetaEvolve [13]	Learning	✗	✓	✗	✓	RL	1	✓
TTT-Discover [14]	Learning	✗	✓	✗	✓	RL	1	✓
EFT (Ours)	**Model, data**	✓	✓	✓	✓	**SFT, RL**	**371**	✓

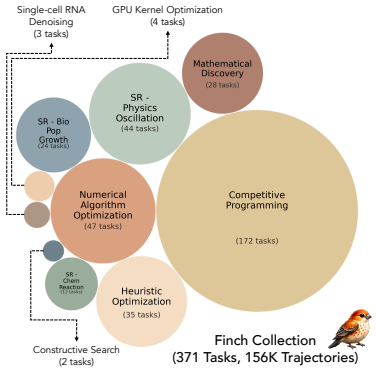


Figure 3: An overview of the optimization task groups (a total of 371 tasks) in $\mathcal{F}$inch Collection, where bubble size indicates the number of tasks in each group.

Qwen3.5-397B-A17B [16], across 10 domains and 371 tasks. Using $\mathcal{F}$inch Collection, we fine-tune open-source models from 2B to 9B, and obtain a new model family, $\mathcal{F}$inch-{2, 4, 8, 9}B.

To demonstrate the cross-task generalization conferred by $\mathcal{F}$inch Collection, we first employ $\mathcal{F}$inch as mutation operators in test-time search scaffolds. $\mathcal{F}$inch outperforms its base counterparts on 22 held-out tasks and achieves performance comparable to best-known solutions previously obtained by a proprietary model, despite using a much smaller open-source backbone. Notably, as shown in Figure 1 (right), when solving a competitive programming task, we observe that while base LLM tends to apply only in-domain strategy (i.e., Gauss-Seidel uniform weight), $\mathcal{F}$inch transfers strategies across domains (e.g., applying log-domain alternating least squares from recommender system, Levenberg-Marquardt from numerical optimization to solve a competitive programming problem), suggesting that **EFT** gives rise to emergent behaviors in discovery tasks. Furthermore, scaling the number of training tasks in $\mathcal{F}$inch Collection from 15 to 355 improves $\mathcal{F}$inch's held-out performance by **14.1%** on average across held-out tasks. Finally, to assess whether $\mathcal{F}$inch can also learn from its own search experience, we apply test-time learning to both $\mathcal{F}$inch (i.e., w/ **EFT**) and base LLMs (i.e., w/o **EFT**) on three tasks. We find that $\mathcal{F}$inch achieves state-of-the-art performance on two circle packing tasks and outperforms its base-model counterpart on the Erdős Overlap Minimum Problem.

2 Preliminaries

Optimization setup. We consider an optimization task $\tau \in \mathcal{T}$ with an initial candidate solution x_0 and an iteration budget T. The candidate set generated during search is denoted by $\mathcal{X} = \{x_0, \ldots, x_T\}$, where x_0 may be a program [7], math construction [17], or prompt [15], depending on the task. At iteration t, an evolutionary scaffold $\mathcal{S}$ uses a mutation operator $\mathcal{M}_\theta$, typically an LLM, to produce a new candidate from the parent solution and search history: $x_t = \mathcal{S}(x_{t-1}, I, \mathcal{H}_{t-1}; \mathcal{M}_\theta)$, where $\mathcal{H}_{t-1}$ contains selected prior candidates and feedback. An evaluator $\mathcal{E}$ assigns a score and auxiliary artifacts such as logs or natural-language feedback. The goal is

$$x^\star = \arg\operatorname*{opt}_{x \in \mathcal{X}} \mathcal{E}(x), \qquad \text{opt} \in \{\max, \min\}, \tag{1}$$

where the optimization direction is determined by task: $\max$ for accuracy and $\min$ for c5 bound in Erdos problem.

Discovery. Following prior work [14], we call $x^\star$ a *discovery* if it improves upon the previous best-known solution x_{sota} within budget T, i.e., $\mathcal{E}(x^\star) > \mathcal{E}(x_{\text{sota}})$ for maximization tasks, with the inequality reversed for minimization tasks.

Evolutionary Search Scaffold. To discover novel solutions τ, existing LLM-based evolutionary methods use either search-based or learning-based scaffolds. Search-based scaffolds keep θ fixed

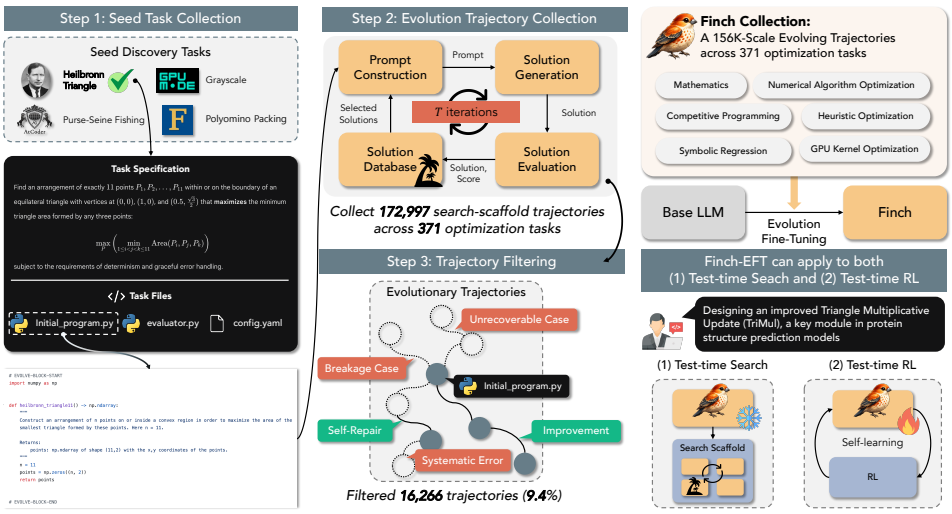


Figure 4: Overview of the $\mathcal{F}$inch Collection construction pipeline, consisting of (1) seed optimization task collection, (2) trajectory collection via an evolutionary search scaffold (i.e., OpenEvolve [15]), and (3) trajectory filtering for unrecoverable cases, breakage cases, and candidate solutions that incur systematic errors (e.g., timeout errors).

and rely on external search mechanisms, while learning-based scaffolds update θ at test time. Both approaches are fundamentally built upon four core modular components: (i) construct a prompt from the parent solution, task instruction, history, and feedback; (ii) generate a candidate using $\mathcal{M}_\theta$ through either `diff-based edit` or `full rewrite`; (iii) evaluate the candidate with $\mathcal{E}$; and (iv) store eligible candidates in a population database $\mathcal{D}$. The updated database $\mathcal{D}_t$ then informs the next iteration until the compute budget or target score is reached. We summarize existing evolutionary methods in Tab 1.

3 Evolution Fine-Tuning

Evolutionary scaffolds can discover strong solutions at test time, but the discovery procedure is typically external to the model. We introduce **Evolution Fine-Tuning (EFT)**, a mid-training procedure that transfers this test-time discovery behavior into smaller open-source LLMs. EFT converts evolutionary search trajectories into supervised training examples, so that the model learns to act as a stronger mutation operator before deployment. As summarized in Table 1, EFT is orthogonal to the choice of test-time scaffold: the resulting model can be used inside search-based scaffolds with frozen weights, or further adapted by learning-based scaffolds such as test-time RL.

3.1 $\mathcal{F}$inch Collection Construction

Overview. Figure 4 illustrates the construction of $\mathcal{F}$inch Collection. Each task consists of a task specification, an initial candidate solution, an evaluator, and configuration files. We run an evolutionary scaffold over these task files to produce a sequence of parent-to-child solution transitions. Each transition records the prompt, parent solution, generated candidate, evaluator output, score change, and execution artifacts. We then remove trajectories whose feedback is unreliable or whose transitions would provide misleading supervision. The final collection contains approximately 156K evolutionary trajectories across 371 optimization tasks.

Step 1: Seed Optimization Task Collection. Training data for optimization is difficult to synthesize because many target problems are NL-hard, lack known global optima, and require expert-designed evaluators (see Table 8 in Appendix D). Instead of generating artificial tasks, we source seed tasks from existing optimization benchmarks whose objectives are executable and externally validated. We select tasks according to three criteria: (i) the task should require nontrivial search, (ii) should not

reduce to matching a known ground-truth answer, and (iii) should provide a deterministic evaluator that assigns a continuous or comparable score to candidate solutions.

In total, as shown in Figure 3, we collect 371 seed tasks from 10 benchmarks, spanning from mathematical discovery, competitive programming, heuristic optimization, numerical algorithm optimization, symbolic regression, GPU kernel optimization, constructive search, and biological denoising benchmarks. These include AlphaEvolve's mathematical discovery problems [7], FrontierCS [22], ALE-Bench [17], AlgoTune [23], GPU Mode [24], LLM-SRBench [5], Function Minimization and K-Module tasks from OpenEvolve [15], scRNA-seq denoising [25, 14], and variants of Erdős problems [26, 27]. The full task list is provided in Appendix G.

Step 2: Evolutionary Trajectory Collection. For each seed task τ, we run an evolutionary scaffold $\mathcal{S}$ for a fixed budget of iterations. At iteration t, the scaffold selects a parent solution x_{t-1} and constructs a prompt from the task instruction I, the parent, the search history $\mathcal{H}_{t-1}$, and evaluator artifacts. For each task, N_τ trajectories are generated where $N_\tau < T$, where trajectories flagged as errors are discarded. A teacher mutation operator $\mathcal{M}_\theta$ then generates a candidate solution x_t, which is executed by the task evaluator $\mathcal{E}$. The resulting trajectory stores $(I, x_{t-1}, \mathcal{H}_{t-1}, x_t, \mathcal{E}(x_t), \mathcal{F}_t)$, where $\mathcal{F}_t$ includes execution logs, error traces, and evaluator feedback.

We instantiate $\mathcal{M}$ with OpenEvolve [15] and use Qwen3.5-397B-A17B [16] as the teacher mutation operator. To expose the student model to both local refinement and global exploration, we collect trajectories under two mutation strategies: `diff-based edit` and `full rewrite`. The `diff-based edit` strategy asks the model to edit an existing solution and therefore emphasizes exploitation, while `full rewrite` asks the model to rewrite the solution and therefore encourages broader exploration. For diversity, we run the scaffold multiple times per task with stochastic decoding. Unless otherwise specified, we use temperature 0.7, top-$p = 0.95$, and a maximum generation length of 30K tokens. In total, we obtain 172,997 trajectories.

Step 3: Trajectory Filtering. Raw evolutionary traces contain failures that should not be imitated. From 172,997 raw trajectories, we retain 156,731 (90.6%) by applying the following criteria:

- First, we remove **systematic errors**, where the evaluator output is unreliable or the score delta cannot be computed, removing 6,321 trajectories (3.7%) in total. Examples include missing parent scores (57.5%), timeout errors (14.2%), syntax-check failures, import failures, and evaluator crashes. The complete list of systematic errors is provided in Table 9.
- Second, we remove **unrecoverable & breakage cases**: Unrecoverable cases are both parent and child solutions are erroneous (294; 0.2%), and breakage cases are an error-free parent yields an erroneous child (1,281; 0.8%). Both constitute hard negatives, destabilizing the training signal.
- Lastly, we remove **excessively long inputs**, to keep training stable and tractable. We dsicard responses longer than 16,384 tokens and examples whose total serialized input-output length exceeds 32,768 tokens, removing 8,370 (5.0%).

After filtering, $\mathcal{F}$inch Collection contains approximately 156K trajectories across 371 tasks.

Each retained trajectory is converted into a supervised fine-tuning instance. The input consists of the same information available to the mutation operator inside the scaffold: the task instruction, parent solution, selected history, previous scores, and evaluator artifacts. The target output is the teacher-generated candidate solution. This format directly trains the model to map an evolutionary state to a plausible next mutation: $(I, x_{t-1}, \mathcal{H}_{t-1}, \mathcal{F}_{t-1}) \mapsto x_t$.

When task scores are available, we classify each trajectory based on its improvement outcome (Δ), i.e., $\Delta = \mathcal{E}(x_t) - \mathcal{E}(x_{t-1})$, into three categories: $\Delta > 0$ (Imp), $\Delta = 0$ (NC), and $\Delta < 0$ (Reg). As shown in Figure 5, this yields 61,802 (39.4%) Imp, 30,130 (19.2%) NC, and 64,799 (41.3%) Reg trajectories. In this work, we primarily use Imp trajectories for evolution fine-tuning to prevent $\mathcal{F}$inch from imitating non-improving behaviors due to Reg, as shown in Table 2. Furthermore, to enable $\mathcal{F}$inch to learn how to evolve parent programs effectively

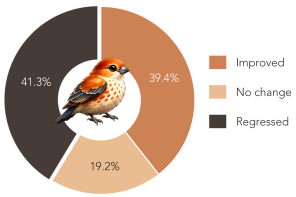


Figure 5: Distribution of trajectory improvement in $\mathcal{F}$inch Collection.

Table 2: Effect of improvement type on EFT.

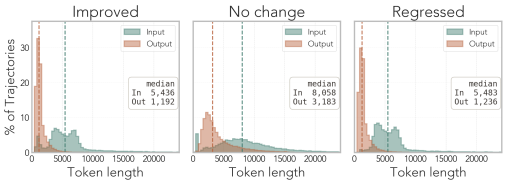
Model	Erdos (↓)	AC1 (↓)	AC2 (↑)
Qwen3-8B	0.403585	1.5177	0.8980
+ Imp	**0.381236**	**1.5154**	**0.9001**
+ Imp + NC + Reg	0.492126	1.5186	0.8789

Task Group	# Tasks	# Traj.
Competitive Programming	172	43,469
Numerical Algorithm Optimization	47	4,835
SR Physics Oscillation	44	51,210
Heuristic Optimization	35	5,924
Mathematical Discovery	28	6,508
SR Bio Pop Growth	24	28,034
SR Chem Reaction	12	13,975
GPU Kernel Optimization	4	1,088
Single cell RNA Denoising	3	503
Constructive Search	2	1,185
Total	**371**	**156,731**

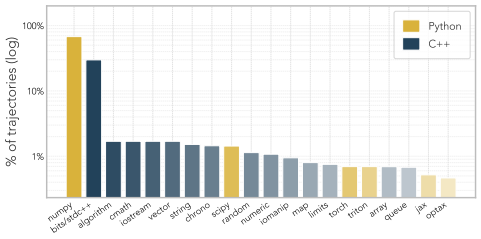
(a) Tasks and trajectories by task group

Language	Ratio	Mutation	Ratio
Python	68.5%	`diff-based edit`	50.3%
C++	31.5%	`full rewrite`	49.7%

(b) Languages and mutation strategies



(c) Trajectory length distribution by improvement type



(d) Top 20% packages used in initial program

Figure 6: Dataset composition and trajectory characteristics. (a) Number of tasks and trajectories for each task group. (b) Programming language and mutation strategy proportions across trajectories. (c) Distribution of trajectory lengths by improvement type (Imp, NC, and Reg). (d) Top-20% packages used in initial programs.

within the solution space, we leverage both Imp and Reg trajectories through a preference learning algorithm (i.e.,, KTO [28]) [1]. Consequently, our $\mathcal{F}$inch Collection provides a comprehensive set of improvement trajectories, serving as a valuable resource for internalizing discovery capabilities into LLMs.

3.2 Analysis of $\mathcal{F}$inch Collection

Overall, $\mathcal{F}$inch Collection contains 156,731 (~156K) evolutionary search trajectories, covering 371 tasks across 10 task groups. As shown in Figure 6 (a), Competitive Programming occupies the largest share with 172 individual tasks and is derived from FrontierCS [22], followed by Numerical Algorithm Optimization, which builds upon ALE-Bench [17]. In Figure 6 (b), 68.5% of trajectories in $\mathcal{F}$inch Collection are based on Python, with a considerable portion in C++ (31.5%). Regarding mutation strategy, $\mathcal{F}$inch Collection exhibits a well-balanced distribution between `diff-based edit` (50.3%) and `full rewrite` (49.7%). Together, these statistics suggest that models trained on $\mathcal{F}$inch Collection acquire optimization capabilities across both Python and C++, as well as proficiency in both `diff-based edit` and `full rewrite` strategies. Moreover, Figure 6 (c) shows that the input length (avg. 6,865 tokens) is substantially longer than the output length (avg. 8,902 tokens, 1.3×), indicating that $\mathcal{F}$inch learns to optimize parent programs by selectively leveraging useful feedback signals from a large context of prior programs. In addition, within NC, the median output length is 2.9× longer, suggesting that Qwen3.5-397B-A17B tends to engage in more extensive reasoning even when no improvement is achieved. Meanwhile, the output length in Reg trajectories is comparable to that in Imp ones, implying that regressions arise not from insufficient reasoning, but from misguided exploration within the solution space. Moreover, in Figure 6 (d), $\mathcal{F}$inch Collection most frequently utilizes `numpy`, followed by `bits/stdc++`, suggesting that the dataset spans both numerical computing and competitive programming workloads, thereby exposing models to diverse optimization patterns across multiple domains.

3.3 $\mathcal{F}$inch: Evolution Fine-Tuned Language Model

We introduce a new family of evolution fine-tuned LLMs, $\mathcal{F}$inch, trained on $\mathcal{F}$inch Collection across 355 tasks, excluding 16 held-out tasks. As base models, we use the Qwen3.5 [16] series at

[1] To maximize the contrastive signal that guides $\mathcal{F}$inch toward self-judging which solutions are promising and which fall short, we restrict KTO training to Imp and Reg in this work. Nevertheless, we believe NC is also a valuable resource for internalizing discovery capability, as suggested in Table 7.

Table 3: Performance comparison of $\mathcal{F}$inch to base models across Mathematical Discovery, Algorithm Engineering, and System Performance benchmarks. † Scores reported in [21]. ‡ Scores reported in [12]. Δ denotes the relative improvement (%) of $\mathcal{F}$inch over the corresponding same-size base model, sign-adjusted so that positive values always indicate improvement regardless of metric direction. **Avg. Gain** is the arithmetic mean of available Δ values, excluding **ahc058** where the base score is near zero and yields a disproportionately large ratio.

	Mathematical Discovery					Algorithm Engineering		System Performance				
Model	**Erdos(↓)**	**AC1(↓)**	**AC2(↑)**	**CP(n=26)(↑)**	**Hadamard(↑)**	**ahc039(↑)**	**ahc058(↑)**	**EPLB(↑)**	**PRISM(↑)**	**LLM-SQL(↑)**	**Transaction(↑)**	**Avg.(↑)**
Best Human	0.380927	1.5097	0.9015	2.634000	0.935673	566,997	847,674,723	0.1265	21.89	0.6920	2724.80	–
Initial Program	0.495056	1.5186	0.8558	0.959764	0.143275	534,850	0	0.1265	21.89	0.6856	2824.86	–
OpenEvolve + Proprietary Models												
Claude-Opus-4.6 †	0.381880	–	–	2.629300	–	–	–	0.1270	26.26	0.7160	3774.00	–
Gemini-3-Pro ‡	–	–	–	2.541400	–	–	–	0.1272	26.24	0.7258	4273.50	–
GPT-5 ‡	–	–	–	2.541400	–	–	–	0.1272	26.23	0.7155	4237.30	–
OpenEvolve + Open-source Models												
Qwen3.5-2B	0.381737	1.5186	0.8646	1.253056	0.478009	546,078	0	0.1265	21.89	0.6856	2832.86	–
$\mathcal{F}$inch-2B	0.381346	1.5184	0.8920	1.535134	0.400476	545,256	329,359,253	0.1269	22.26	0.6860	2949.85	–
Δ	+0.10%	+0.01%	+3.17%	+22.51%	-16.22%	-0.15%	–	+0.32%	+1.69%	+0.06%	+4.13%	**+1.56%**
Qwen3.5-4B	0.416924	1.5186	0.8802	1.680787	0.384332	542,077	0	0.1266	21.89	0.6856	2732.24	–
$\mathcal{F}$inch-4B	0.386460	1.5173	0.8933	1.806808	0.146199	551,844	331,466,883	0.1267	22.87	0.6857	4761.90	–
Δ	+7.31%	+0.09%	+1.49%	+7.50%	-61.96%	+1.80%	–	+0.08%	+4.48%	+0.01%	+74.30%	**+3.40%**
Qwen3-8B	0.403585	1.5177	0.8980	1.797576	0.452330	557,081	0	0.1269	23.81	0.6858	3174.60	–
$\mathcal{F}$inch-8B	0.381236	1.5154	0.9001	1.822617	0.501743	557,168	135,184,684	0.1270	24.70	0.7341	3257.33	–
Δ	+5.54%	+0.15%	+0.23%	+1.39%	+10.92%	+0.02%	–	+0.08%	+3.74%	+7.04%	+2.61%	**+3.17%**
Qwen3.5-9B	0.385512	1.5186	0.8801	1.172702	0.397184	553,582	134,486,700	0.1269	22.36	0.6858	3584.23	–
$\mathcal{F}$inch-9B	0.381100	1.5141	0.9122	1.936000	0.480585	553,759	525,286,896	0.1265	23.93	0.7024	3636.36	–
Δ	+1.14%	+0.30%	+3.65%	+65.09%	+21.00%	+0.03%	+290.59%	-0.32%	+7.02%	+2.42%	+1.45%	**+10.24%**

three model sizes—2B, 4B, and 9B—as well as the Qwen3-8B [29] for $\mathcal{F}$inch-8B. As shown in Figure 3, the trajectory distribution is highly imbalanced, with Symbolic Regression (SR) trajectories constituting the majority, which could adversely affect training. To mitigate this, we use only one trajectory per task during training[2]. We fine-tune the base models using only `Imp` trajectories from $\mathcal{F}$inch Collection via full SFT, resulting in a total of 30,445 training trajectories. For validation, we use 900 trajectories uniformly sampled across tasks. For training, we employ the LLaMA-Factory [30] framework. All models are trained for one epoch with a global batch size of 128 and a learning rate of 1e-5. All experiments are conducted on eight NVIDIA H200 140GB GPUs.

4 Experiments

4.1 Experimental Setup

Evaluation Tasks. To evaluate the **cross-task discovery generalization** of $\mathcal{F}$inch, we use a diverse set of optimization tasks that are, to the extent possible, disjoint from those used to train $\mathcal{F}$inch. These tasks are drawn from benchmarks widely adopted in prior works [11, 12, 31, 14, 32]. In total, we evaluate $\mathcal{F}$inch across five domains and 22 tasks: **(1) Mathematical Discovery**, including the Erdős Overlap Minimum Problem (Erdos), First Autocorrelation Inequality (AC1), Second Autocorrelation Inequality (AC2), Circle Packing in a Unit Square with $n = 26$ (CP), and Hadamard Maximum Determinant (Hadamard); **(2) Algorithm Engineering**, including two tasks, ahc039 and ahc048; **(3) System Performance**, including four tasks, EPLB, PRISM, LLM-SQL, and Transaction; **(4) Competitive Programming**, comprising six tasks, each scored on a scale from 0 to 100. These include Problem 263 (P263) from CALICO[3], UC Berkeley's official programming contest featuring open-ended optimization problems, and five newly added FrontierCS v1.1 problems, Problem 301–305 (P301–305); and **(5) Algorithmic Heuristics**, including five tasks, Convolve2D Full Fill (Convolve2D), PolynomialReal (Polynomial), Positive Semidefinite Cone Projection (PSD), 2D Affine Transform (Affine Transform), and FFT Convolution (FFT Conv.). Although some tasks are drawn from the same benchmark suite, we treat them as independent evaluation tasks because discovery tasks often differ substantially in their objectives, search spaces, and solution forms. For evaluation, following the definition of discovery, we report the maximum task-specific score achieved within T iterations. Each optimization task uses its own task-specific metric, such as the `c5_bound`

[2]Recall from Step 2 of dataset construction that three trajectories are collected per task.

[3]https://frontier-cs.org/blog/calico/

Table 4: Competitive Programming performance of $\mathcal{F}$inch compared to base models across six optimization tasks.

Model	P263 (↑)	P301 (↑)	P302 (↑)	P303 (↑)	P304 (↑)	P305 (↑)	Avg. (↑)
Qwen3.5-2B	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$\mathcal{F}$inch-2B	0.38	1.63	0.12	3.16	0.00	31.43	**6.12**
Qwen3.5-4B	8.15	20.99	27.07	0.00	0.00	30.89	14.52
$\mathcal{F}$inch-4B	27.44	68.17	24.41	31.79	0.00	40.03	**31.97**
Qwen3-8B	23.72	44.67	36.78	10.84	0.00	29.23	24.21
$\mathcal{F}$inch-8B	38.12	39.41	36.68	10.84	0.00	22.34	**24.56**
Qwen3.5-9B	55.09	27.59	35.63	35.54	5.81	35.14	32.46
$\mathcal{F}$inch-9B	86.10	58.78	36.68	34.02	22.11	38.38	**46.01**

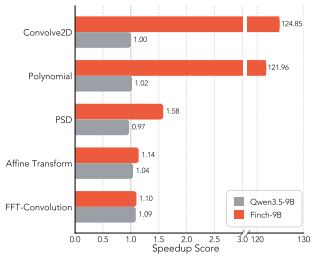


Figure 7: Algorithmic Heuristics performance of $\mathcal{F}$inch compared to Qwen3.5-9B.

Table 5: Effect of Offline RL (CP: avg. competitive programming score).

Model	Erdos (↓)	AC1 (↓)	AC2 (↑)	CP (↑)
Best Human	**0.380927**	1.5097	0.9015	–
Qwen3.5-4B	0.416924	1.5186	0.8802	14.52
$\mathcal{F}$inch-4B	0.386460	1.5173	0.8933	31.97
$\mathcal{F}$inch-4B + KTO	0.381809	1.5151	0.9121	**36.30**
Qwen3-8B	0.403585	1.5177	0.8980	24.21
$\mathcal{F}$inch-8B	**0.381236**	1.5154	0.9001	24.56
$\mathcal{F}$inch-8B + KTO	0.381596	**1.5089**	**0.9146**	**37.30**

Table 6: Performance of $\mathcal{F}$inch combined with online (test-time) RL scaffolds across math optimization tasks.

Scaffold	Model	Erdos (↓)	CP (n=26) (↑)	CP (n=32) (↑)
ThetaEvolve	R1-Qwen3-8B	-	**2.635983**	-
TTT-Discover	GPT-OSS-120B	0.380876	-	-
	Qwen3-8B	**0.380932**	**2.635983**	2.939572
nanodiscover	Qwen3-8B	0.380956	**2.635983**	**2.939573**
	$\mathcal{F}$inch-8B	**0.380948**	**2.635983**	**2.939573**

for the Erdős overlap problem. Detailed descriptions of these task-specific metrics are provided in Appendix G.

Baselines. We evaluate $\mathcal{F}$inch's capability as a mutation operator when combined with a test-time search scaffold; throughout this work, we use OpenEvolve [15] as the default scaffold unless otherwise noted. Specifically, we measure the relative improvement of $\mathcal{F}$inch over the initial program score and compare it against two baselines: (1) the base model with a search scaffold; and (2) the base model with a learning scaffold (e.g.,, TTT-Discover [14]). However, since running the original TTT-Discover is prohibitively expensive—each task requires up to 50 epochs to reproduce, costing approximately 500 USD on average—we instead adopt nanodiscover[4], an open-source reproduction of TTT-Discover that does not depend on the Tinker API. A detailed description of nanodiscover is provided in Appendix F.

Inference Details. For the search-based scaffold, following prior work [12], we set $T = 100$ with a parallel evaluation size of 1, temperature 0.7, top-p 0.95, and a maximum of 30K tokens. For the remaining scaffold-specific hyperparameters, such as the island size, we adopt the default settings specified for each task. For the learning-based scaffold, we follow the same configurations as the original TTT-Discover, which is presented in Appendix F.

4.2 Impact of Evolution Fine-Tuning

EFT improves cross-task discovery generalization across all model scales. Table 3 shows that $\mathcal{F}$inch, evolution fine-tuned on $\mathcal{F}$inch Collection, achieves substantial relative performance gains across 22 tasks, with the largest improvements observed on ahc058 (+290.59%) and Transaction (+74.30%). In Table 4 and Figure 7, we observe that EFT is also effective on complex optimization-intensive tasks, including NP-hard competitive programming and algorithmic heuristics. Moreover, larger models benefit more from $\mathcal{F}$inch Collection: $\mathcal{F}$inch-9B obtains a larger relative gain (+10.31%) than $\mathcal{F}$inch-4B (+3.40%). Our findings further demonstrate that EFT enables smaller models to match or even exceed the performance of non-EFT models twice their size; for instance, $\mathcal{F}$inch-4B achieves 0.386460 on Erdos, comparable to Qwen3-8B's 0.403585. Taken together, these results suggest that $\mathcal{F}$inch Collection enables LLMs to internalize discovery capabilities, thereby exhibiting cross-task discovery generalization.

[4] https://github.com/cheongalc/nanodiscover/

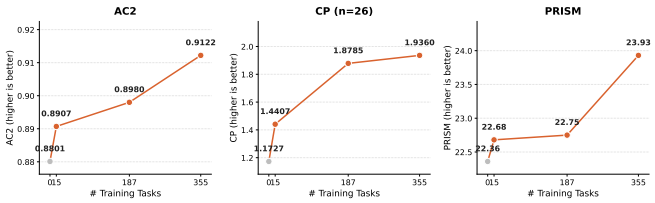


Figure 8: Scaling trends with increasing numbers of training tasks in $\mathcal{F}$inch Collection, evaluated on AC2, CP ($n = 26$), and PRISM.

Table 7: Effect of improvement type on EFT (CP: avg. competitive programming score)

Model	Erdos (↓)	AC2 (↑)	CP (↑)
Best Human	0.380927	0.9015	-
Qwen3-8B	0.403585	0.8980	24.21
+ Imp	0.381236	0.9001	24.56
+ Imp + NC	0.381261	0.9052	19.56
Qwen3.5-9B	0.385512	0.8801	32.46
+ Imp	**0.381100**	0.9122	46.01
+ Imp + NC	0.399486	**0.9148**	**50.12**

Teaching $\mathcal{F}$inch to distinguish good from bad solutions further enhances cross-task discovery generalization. After EFT, we further train $\mathcal{F}$inch using preference learning (KTO [28]) on Imp and Reg jointly, in order to examine whether discovery capability can be further internalized by enabling $\mathcal{F}$inch to self-judge which solutions are good and which are not. As shown in Table 5, offline RL consistently improves $\mathcal{F}$inch's discovery capability; notably, $\mathcal{F}$inch-8B with KTO surpasses the best human score on both the AC1 and AC2 tasks. These results suggest that (1) $\mathcal{F}$inch Collection provides a useful and complementary training signal beyond supervised fine-tuning, and (2) internalizing the ability to discriminate good from bad solutions is a viable path toward instilling discovery skills directly into the model's parameters, rather than relying solely on test-time search.

EFT serves as mid-training for test-time RL. We apply test-time RL to $\mathcal{F}$inch using nanodiscover. Table 6 shows that $\mathcal{F}$inch achieves the best performance on two circle-packing tests and improves performance on Erdős (+3.2%). These results suggest that EFT can serve as a form of mid-training that strengthens test-time RL. However, compared with the original TTT-Discover using GPT-OSS-120B, $\mathcal{F}$inch still achieves lower performance, indicating that it remains difficult for smaller models (e.g., 8B-scale) to discover push-frontier solutions.

5 Analysis

Scaling the number of tasks improves cross-task discovery transfer. As shown in Figure 8, performance on AC2, CP ($n = 26$), and PRISM exhibits a clear positive scaling trend as the number of training tasks in $\mathcal{F}$inch Collection increases. These results indicate that $\mathcal{F}$inch Collection provides a scalable training signal: increasing the amount of $\mathcal{F}$inch Collection consistently improves discovery performance. This suggests that the gains from EFT are not merely task-specific, but can further grow with larger task collections, more trajectories, and increased model capacity.

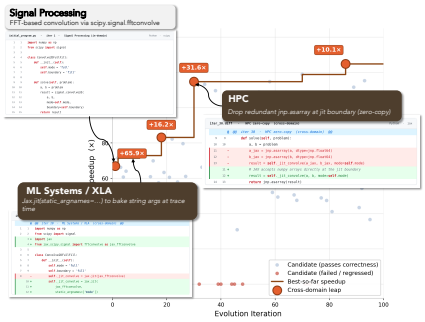


Figure 9: Case Study on Convolve2D.

$\mathcal{F}$inch can transfer discovery pattern by adopting different domain knowledge. Figure 9 presents a case study on the Convolve2D task. In this example, $\mathcal{F}$inch-9B modifies the implementation from `scipy` to `jax` to improve computational efficiency. We hypothesize that this behavior emerges because $\mathcal{F}$inch Collection contains a substantial number of trajectories involving the `jax` library, particularly from tasks such as uncertainty inequalities and matrix multiplication, as shown in Figure 6 (d). These results suggest that $\mathcal{F}$inch Collection enables LLMs to internalize transferable discovery patterns across domains.

6 Conclusion

In this work, we show that evolution fine-tuning (EFT) — fine-tuning LLMs on a large collection of evolutionary trajectories spanning 371 tasks — improves the model's discovery capability on 22 held-out tasks, demonstrating cross-task discovery generalization. Furthermore, we show that $\mathcal{F}$inch exhibits a synergistic effect when combined with test-time RL.

Acknowledgement

This research was supported by the "Advanced GPU Utilization Support Program" funded by the Government of the Republic of Korea (Ministry of Science and ICT). We thank the SkyDiscover team for their valuable feedback on the dataset construction process, the use of the SkyDiscover framework, and the overall direction of this research. In particular, we would like to thank Shu Liu, Shubham Agarwal, and Mert Cemri for their insightful comments and discussions. We also thank the OpenEvolve team, especially Ritik Vijayvergiya and Asankhaya Sharma, for their helpful feedback on the use of the OpenEvolve framework and for their thoughtful comments on this work. Finally, we sincerely thank Tahee Jung from Amazon for providing extensive and valuable feedback throughout this project.

References

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

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

[3] Ethan Patrick White. A new bound for erdős' minimum overlap problem. *Acta Arithmetica*, 208:235–255, 2023.

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

[5] Parshin Shojaee, Ngoc-Hieu Nguyen, Kazem Meidani, Amir Barati Farimani, Khoa D Doan, and Chandan K Reddy. Llm-srbench: A new benchmark for scientific equation discovery with large language models. *arXiv preprint arXiv:2504.10415*, 2025.

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

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

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

[9] Henrique Assumpção, Diego Ferreira, Leandro Campos, and Fabricio Murai. Codeevolve: An open source evolutionary coding agent for algorithm discovery and optimization. *arXiv preprint arXiv:2510.14150*, 2025.

[10] Minghao Yan, Bo Peng, Benjamin Coleman, Ziqi Chen, Zhouhang Xie, Shuo Chen, Zhankui He, Noveen Sachdeva, Isabella Ye, Weili Wang, et al. Pacevolve: Enabling long-horizon progress-aware consistent evolution. *arXiv preprint arXiv:2601.10657*, 2026.

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

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

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

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

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

[16] Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026.

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

[18] 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. Gepa: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.

[19] Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin godel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025.

[20] Jenny Zhang, Bingchen Zhao, Wannan Yang, Jakob Foerster, Jeff Clune, Minqi Jiang, Sam Devlin, and Tatiana Shavrina. Hyperagents. *arXiv preprint arXiv:2603.19461*, 2026.

[21] Ao Qu, Han Zheng, Zijian Zhou, Yihao Yan, Yihong Tang, Shao Yong Ong, Fenglu Hong, Kaichen Zhou, Chonghe Jiang, Minwei Kong, et al. Coral: Towards autonomous multi-agent evolution for open-ended discovery. *arXiv preprint arXiv:2604.01658*, 2026.

[22] Qiuyang Mang, Wenhao Chai, Zhifei Li, Huanzhi Mao, Shang Zhou, Alexander Du, Hanchen Li, Shu Liu, Edwin Chen, Yichuan Wang, et al. Frontiercs: Evolving challenges for evolving intelligence. *arXiv preprint arXiv:2512.15699*, 2025.

[23] Ori Press, Brandon Amos, Haoyu Zhao, Yikai Wu, Samuel K Ainsworth, Dominik Krupke, Patrick Kidger, Touqir Sajed, Bartolomeo Stellato, Jisun Park, et al. Algotune: Can language models speed up general-purpose numerical programs? *arXiv preprint arXiv:2507.15887*, 2025.

[24] GPU MODE. GPU MODE. https://www.gpumode.com/home, 2026. Accessed: 2026-05-03.

[25] Malte D Luecken, Scott Gigante, Daniel B Burkhardt, Robrecht Cannoodt, Daniel C Strobl, Nikolay S Markov, Luke Zappia, Giovanni Palla, Wesley Lewis, Daniel Dimitrov, et al. Defining and benchmarking open problems in single-cell analysis. *Nature Biotechnology*, 43(7):1035–1040, 2025.

[26] Tony Feng, Trieu Trinh, Garrett Bingham, Jiwon Kang, Shengtong Zhang, Sang-hyun Kim, Kevin Barreto, Carl Schildkraut, Junehyuk Jung, Jaehyeon Seo, et al. Semi-autonomous mathematics discovery with gemini: A case study on the erd\h {o} s problems. *arXiv preprint arXiv:2601.22401*, 2026.

[27] Thomas Bloom. Erdos problems. https://www.erdosproblems.com/, 2026. Accessed: 2026-05-03.

[28] Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

[29] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[30] Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 3: system demonstrations)*, pages 400–410, 2024.

[31] Shu Liu, Mert Cemri, Shubham Agarwal, Alexander Krentsel, Ashwin Naren, Qiuyang Mang, Zhifei Li, Akshat Gupta, Monishwaran Maheswaran, Audrey Cheng, Melissa Pan, Ethan Boneh, Kannan Ramchandran, Koushik Sen, Alexandros G. Dimakis, Matei Zaharia, and Ion Stoica. Skydiscover: A flexible framework for ai-driven scientific and algorithmic discovery, 2026.

[32] Haotian Ye, Haowei Lin, Jingyi Tang, Yizhen Luo, Caiyin Yang, Chang Su, Rahul Thapa, Rui Yang, Ruihua Liu, Zeyu Li, et al. Evaluation-driven scaling for scientific discovery. *arXiv preprint arXiv:2604.19341*, 2026.

[33] Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The flan collection: Designing data and methods for effective instruction tuning. In *International conference on machine learning*, pages 22631–22648. PMLR, 2023.

[34] Gang Liao, Hongsen Qin, Ying Wang, Alicia Golden, Michael Kuchnik, Yavuz Yetim, Jia Jiunn Ang, Chunli Fu, Yihan He, Samuel Hsia, et al. Kernelevolve: Scaling agentic kernel coding for heterogeneous ai accelerators at meta. *arXiv preprint arXiv:2512.23236*, 2025.

[35] He Du, Qiming Ge, Jiakai Hu, Aijun Yang, Zheng Cai, Zixian Huang, Sheng Yuan, Qinxiu Cheng, Xinchen Xie, Yicheng Chen, et al. Kernel-smith: A unified recipe for evolutionary kernel optimization. *arXiv preprint arXiv:2603.28342*, 2026.

[36] Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026.

[37] Manish Shetty, Naman Jain, Jinjian Liu, Vijay Kethanaboyina, Koushik Sen, and Ion Stoica. Gso: Challenging software optimization tasks for evaluating swe-agents. *arXiv preprint arXiv:2505.23671*, 2025.

[38] AutoLab Team. Autolab: Can models begin to participate in the loops that drive scientific and engineering progress?, 2026.

[39] Haowei Lin, Haotian Ye, Wenzheng Feng, Quzhe Huang, Yujun Li, Hubert Lim, Zhengrui Li, Xiangyu Wang, Jianzhu Ma, Yitao Liang, et al. Can language models discover scaling laws? *arXiv preprint arXiv:2507.21184*, 2025.

[40] Charles Darwin. On the origin of species. In *Scientific Methodology in Nineteenth Century Britain*, pages 133–181. Routledge, 2025.

[41] Peter R Grant and B Rosemary Grant. Unpredictable evolution in a 30-year study of darwin's finches. *science*, 296(5568):707–711, 2002.

[42] Lev Vygotsky et al. *Interaction between learning and development*. Linköpings universitet Linköping, Sweden, 2011.

A Limitations

Mixed Test-Time Search Scaffolds. In this work, we use only the OpenEvolve scaffold for both collecting trajectories and evaluating $\mathcal{F}$inch. A concern is that a model trained on OpenEvolve-style trajectories may not generalize well when combined with stronger scaffolds, such as EvoX [12]. To mitigate this issue in future work, collecting trajectories from mixed scaffolds will be necessary as a form of template input variation [33] to improve cross-scaffold generalization.

Extending Test-Time RL Experiments to Diverse and Realistic Tasks. In this work, we demonstrate that $\mathcal{F}$inch exhibits a positive synergistic effect with test-time RL only on mathematical tasks. However, many practical tasks exist in the real world, such as kernel engineering. It is therefore important to verify whether our model also exhibits a positive synergistic effect with test-time RL on these tasks in order to establish the broader effectiveness of our approach.

B Broader Impacts

EFT democratizes LLM-driven discovery by transferring optimization capabilities from expensive proprietary models to small open-weight models, reducing search costs, enabling fully local discovery pipelines, and providing $\mathcal{F}$inch Collection as a reusable resource for future research. At the same time, these capabilities may be misused for harmful optimization objectives, reward hacking, or over-reliance on automatically generated discoveries without sufficient verification. To mitigate these risks, we train only on public scientific and engineering benchmarks, preserve the upstream safety properties of base models, and recommend human oversight, scoring-function auditing, and trajectory logging for downstream deployment.

C Related Work

LLM-driven Evolutionary Search Scaffolds. Recent work shows that LLM-driven evolutionary search scaffolds can produce novel solutions across diverse optimization tasks. These methods can be broadly categorized into *search-based* and *learning-based* approaches. Search-based scaffolds primarily differ in how they archive candidate solutions and select parents for mutation. For instance, AlphaEvolve [7] employs MAP-Elites with island-based populations to balance exploration and exploitation, while OpenEvolve adopts a similar MAP-Elites framework with periodic migration. CodeEvolve [9] integrates island-based genetic algorithms with inspiration-based crossover and meta-prompting, and ShinkaEvolve [8] replaces fixed quality–diversity grids with a sample-efficient regime combining weighted sampling, novelty rejection, and bandit-based LLM selection. Other methods explore alternative selection and adaptation strategies: GEPA operates along Pareto frontiers; PACEvolve [10] and AdaEvolve [11] emphasize long-horizon, progress-aware, and adaptive proposal mechanisms; and EvoX [12] and SkyDiscover [31] study meta-evolution and unified discovery frameworks. Application-specific extensions include KernelEvolve [34] and Kernel-Smith [35] for GPU kernel optimization, while the Darwin Gödel Machine [19], HyperAgents [20], and CORAL [21] investigate open-ended self-improvement and multi-agent evolution. Meta-Harness [36] further shifts the optimization target from candidate solutions to the *harness* itself. In contrast, learning-based approaches such as ThetaEvolve and TTT-Discover demonstrate that combining evolutionary search with test-time learning enables LLMs to internalize discovery capabilities.

Benchmarks for Optimization and Scientific Discovery. Progress in LLM-driven discovery has been driven by benchmarks that emphasize long-horizon, open-ended problem solving under strong baselines. KernelBench [4] and GSO [37] evaluate iterative optimization of GPU kernels and programs, while AlgoTune [23] measures performance gains in numerical routines. ALE-Bench [17] focuses on long-horizon algorithm engineering tasks derived from programming contests. FrontierCS [22] and AutoLab [38] introduce *living* benchmarks that evolve alongside frontier models and support end-to-end scientific–engineering loops. For scientific-law discovery, LLM-SRBench [5] evaluates whether models can recover symbolic equations from data, while [39] examine the rediscovery of empirical scaling laws.

D Further Details and Discussion

D.1 Why is the model named $\mathcal{F}$inch?

Darwin's finches, despite belonging to the same species, have evolved differently in response to the diverse ecological environments of the Galápagos Islands [40, 41]. This observation illustrates their remarkable ability to adapt to a wide range of environmental conditions and to thrive within them.

Inspired by this phenomenon, we name our model $\mathcal{F}$inch. Analogous to Darwin's finches, our model is designed to adapt across diverse environments—here corresponding to different optimization tasks—and to effectively operate within them. In particular, it reflects the model's ability to flexibly adapt to various tasks (e.g., mathematics, kernel engineering, biology denoising) and to discover better solutions within each task setting.

D.2 Definition of Scaffolds

The term *scaffold* is borrowed from developmental psychology. Vygotsky [42] characterizes scaffolding as external support that elevates a learner's *assisted performance* beyond their unaided reach, and further distinguishes between progress driven by *externally supplied* structure and progress driven by the learner's *internalization* of past experience. Following this distinction, we taxonomize evolutionary search scaffolds into **supervised** and **unsupervised** scaffolds.

Definition: Supervised Scaffold. A supervised scaffold (i.e.,, search-based) [15, 8, 11, 12, 31] keeps the mutation operator's parameters θ fixed and drives discovery through a hand-engineered modular framework that compensates for the LLM's limited intrinsic capacity for discovery. It plays the role of an external tutor, as the policies governing how candidate solutions are selected and stored are typically designed by human experts.

Definition: Unsupervised Scaffold. An unsupervised scaffold (i.e.,, learning-based) [13, 14] updates θ at test time via reinforcement learning (RL) over the LLM's self-generated experiences. Prior works still retain a database and a selection strategy (e.g.,, PUCT), but the surrounding framework is comparatively lightweight, corresponding to Vygotsky's *internalization* phase.

Strong performance under a supervised scaffold is not evidence that $\mathcal{M}_\theta$ has *internalized* discovery: since θ is frozen, any gain is jointly attributable to the LLM and its external framework, and corresponds to Vygotsky's assisted performance. Genuine internalization requires improvements to accrue to $\mathcal{M}_\theta$ itself—persisting even after the scaffolding is stripped away—which motivates the unsupervised regime. In this work, our goal is to equip the LLM with internalized discovery capability, enabling it to operate effectively under both supervised and unsupervised scaffolds.

D.3 Distinguishing Optimization from Reasoning and Agentic Tasks

We position optimization (or discovery) tasks as fundamentally distinct from both reasoning and agentic tasks, as summarized in Table 8.

Unlike reasoning tasks, which assume well-defined problems with known ground-truth solutions, optimization tasks are inherently open-ended: the objective is not to recover a known answer, but to discover improved or entirely novel solutions. This lack of ground truth shifts the success criterion from correctness to relative improvement over prior best-known results.

Compared to agentic tasks, which emphasize executing sequences of actions to satisfy predefined acceptance criteria in real-world environments, optimization tasks require searching over a substantially larger and more abstract solution space (e.g., programs, algorithms, or mathematical constructions). As a result, they exhibit longer time horizons and demand iterative refinement rather than single-pass execution.

These differences lead to a distinct evaluation paradigm. While reasoning and agentic tasks are typically evaluated using binary success signals, optimization tasks rely on deterministic but continuous metrics that enable partial progress to be measured and accumulated over time.

Taken together, optimization tasks necessitate a different role for language models: instead of acting purely as a reasoning engine or a planner, the model serves as a mutation operator within

Table 8: Key differences across three standard tasks: Reasoning vs. Agentic vs. Optimization.

Characteristic	Reasoning	Agentic	Optimization
Problem			
Objective	Solving a specific, well-defined problem	Complete real-world task	Discovering novel solutions, algorithms, or structures
Ground Truth	Known (Solution)	Known (Acceptance Criteria)	Unknown
Structure	Closed-form problem solving	Constrained execution	Open-ended problem solving
Complexity	P or NP (decidable)	Compositional	NP-Hard
Success Criterion	Solve — match ground truth	Complete — satisfy acceptance criteria	Improve — exceed previous best-known
Expertise	Expert-curated exam problems designed with known answers	Real-world application	Expert-verified open problems
Evaluation			
Verification	Deterministic, Binary	Deterministic, Binary	Deterministic, Continuous
Metric	Accuracy	Resolve rate	Relative improvement
Execution			
Env. Interaction	✗	✓	✓
Time Horizon	Short	Mid	Long
Role of LLM	Reasoning Engine	Planner and Executor	Mutation Operator
Search Space	Discrete answer space	Action trajectories over system state	Combinatorial / continuous program space

an evolutionary search process, iteratively proposing candidate solutions that can be evaluated and improved. This distinction motivates the design of $\mathcal{F}$inch, which is specifically tailored to support discovery-driven optimization.

E Additional Details of Dataset Construction Method

E.1 Systematic Error Breakdown Analysis

E.2 Benchmark Licenses

We construct our seed tasks from publicly available benchmarks, all of which are released under permissive open-source licenses (e.g., MIT, Apache 2.0, or CC-BY 4.0). Table 10 summarizes the license terms for the corresponding GitHub repositories and associated datasets for each benchmark. Our use of these resources is limited to non-commercial academic research and complies with the terms of their respective licenses, including all attribution requirements. Accordingly, $\mathcal{F}$inch Collection is released under the CC-BY 4.0 license, while our code and $\mathcal{F}$inch weights are released under the Apache 2.0 license. Both are compatible with the licenses of all upstream benchmarks listed above.

F Additional Implementation Details

Implementation details for $\mathcal{F}$inch-8B nanodiscover runs. nanodiscover is an open-source reproduction of TTT-Discover that does not depend on the Tinker API. It is publicly available at https://github.com/cheongalc/nanodiscover. Each search epoch in nanodiscover consists of five stages mirroring the TTT-Discover pipeline: (1) sampling parent solutions from the archive, (2) generating child solutions from parent solutions, (3) evaluating child solutions, (4) updating the archive, and (5) test-time training. Ray Data LLM (which orchestrates vLLM under the hood) is used for step (2), while DeepSpeed is used for step (5). Unless otherwise noted, all hyperparameters were matched to TTT-Discover as closely as possible.

All runs were conducted for 50 epochs on a single node with 4 GPUs and 96 logical CPU cores. For the Erdos task, the prompt informed the model that the evaluation budget was 1000 seconds, while the actual timeout was set to 1100 seconds. For both circle-packing tasks, the model was not informed of

Table 9: Breakdown of system-level errors filtered out (6,321 trajectories). Counts and percentages are calculated over the total of 6,321 trajectories.

Error category	Count	%	Description
`parent_missing_combined_score`	3,641	57.60	Parent program itself never produced a `combined_score`, so the parent→child improvement delta is undefined regardless of what the child did.
`artifact_error_type:timeout`	901	14.25	Evaluator wall-clock timeout: child program ran past its time budget (`artifacts.timeout = True`, `error_type = "timeout"`).
`failure_stage:correctness`	794	12.56	Child program ran to completion but produced numerically incorrect outputs (the evaluator's correctness check rejected the result).
`child_metrics_error_string`	605	9.57	`child_metrics.error` holds a human-readable failure message (e.g. `"C1 mismatch: reported X, computed Y"`) rather than a clean numeric metric.
`artifact_error_type:TimeoutError`	164	2.59	Async `TimeoutError` raised from the evaluator's task wrapper (typically during `cascade_setup`, `stage1`, or `stage2`).
`failure_stage:benchmark`	147	2.33	Failure inside the benchmark/timing harness *after* the correctness check passed (e.g. a crash during the performance-measurement loop).
`artifact_error_string`	61	0.97	`artifacts.error` contains a runtime exception string + traceback (e.g. `NameError`, syntax error, JAX trace error) with no structured `error_type` or `failure_stage`.
`failure_stage:syntax_check`	5	0.08	Generated code failed the pre-execution syntax check (could not be parsed or compiled).
`failure_stage:import`	2	0.03	Generated module failed to import (missing symbol, `ImportError`, or top-level exception during import).
`failure_stage:validation`	1	0.02	Generated solution failed schema/shape validation before execution.
Total	**6,321**	**100.00**	

Table 10: Benchmark Licenses

Benchmark	Github	Dataset
LLM-SRBench [5]	MIT License	-
FrontierCS [22]	MIT License	Apache 2.0
ALE-Bench [17]	Apache 2.0	-
GPU Mode	-	-
AlgoTune [23]	MIT License	MIT License
AlphaEvolve's Math Problems [7]	Apache License 2.0	CC-BY 4.0
Function Minimization [15]	Apache License 2.0	Apache License 2.0
SLDBench [39]	MIT License	-

the evaluation budget, and the actual timeout was 530 seconds. These timing configurations follow the TTT-Discover setup.

G Full List of Optimization Tasks in $\mathcal{F}$inch Collection

Table 11: Per-task descriptions for the **Mathematical Discovery** task group (14 tasks) evaluated under $\mathcal{F}$inch Collection. The "# of Traj." column reports the number of evolution trajectories we collected for each task (group total: 19,324).

Task Name	Objective	Description	# of Traj.
`circle_packing_rect`	$\max \sum r_i$	Pack equal-radius circles inside an axis-aligned rectangle without overlap; maximize the common radius under boundary and non-overlap constraints.	2,706
`erdos_min_overlap`	$\min M(n)$	Construct a witness function (step function on $[-1, 1]$) that gives a constructive upper bound on Erdős's minimum-overlap constant $M(n)$.	1,269
`first_autocorr_ineq`	$\min C_1$	First autocorrelation inequality: minimize $\|f * f\|_\infty$ over functions $f \geq 0$ supported on $[-1/4, 1/4]$ with $\int f = 1$.	1,792
`second_autocorr_ineq`	$\max C_2$	Second autocorrelation inequality: extremal $\|f*f\|$ under prescribed support and mass constraints.	1,451
`third_autocorr_ineq`	$\min C_3$	Third autocorrelation inequality: tighten the upper bound on the third autocorrelation constant arising in additive combinatorics.	836
`heilbronn_triangle`	max min area	Place n points in the unit square $[0, 1]^2$; maximize the area of the smallest triangle formed by any three points.	1,124
`heilbronn_convex_13`	max min area	Heilbronn-on-a-convex-region variant with n=13 points: maximize the minimum convex-hull area over all subsets of $k > 3$ points.	2,806
`heilbronn_convex_14`	max min area	Heilbronn-on-a-convex-region variant with n=14 points (otherwise as above).	1,454
`hexagon_packing_11`	min enclosing	Pack n=11 unit regular hexagons inside the smallest enclosing regular hexagon.	1,194
`hexagon_packing_12`	min enclosing	Pack n=12 unit regular hexagons inside the smallest enclosing regular hexagon.	1,198
`minimizing_max_min_dist_2`	max $d_{\min}/d_{\max}$	Place points in $[0, 1]^2$ so that the ratio of minimum to maximum pairwise distance is as close to 1 as possible (uniform 2D point distribution).	1,175
`minimizing_max_min_dist_3`	max $d_{\min}/d_{\max}$	Same as above but for points in $[0, 1]^3$ (uniform 3D point distribution).	1,082
`signal_processing`	multi-objective	Design a causal real-time filter for a noisy non-stationary time series, balancing fidelity, smoothness, lag, and false-trend detection.	908
`sums_diffs_finite_sets`	$\min \lvert A+A\rvert/\lvert A-A\rvert$	Construct a finite set $A \subset \mathbb{Z}$ minimizing the ratio between sumset and difference-set cardinalities (additive combinatorics).	329

Table 12: Per-task descriptions for the **Numerical Algorithm Optimization** task group (46 tasks) evaluated under $\mathcal{F}$inch Collection. The "# of Traj." column reports the number of evolution trajectories we collected for each task (group total: 5,932).

Task Name	Objective	Description	# of Traj.
`aes_gcm_encryption`	max speedup	Speed up AES-GCM authenticated encryption against a `cryptography`-library reference.	199
`affine_transform_2d`	max speedup	Speed up 2D affine image warping with cubic-spline interpolation against a SciPy reference.	200
`aircraft_wing_design`	max speedup	Speed up the geometric-programming aircraft-wing design problem (drag minimization under aerodynamic constraints).	197
`articulation_points`	max speedup	Find all articulation points in an undirected graph (cut vertices whose removal disconnects the graph).	200
`base64_encoding`	max speedup	Speed up Base64 encoding of binary data against a `base64` stdlib reference.	199

Continued on next page

Table 12 (continued)

Task Name	Objective	Description	# of Traj.
`battery_scheduling`	max speedup	Convex battery charge/discharge scheduling under price, capacity, and ramp constraints (CVXPY reference).	200
`chacha_encryption`	max speedup	Speed up ChaCha20 stream-cipher encryption against a `cryptography`-library reference.	199
`channel_capacity`	max speedup	Compute the channel capacity of a discrete memoryless channel by maximizing mutual information (CVXPY reference).	197
`chebyshev_center`	max speedup	Find the center of the largest inscribed ball of a polyhedron $P = \{x \mid a_i^\top x \leq b_i\}$ via an LP.	197
`cholesky_factorization`	max speedup	Speed up dense Cholesky factorization of a symmetric positive-definite matrix.	98
`clustering_outliers`	max speedup	Outlier-robust clustering of points in multidimensional space (HDBSCAN-style reference).	100
`communicability`	max speedup	Compute communicability $C(u, v)$ between all node pairs in an undirected graph (NetworkX reference).	93
`convex_hull`	max speedup	Compute the convex hull of a 2D point set (smallest convex polygon containing all points).	95
`convolve2d_full_fill`	max speedup	Speed up 2D convolution with full-output, fill boundary against `scipy.signal.convolve2d`.	198
`convolve_1d`	max speedup	Speed up 1D convolution over a list of array pairs against a SciPy reference.	100
`correlate_1d`	max speedup	Speed up 1D cross-correlation over a list of array pairs against a SciPy reference.	97
`count_connected_components`	max speedup	Count connected components of an undirected edge-list graph.	99
`count_riemann_zeta_zeros`	max speedup	Count nontrivial zeros of the Riemann zeta function with imaginary part in $(0, t]$.	98
`cumulative_simpson_1d`	max speedup	Cumulative integral of a 1D function via Simpson's rule (SciPy reference).	100
`cumulative_simpson_multid`	max speedup	Cumulative integral along the last axis of a multidimensional array via Simpson's rule.	98
`dct_type_I_scipy_fftpack`	max speedup	Speed up Type-I Discrete Cosine Transform against `scipy.fftpack`.	100
`delaunay`	max speedup	Compute Delaunay triangulation of a 2D point set (SciPy reference).	100
`dijkstra_from_indices`	max speedup	Shortest paths from a subset of source nodes to all others on a weighted undirected CSR graph.	99
`dst_type_II_scipy_fftpack`	max speedup	Speed up Type-II Discrete Sine Transform on a 2D array against `scipy.fftpack`.	100
`dynamic_assortment_planning`	max speedup	Dynamic assortment planning over T periods with N products under per-product capacities; maximize expected revenue.	98
`earth_movers_distance`	max speedup	Solve the optimal-transport problem between two histograms with a given cost matrix (POT reference).	96
`edge_expansion`	max speedup	Compute the edge expansion $\partial S/\lvert S\rvert$ for a node subset S in a directed graph.	99
`eigenvalues_complex`	max speedup	Eigenvalues of a real square matrix that may have complex eigenvalues (LAPACK reference).	91
`eigenvalues_real`	max speedup	Eigenvalues of a symmetric real matrix (LAPACK reference).	100
`eigenvectors_complex`	max speedup	Eigenpairs of a real square matrix (complex eigenvalues/eigenvectors).	200
`eigenvectors_real`	max speedup	Eigenpairs of a real symmetric matrix (orthonormal eigenvectors).	96
`elementwise_integration`	max speedup	Elementwise definite integration of Wright's Bessel function across an array of arguments.	99
`fft_cmplx_scipy_fftpack`	max speedup	N-dimensional complex FFT of a complex matrix against `scipy.fftpack`.	199
`fft_convolution`	max speedup	FFT-based convolution of two signals (overlap-add / circular-FFT reference).	200

Continued on next page

Table 12 (continued)

Task Name	Objective	Description	# of Traj.
`fft_real_scipy_fftpack`	max speedup	N-dimensional FFT of a real-valued matrix against `scipy.fftpack`.	99
`firls`	max speedup	Design a least-squares FIR filter for given frequency-band edges (SciPy `firls` reference).	99
`generalized_eigenvalues_complex`	max speedup	Generalized eigenvalues of (A, B) where A, B are real and the spectrum may be complex.	99
`generalized_eigenvalues_real`	max speedup	Generalized eigenvalues of (A, B) with real spectrum (symmetric/positive-definite case).	100
`generalized_eigenvectors_complex`	max speedup	Generalized eigenpairs of (A, B) with complex spectrum.	99
`generalized_eigenvectors_real`	max speedup	Generalized eigenpairs of (A, B) with real spectrum.	100
`graph_global_efficiency`	max speedup	Average inverse shortest-path length over all node pairs of an undirected graph (NetworkX reference).	98
`graph_isomorphism`	max speedup	Find a node mapping between two isomorphic undirected graphs (NetworkX VF2 reference).	100
`graph_laplacian`	max speedup	Compute the combinatorial or symmetric-normalized Laplacian of a sparse undirected graph.	99
`group_lasso`	max speedup	Logistic regression with group-lasso penalty over J feature groups (CVXPY reference).	100
`gzip_compression`	max speedup	Speed up Gzip compression of binary data against the `gzip` stdlib reference.	99
`lu_factorization`	max speedup	Speed up dense LU factorization with partial pivoting against a LAPACK reference.	199

Table 13: Per-task descriptions for the **Heuristic Optimization** task group (35 tasks) evaluated under $\mathcal{F}$inch Collection. The "# of Traj." column reports the number of evolution trajectories we collected for each task (group total: 18,902).

Task Name	Objective	Description	# of Traj.
`ahc001` (Ad Placement)	max score	Place axis-parallel rectangular ads on a 10000×10000 canvas, each containing a fixed query point and matching a target area; maximize the sum of advertiser-satisfaction scores.	929
`ahc002` (Walking on Tiles)	max score	From start cell on a 50×50 tiled grid, walk a self-avoiding path (no tile reused); maximize the sum of cell scores along the path.	471
`ahc003` (Shortest Path Queries)	max score	On a 30×30 grid graph with unknown edge lengths, answer 1000 online shortest-path queries while learning edge lengths from noisy feedback.	494
`ahc004` (Cyclic Substring Cover)	max score	Pack given strings as horizontal/vertical cyclic substrings of an $N\times N$ matrix; maximize the number of strings covered while minimizing matrix usage.	478
`ahc005` (Patrol Route)	min time	On an $N\times N$ obstacle map with weighted road squares, design a closed walk from the start so that every road square is line-of-sight visible; minimize total travel time.	425
`ahc006` (Food Delivery)	min length	Out of 1000 delivery orders on an 801×801 grid, select 50 and produce a closed tour from the office that visits each pickup before its drop-off; minimize tour length.	332
`ahc007` (Online MST)	min cost	Online Steiner-MST: edges of an embedded graph arrive one by one with revealed length $\in [d_i, 3d_i]$; decide on the fly whether to include each edge to ultimately span all terminals at minimum cost.	355
`ahc008` (Pet Herding)	max satisfaction	Control M humans on a 30×30 grid for 300 turns, placing fences to isolate N pets into pet-free regions; maximize the per-human area-isolation reward.	1,207
`ahc009` (Maze Walk)	max P(reach)	Output a single fixed action sequence $\in \{U, D, L, R\}^*$ that, under random execution slippage, maximizes the probability of reaching the goal in a 20×20 walled maze.	308

Continued on next page

Table 13 (continued)

Task Name	Objective	Description	# of Traj.
`ahc010` (Loop Tiles)	max length	Rotate 30×30 railroad-pattern tiles to form one closed loop of maximum total length.	366
`ahc011` (Sliding Tree Puzzle)	max connectivity	Slide tiles on an $N\times N$ board so the line patterns form one large connected tree; maximize tree size while staying within the move budget.	1,257
`ahc012` (Strawberry Cake)	max satisfaction	Cut a circular cake of radius 10^4 with at most K straight lines so that the strawberry counts in resulting pieces match a target multiset.	357
`ahc014` (RectJoin)	max score	In the RectJoin grid game, place dots and draw axis-aligned/diagonal rectangles repeatedly; maximize the weighted score of placed dots.	358
`ahc015` (Halloween Candy)	max clusters	In a 10×10 box where 100 candies of three flavors fall in a known order to random empty cells, choose a tilt direction before each candy to maximize same-flavor connected-component sizes.	1,255
`ahc016` (Graph Encoding)	min error	Given target M and noise rate ε, design M reference graphs $G_0,\ldots,G_{M-1}$ that remain pairwise distinguishable after random edge flipping and label permutation.	1,160
`ahc017` (Edge Repair Schedule)	min disruption	Schedule the one-time repair of each edge of a planar weighted graph across D days (at most K repairs/day); minimize the total inflation in shortest-path distances during repairs.	367
`ahc019` (Polycube Silhouette)	max match	Combine polycube blocks into a single 3D object whose front and right silhouettes match two given 2D monochrome targets.	184
`ahc020` (Antenna Placement)	max coverage	Choose a connected subgraph and per-vertex broadcast power on a planar graph with weighted edges so that all K residents are covered; minimize total power+edge cost.	188
`ahc021` (Pyramid Sort)	min swaps	Reorder balls in an N-tier pyramid via adjacent swaps so that the value at each position is at least its parent; minimize swap count.	189
`ahc024` (Map Compression)	min size	Re-tile an $n\times n$ multi-ward city map onto a smaller grid while preserving every ward's connectivity and inter-ward adjacency; minimize the resulting grid size.	1,231
`ahc025` (Weight Balancing)	min imbalance	Using only balance-scale comparisons of item subsets, partition N items of unknown weight into D groups of equal total weight within Q queries.	1,018
`ahc026` (Cardboard Boxes)	min cost	Carry out n uniquely-numbered boxes from m stacks in ascending order using up to 5000 stack-relocation moves; minimize total move cost.	1,094
`ahc027` (Cleaning Robot)	min dirt	Design a closed cleaning route on an $N\times N$ walled grid; minimize the steady-state average dirtiness across cells with cell-specific dirt-accumulation rates.	1,135
`ahc028` (Keyboard Typing)	min strokes	Given an $N\times N$ keyboard with letters in cells and M target words to type, output a finger trajectory whose visited letters contain every target word as a contiguous substring; minimize total moves.	191
`ahc030` (Polyomino Detection)	min queries	On an $N\times N$ island hiding M polyomino oil-fields of known shape, locate every field via interactive cell/region drilling queries with noisy feedback.	182
`ahc031` (Event Hall Reservation)	min penalty	Partition a $W\times W$ hall into N axis-aligned rectangles per day for D days, each rectangle satisfying a daily area request; minimize unmet-area + cross-day partition-change penalties.	181
`ahc032` (Stamp Pressing)	max sum	On an $N\times N$ integer board, repeatedly press 3×3 stamps from M stamp templates (mod 10^9+7); maximize the final cell-sum.	185
`ahc034` (Dump Truck Leveling)	min cost	Drive a dump truck on an $N\times N$ height-grid (total height zero), loading and unloading dirt to flatten the field; minimize total transport cost.	180

Continued on next page

Table 13 (continued)

Task Name	Objective	Description	# of Traj.
`ahc035` (Seed Crossing)	max yield	Over a sequence of crossing rounds, schedule pairings of $2N(N-1)$ multi-trait seeds to evolve a high-quality final population.	180
`ahc039` (Purse-Seine Fishing)	max catch	Construct a rectilinear polygon in the plane that encloses as many mackerels and excludes as many sardines as possible, subject to a perimeter budget.	882
`ahc041` (Rooted Tree Forest)	max attractiveness	Partition a planar graph into rooted trees so that the weighted sum $\sum_v (h_v+1)A_v$ over depths and beauty values is maximized.	188
`ahc042` (Oni-Fukunokami Push)	max remaining	On an $N\times N$ board with Oni and Fukunokami tokens, repeatedly push entire rows/columns to drive Oni off the board while keeping as many Fukunokami as possible.	186
`ahc044` (Cleaning Schedule Automaton)	min error	Choose a per-employee finite-state-automaton transition (a_i, b_i) so that, after a long simulated week sequence, employee i's cleaning count converges to a target T_i.	185
`ahc045` (Group Exploration with Uncertain Coordinates)	min length	Cluster N cities into tour groups of fixed sizes when each city's coordinates are only known up to a rectangular range, using a limited number of true-distance queries.	179
`ahc046` (Skating with Blocks)	min actions	On an $N\times N$ skating rink, visit target squares in order using `Move`, `Slide`, and block-placement actions; minimize total action count.	1,025

Table 14: Per-task descriptions for the **GPU Kernel Optimization** task group (4 tasks) evaluated under $\mathcal{F}$inch Collection. The "# of Traj." column reports the number of evolution trajectories we collected for each task (group total: 5,448).

Task Name	Objective	Description	# of Traj.
`grayscale`	max throughput	Triton/PyTorch kernel for RGB→grayscale image conversion on H100/H200 GPUs.	1,365
`mla_decode`	max throughput	Triton kernel for DeepSeek-V2/V3 multi-head latent-attention decoding.	1,406
`trimul`	max throughput	Triton kernel for the AlphaFold-3 triangle multiplicative update.	1,311
`vecadd`	max throughput	Triton/PyTorch kernel for FP16 vector addition on H100/H200 GPUs.	1,366

Table 15: Per-task descriptions for the **Constructive Search** task group (2 tasks) evaluated under $\mathcal{F}$inch Collection. The "# of Traj." column reports the number of evolution trajectories we collected for each task (group total: 3,536).

Task Name	Objective	Description	# of Traj.
`function_minimization`	$\min f$	Black-box minimization of $f(x,y) = \sin(x)\cos(y) + \sin(xy) + (x^2+y^2)/20$ on $[-5,5]^2$; evolve a random-search seed into an adaptive optimizer.	3,387
`k_module_problem`	max correct	Discover the correct 4-component pipeline configuration (5^4=625 possibilities) when the only feedback per query is the count of correctly placed modules (deceptive landscape).	149

Table 16: Per-task descriptions for the **SR – Bio Pop Growth** task group (24 tasks) evaluated under $\mathcal{F}$inch Collection. The "# of Traj." column reports the number of evolution trajectories we collected for each task (group total: 28,034).

Task Name	Objective	Description	# of Traj.
BPG0	min fit error	Ground-truth: $0.954\,(1 - P/96.9)\,P + 0.954\,P^{0.333}$.	1,178
BPG1	min fit error	Ground-truth: $0.316\,P\,e^{-0.0541t} + 0.316\,P^2/(9.87\,P + 1)$.	1,176
BPG2	min fit error	Ground-truth: $0.257\,P\sin(0.722\,t) + 0.115\,P\,e^{-0.0304t}$.	1,169
BPG3	min fit error	Ground-truth: $0.845\,(-1 + P/5.12)(1 - P/34.4)\,P + 0.845\,(1 - e^{-0.0969P})\,P$.	1,161
BPG4	min fit error	Ground-truth: $0.173\,(1 - P/48.5)\,P + 0.173\,P/(1 + e^{-1.52(-0.924+P)})$.	1,166
BPG5	min fit error	Ground-truth: $0.92\,(1 - P/84)\,P + 0.92\,P^2/(7.53\,P + 1)$.	1,169
BPG6	min fit error	Ground-truth: $-6.22 \cdot 1.41\,P + 0.858\,(1 - P/79.2)\,P + 0.858\,P^{0.333} + 0.858\,P$.	1,161
BPG7	min fit error	Ground-truth: $0.721\,(-1 + P/6.65)(1 - P/13.5)\,P + 0.721\,(1 - P/13.5)\,P + 0.721\,P^{0.333}$.	1,162
BPG8	min fit error	Ground-truth: $0.991\,(1 - P/39)\,P + 0.991\,P^{0.333} + 0.991\,P$.	1,166
BPG9	min fit error	Ground-truth: $0.17\,(-1 + P/1.05)(1 - P/10.2)\,P + 0.17\,(1 - P/10.2)\,P + 0.17\,(1 - e^{-0.0971P})\,P$.	1,173
BPG10	min fit error	Ground-truth: $0.101\,P^{0.333} + 0.101\,P$.	1,164
BPG11	min fit error	Ground-truth: $0.712\,(1 - P/68.9)\,P + 0.712\,P^{0.333} + 0.712\,P\,e^{-0.0346t}$.	1,172
BPG12	min fit error	Ground-truth: $0.877\,P\sin(0.567\,t) + 0.701\,(1 - P/65.8)\,P$.	1,166
BPG13	min fit error	Ground-truth: $0.201\,(-1 + P/5.64)(1 - P/10.2)\,P + 0.201\,P + 0.201\,P/(1 + e^{-5.64(-0.634+P)})$.	1,171
BPG14	min fit error	Ground-truth: $0.114\,(1 - P/40.7)\,P + 0.114\,(1 - e^{-0.0837P})\,P + 0.114\,P\,e^{-0.0837t}$.	1,161
BPG15	min fit error	Ground-truth: $0.487\,P^{0.333} + 0.487\,P\,e^{-0.0858t}$.	1,167
BPG16	min fit error	Ground-truth: $0.868\,(-1 + P/6.06)(1 - P/14.2)\,P + 0.868\,P^{0.333} + 0.868\,P$.	1,169
BPG17	min fit error	Ground-truth: $0.769\,(-1 + P/8.67)(1 - P/14.3)\,P + 0.769\,P^{0.333}$.	1,171
BPG18	min fit error	Ground-truth: $0.477\,P\sin(0.776\,t) + 0.445\,(1 - P/51.1)\,P + 0.445\,P$.	1,163
BPG19	min fit error	Ground-truth: $0.421\,(1 - P/84.4)\,P + 0.421\,P/t^{2.59}$.	1,175
BPG20	min fit error	Ground-truth: $0.139\,(-1 + P/8.04)(1 - P/70)\,P + 0.139\,(1 - P/70)\,P + 0.139\,P/(1 + e^{-8.04(-0.589+P)})$.	1,166
BPG21	min fit error	Ground-truth: $0.14\,(-1 + P/4.53)(1 - P/78.5)\,P + 0.14\,P/t^{4.53}$.	1,168
BPG22	min fit error	Ground-truth: $0.118\,(1 - e^{-0.0272P})\,P + 0.118\,P\,e^{-0.0272t}$.	1,171
BPG23	min fit error	Ground-truth: $0.598\,(1 - P/32.9)\,P + 0.598\,(1 - e^{-0.0768P})\,P$.	1,169

Table 17: Per-task descriptions for the **SR – Chem Reaction** task group (12 tasks) evaluated under $\mathcal{F}$inch Collection. The "# of Traj." column reports the number of evolution trajectories we collected for each task (group total: 13,975).

Task Name	Objective	Description	# of Traj.
CRK0	min fit error	Ground-truth: $-0.19\,A^2 + 0.19\,A^2/(0.75\,A^4 + 1)$.	1,174
CRK1	min fit error	Ground-truth: $-0.773\,A^2 - 0.773\,A + 0.773\,\cos(\log(A + 1))$.	1,157

Continued on next page

Table 17 (continued)

Task Name	Objective	Description	# of Traj.
CRK2	min fit error	Ground-truth: $-0.195\,A + 0.195\,\cos(\log(A+1))$.	1,160
CRK3	min fit error	Ground-truth: $-0.548\,A^2 - 0.548\,A\,e^{-0.548t} + 0.548\,\cos(\log(A+1))$.	1,159
CRK4	min fit error	Ground-truth: $-0.115\,A^2 + 0.115\,A\,\log(0.257\,t+1)$.	1,167
CRK5	min fit error	Ground-truth: $-0.325\,\sqrt{A} + 0.325\,A^{0.333}$.	1,164
CRK6	min fit error	Ground-truth: $-0.447\,A\,e^{-0.447t} + 0.447\,\sin(\sqrt{A})$.	1,160
CRK7	min fit error	Ground-truth: $-0.732\,A\,e^{-0.732t} + 0.732\,\cos(\log(A+1))$.	1,164
CRK8	min fit error	Ground-truth: $-0.679\,A^2 - 0.679\,A + 0.679\,\sin(\log(A+1))$.	1,177
CRK9	min fit error	Ground-truth: $-0.26\,\sqrt{A} + 0.26\,\cos(\log(A+1))$.	1,162
CRK10	min fit error	Ground-truth: $-0.175\,A^2 + 0.175\,\sin(\log(A+1))$.	1,162
CRK11	min fit error	Ground-truth: $-0.882\,A^2 + 0.882\,\sin(\sqrt{A})$.	1,169

Table 18: Per-task descriptions for the **SR – Physics Oscillation** task group (44 tasks) evaluated under $\mathcal{F}$inch Collection. The "# of Traj." column reports the number of evolution trajectories we collected for each task (group total: 51,211).

Task Name	Objective	Description	# of Traj.
PO0	min fit error	Ground-truth: $F_0\sin t - \beta\sin v - \omega_0^2 x^3 - \omega_0^2 x\,e^{-\lvert x\rvert}$.	1,171
PO1	min fit error	Ground-truth: $F_0\sin t - \omega_0^2 x - \omega_0^2 x\,e^{-\lvert x\rvert}$.	1,173
PO2	min fit error	Ground-truth: $-\alpha v^3 - \mu(1-x^2)v - \omega_0^2 x - \omega_0^2 x\,e^{-\lvert x\rvert}$.	1,161
PO3	min fit error	Ground-truth: $F_0\sin t - \beta\sin v - 2\beta v$.	1,172
PO4	min fit error	Ground-truth: $F_0\sin t - \alpha v^3 - \omega_0^2(\gamma\lvert v\rvert^{0.333}+1)x - \omega_0^2 x$.	1,166
PO5	min fit error	Ground-truth: $-\beta\sin v - 2\beta v - \omega_0^2(\gamma\lvert v\rvert^{0.333}+1)x - \omega_0^2 x^3 - \omega_0^2 x$.	1,165
PO6	min fit error	Ground-truth: $-\beta\log(\lvert v\rvert+1) - 2\beta v - \omega_0^2 x^3$.	1,167
PO7	min fit error	Ground-truth: $-\alpha v^3 - \beta\lvert v\rvert^{0.333} - \omega_0^2 x^3$.	1,171
PO8	min fit error	Ground-truth: $-\beta\lvert v\rvert^{0.333} - \omega_0^2 x^3$.	1,169
PO9	min fit error	Ground-truth: $F_0\sin t - \mu(1-x^2)v - \omega_0^2(\gamma\lvert v\rvert^{0.333}+1)x - \omega_0^2 x$.	1,163
PO10	min fit error	Ground-truth: $F_0\sin t - \omega_0^2(\gamma t+1)x - \omega_0^2 x^3 - \omega_0^2 x$.	1,170
PO11	min fit error	Ground-truth: $-\beta\sin v - \omega_0^2(\gamma t+1)x - \omega_0^2 x^3$.	1,168
PO12	min fit error	Ground-truth: $F_0\sin t - \alpha v^3 - \beta\lvert v\rvert^{0.333} - \omega_0^2(\gamma t+1)x - \omega_0^2 x$.	1,160
PO13	min fit error	Ground-truth: $F_0\sin t - \mu(1-x^2)v - \omega_0^2(\gamma\lvert v\rvert^{0.333}+1)x$.	1,167
PO14	min fit error	Ground-truth: $F_0\sin t - \beta\log(\lvert v\rvert+1) - \beta\sin v - 2\beta v - \mu(1-x^2)v$.	1,149
PO15	min fit error	Ground-truth: $F_0\sin t - \omega_0^2(\gamma\lvert v\rvert^{0.333}+1)x - \omega_0^2 x - \omega_0^2 x\,e^{-\lvert x\rvert}$.	1,151
PO16	min fit error	Ground-truth: $F_0\sin t - \beta\sin(x)\,v - \beta\sin v - \omega_0^2 x^3$.	1,166
PO17	min fit error	Ground-truth: $F_0\sin t - \beta\sin(x)\,v - 2\beta v - \omega_0^2 x$.	1,170
PO18	min fit error	Ground-truth: $-\beta\sin(x)\,v - \omega_0^2 x$.	1,161
PO19	min fit error	Ground-truth: $-2\beta v - \omega_0^2 x\,e^{-\lvert x\rvert}$.	1,154
PO20	min fit error	Ground-truth: $-\alpha v^3 - \beta\log(\lvert v\rvert+1) - 2\beta v - \mu(1-v^2)v - \omega_0^2(\gamma\lvert v\rvert^{0.333}+1)x$.	1,159
PO21	min fit error	Ground-truth: $F_0\sin t - \beta\sin(x)\,v$.	1,165
PO22	min fit error	Ground-truth: $-2\beta v - \beta e^{-\lvert x\rvert}v - \mu(1-x^2)v - \omega_0^2 x^3$.	1,165
PO23	min fit error	Ground-truth: $F_0\sin t - \beta\log(\lvert v\rvert+1) - \omega_0^2 x\,e^{-\lvert x\rvert}$.	1,161
PO24	min fit error	Ground-truth: $F_0\sin t - \alpha v^3 - \beta\log(\lvert v\rvert+1)$.	1,161
PO25	min fit error	Ground-truth: $F_0\sin t - \beta\sin v$.	1,166
PO26	min fit error	Ground-truth: $F_0\sin t - \beta\log(\lvert v\rvert+1) - 2\beta v - \omega_0^2 x^3$.	1,159
PO27	min fit error	Ground-truth: $F_0\sin t - \alpha v^3 - 2\beta v - \beta e^{-\lvert v\rvert}v$.	1,151

Continued on next page

Table 18 (continued)

Task Name	Objective	Description	# of Traj.
PO28	min fit error	Ground-truth: $-2\beta v - \omega_0^2(\gamma\|v\|^{0.333}+1)x - \omega_0^2 x^3 - \omega_0^2 x$.	1,164
PO29	min fit error	Ground-truth: $-\mu(1-x^2)v - \omega_0^2(\gamma t+1)x - \omega_0^2 x^3$.	1,163
PO30	min fit error	Ground-truth: $-\alpha v^3 - \beta\sin(x)\,v - \beta\sin v - \omega_0^2 x^3$.	1,156
PO31	min fit error	Ground-truth: $-\omega_0^2(\gamma\|v\|^{0.333}+1)x - \omega_0^2 x^3$.	1,155
PO32	min fit error	Ground-truth: $F_0\sin t - \alpha v^3 - \beta e^{-\|v\|}v - \omega_0^2 x^3$.	1,154
PO33	min fit error	Ground-truth: $-2\beta v - \mu(1-v^2)v - \omega_0^2(\gamma t+1)x - \omega_0^2 x$.	1,178
PO34	min fit error	Ground-truth: $-2\beta v - \mu(1-v^2)v - \omega_0^2(\gamma\|v\|^{0.333}+1)x$.	1,160
PO35	min fit error	Ground-truth: $F_0\sin t - \beta\sin v - \omega_0^2(\gamma\|v\|^{0.333}+1)x$.	1,166
PO36	min fit error	Ground-truth: $F_0\sin t - \beta e^{-\|x\|}v$.	1,170
PO37	min fit error	Ground-truth: $F_0\sin t - \alpha v^3 - 2\beta v - \omega_0^2(\gamma t+1)x$.	1,172
PO38	min fit error	Ground-truth: $-\beta\sin v - \mu(1-x^2)v - \omega_0^2 x\,e^{-\|x\|}$.	1,160
PO39	min fit error	Ground-truth: $F_0\sin t - \alpha v^3 - \beta e^{-\|x\|}v - \mu(1-v^2)v$.	1,175
PO40	min fit error	Ground-truth: $F_0\sin t - \beta\|v\|^{0.333} - \omega_0^2(\gamma\|v\|^{0.333}+1)x - \omega_0^2 x^3 - \omega_0^2 x$.	1,155
PO41	min fit error	Ground-truth: $-\mu(1-x^2)v - \omega_0^2 x\,e^{-\|x\|}$.	1,166
PO42	min fit error	Ground-truth: $F_0\sin t - \alpha v^3 - \beta\sin(x)\,v - 2\beta v$.	1,173
PO43	min fit error	Ground-truth: $F_0\sin t - \beta\sin(x)\,v - 2\beta v - \mu(1-x^2)v - \omega_0^2 x\,e^{-\|x\|}$.	1,163

Table 19: Per-task descriptions for the **Competitive Programming (Frontier-CS)** task group (172 tasks) evaluated under $\mathcal{F}$inch Collection. The "# of Traj." column reports the number of evolution trajectories we collected for each task (group total: 46,380).

Task Name	Objective	Description	# of Traj.
FCS-0: Pack the Polyominoes (Reflections Allowed)	min area	Minimize the bounding-rectangle area for a polyomino-packing instance allowing rotations and reflections.	539
FCS-1: Treasure Packing	judge score	Treasure-packing constructive optimization.	539
FCS-2: Permutation	judge score	Construct a permutation satisfying problem-specific constraints.	522
FCS-3: Ring Lamp Permutation Recovery (interactive)	min queries	Recover a hidden cyclic permutation via lamp-toggle queries with adjacency feedback.	529
FCS-4: Matrix k-th Smallest (interactive)	min queries	Identify the k-th smallest element of a hidden matrix through interactive queries.	524
FCS-5: Hamiltonian Path Challenge	judge score	Construct or approximate a Hamiltonian path on a given graph instance.	538
FCS-6: World Map	judge score	Constructive world-map design.	522
FCS-7: Build a Computer	judge score	Combinatorial construction problem ("Build a Computer").	515
FCS-8: The Empress	judge score	"The Empress" constructive optimization problem.	523
FCS-9: Tree Permutation Sort via Matchings	min swaps	Sort a tree-labeled permutation to identity using minimum-matching swaps.	548
FCS-10: Tree Distance	judge score	Tree-distance computation problem.	540
FCS-11: Palindrome Path	judge score	Find a palindromic path in a graph or grid.	522
FCS-13: Robot Explosion on Grid (interactive)	≤ 3000 steps	Detonate a grid robot in ≤ 3000 steps using interactive cell-marking queries.	536
FCS-14: Hidden Cycle Length (interactive)	min queries	Determine the length of a hidden cycle through interactive queries.	530
FCS-15: Permutation Minimization via 3-Part Cut/Swap	min lex order	Minimize lexicographic permutation order using $\leq 4n$ cut-and-swap operations.	497
FCS-16: Identify Chord	judge score	Identify a hidden chord on a circle through queries.	514
FCS-17: Permutation (Interactive)	judge score	Interactive permutation-recovery variant.	530
FCS-22: A+B Problem (Hard)	judge score	Hard variant of the A+B problem (high-precision / large-input).	526

Continued on next page

Table 19 (continued)

Task Name	Objective	Description	# of Traj.
FCS-23: A = B (Sequence Transformation)	judge score	Transform sequence A into B via constrained operations.	545
FCS-24: Almost Monochromatic Permutation	judge score	Construct an "almost monochromatic" permutation.	527
FCS-25: Graph Connectivity via Neighborhood Queries (interactive)	min queries	Decide graph connectivity using only neighborhood queries.	530
FCS-26: OgreSort	judge score	Custom comparison-sort variant ("OgreSort").	524
FCS-27: No-Axis-Rectangle Point Placement	max points	Place the maximum number of points in an $n \times m$ grid with no axis-parallel rectangle of any 4 of them.	522
FCS-28: Hacking the Project	judge score	"Hacking the Project" constructive problem.	539
FCS-30: Hidden Mole in Tree (interactive)	min queries	Locate a hidden mole on a tree via interactive vertex queries.	529
FCS-33: Permutation (Modified Version)	judge score	Modified-version of an interactive permutation problem.	518
FCS-35: Unique Element in Doubled Array (interactive)	min queries	Identify the unique element in a doubled array via comparison queries.	527
FCS-36: Hack!	judge score	"Hack!" constructive challenge.	501
FCS-40: Interactive RBS	judge score	Recover a hidden balanced-bracket sequence interactively.	546
FCS-41: Brilliant Umbrella Sequence	judge score	Construct a strictly increasing sequence whose consecutive GCDs are also strictly increasing.	519
FCS-42: Unit Square Packing with Rotations	min area	Pack n unit squares allowing arbitrary rotation in the smallest enclosing square.	554
FCS-43: Bigger Sokoban 40k	judge score	Larger-scale Sokoban-style puzzle (40k cells).	528
FCS-44: Traveling Santa with Carrot Constraint	judge score	Tour problem with carrot-pickup constraints.	528
FCS-45: Balanced Graph Partitioning (DIMACS10)	min cut	Balanced partitioning of DIMACS10 graphs minimizing edge cut.	538
FCS-46: Job Shop Scheduling (JSPLIB)	min makespan	Classical job-shop scheduling on JSPLIB instances.	555
FCS-47: 2D Rectangular Knapsack with 90° Rotations	max value	2D rectangle knapsack with 90° rotations allowed.	538
FCS-48: Sphere Packing in a Cube	min cube	Pack n unit spheres inside the smallest enclosing cube.	546
FCS-50: Weighted Set Cover	min cost	Minimum-cost weighted set cover.	551
FCS-52: Geemu	judge score	Game-state constructive problem ("Geemu").	537
FCS-53: G2. Inter Active (Hard Version)	judge score	Hard interactive variant "G2. Inter Active".	500
FCS-54: Centroid Guess	min queries	Interactive centroid-guessing on a tree.	511
FCS-57: Rooted Tree Identification via Path-Sum Queries (interactive)	min queries	Identify a rooted tree using path-sum queries.	543
FCS-58: Inverse Counting Path	judge score	Inverse-counting path-construction problem.	519
FCS-59: Limited Shuffle Restoring	judge score	Restore an array using a limited number of shuffles.	544
FCS-60: Probing the Disk (interactive)	min queries	Detect a hidden disk's center and radius via line-segment light probes.	537
FCS-61: Let's Go! New Adventure	judge score	"Let's Go! New Adventure" constructive problem.	168
FCS-62: Ball Moving Game	judge score	Constructive ball-moving puzzle.	184
FCS-63: Space Thief	judge score	"Space Thief" constructive problem.	169
FCS-64: Subset Sum Closest to Target	min gap	Subset-sum closest to a target value.	192
FCS-68: Pen Ink Selection (interactive)	min queries	Interactive pen-ink-selection problem.	173
FCS-69: Distinct Magic Words (Spell Power)	max distinct	Maximize the number of distinct substrings ("magic words") in a string.	163
FCS-70: Treasure Hunt	judge score	"Treasure Hunt" constructive problem.	181
FCS-72: Rush-Hour Puzzle	max steps	On a 6×6 Rush-Hour puzzle, maximize the minimum steps required for any solution.	179

Continued on next page

Table 19 (continued)

Task Name	Objective	Description	# of Traj.
FCS-73: Inversion	judge score	"Inversion" constructive problem.	177
FCS-75: Black and White	judge score	"Black and White" constructive problem.	174
FCS-77: Izzy Wager Predictions	min wrong	Minimize wrong guesses across 10,000 interactive prediction rounds.	182
FCS-79: H. Hack	judge score	"H. Hack" constructive problem.	170
FCS-80: Indiana Jones and the Uniform Cave	judge score	"Indiana Jones" interactive cave-traversal problem.	182
FCS-81: Stone Slate String Recovery	judge score	Recover a hidden string written on a stone slate.	175
FCS-82: Permutation Recovery via OR Queries	min queries	Recover a hidden permutation in $\leq$ 4269 bitwise-OR queries.	168
FCS-83: Completely Multiplicative Function Partial-Sum Minimization	min partial sum	For a completely multiplicative $f(\cdot)$, minimize $\max_k \lvert\sum_{i\leq k} f(i)\rvert$.	171
FCS-85: Maze	judge score	"Maze" constructive problem.	188
FCS-86: Hidden Tree	judge score	Recover a hidden tree via queries.	148
FCS-87: Graph Coloring	judge score	Graph-coloring decision/optimization.	179
FCS-89: Tree Reconstruction via Steiner-Membership Queries	min queries	Reconstruct a tree using Steiner-set membership queries.	174
FCS-93: Greedy	judge score	"Greedy" constructive problem.	176
FCS-101: Circuit	judge score	"Circuit" constructive problem.	182
FCS-104: Dishonest Student Attendance (interactive)	min queries	Interactive "dishonest-student attendance" problem.	176
FCS-106: Hidden Bipartite Graph	judge score	Recover a hidden bipartite graph.	184
FCS-107: F. Guess Divisors Count	min queries	Guess a number's divisor count using interactive queries.	173
FCS-108: Ring Lock with Arc Alignment (interactive)	min ops	Open a ring lock by aligning arcs with the fewest interactive operations.	176
FCS-109: Knight's Longest Tour	max length	Find the longest knight's tour on a chess subboard.	175
FCS-110: 8x14 Digit Grid (Maximize Readable Numbers)	max count	Fill an 8×14 digit grid to maximize the number of 8-directional readable numerals.	187
FCS-111: Distinct Pairwise XOR Set	max size	Construct a set whose pairwise XORs are all distinct.	171
FCS-112: SphereSpread	judge score	"SphereSpread" constructive problem.	156
FCS-113: Three Baskets Center-Ball Moves	judge score	Constructive ball-distribution between three baskets.	174
FCS-117: Line	judge score	"Line" constructive problem.	176
FCS-119: Operators	judge score	Constructive operator-insertion puzzle.	175
FCS-120: Da Bai (interactive)	min queries	"Da Bai" interactive problem.	162
FCS-121: DNA Matching	judge score	DNA-matching constructive problem.	181
FCS-122: Hidden Editor Line-Width Discovery (interactive)	min queries	Discover a hidden editor's line width via interactive queries.	167
FCS-123: Hidden Integer via Set-Membership Queries	min queries	Recover a hidden integer through set-membership queries.	178
FCS-124: AveragePermutation	judge score	"AveragePermutation" constructive problem.	179
FCS-125: Hidden Mineral Pair Discovery (Device Query)	min queries	Discover a hidden mineral pair via device queries.	184
FCS-127: Hidden Diamond Box (interactive)	min queries	Locate a hidden diamond inside a box through interactive probes.	171
FCS-132: Robot-based Chairman Location Finder (interactive)	min queries	Locate a hidden chairman on a graph using a movable robot.	178
FCS-133: Line Segment Brush Area	max area	Maximize the brush-stroke area covered by a set of line segments.	177
FCS-134: Guess Number	min queries	Interactive number-guessing variant.	179
FCS-135: Interactive Problem (Standard Solutions Won't Work)	judge score	Adversarial interactive problem requiring non-standard strategy.	167
FCS-137: Kangaroos	judge score	"Kangaroos" constructive problem.	174

Continued on next page

Table 19 (continued)

Task Name	Objective	Description	# of Traj.
FCS-138: Fabulous Fungus Frenzy	judge score	"Fabulous Fungus Frenzy" constructive problem.	183
FCS-140: Mineral Deposits (interactive)	min queries	Interactive mineral-deposit identification problem.	178
FCS-141: Bakery Survey	judge score	"Bakery Survey" constructive problem.	182
FCS-142: Ball Game	judge score	"Ball Game" constructive problem.	179
FCS-143: Texas Hold'em Training (interactive)	judge score	Interactive Texas-Hold'em training problem.	192
FCS-144: Find Median	min queries	Interactively find the median of a hidden array.	174
FCS-145: Meituan Cup: Number Loop	judge score	Meituan Cup "Number Loop" constructive problem.	181
FCS-147: Ad Rectangle Placement	max score	10000×10000 ad-rectangle placement (AHC001-style).	179
FCS-148: Tile Path Score Maximization	max score	50×50 tile-path score maximization (AHC002-style).	181
FCS-149: Grid Shortest Path with Unknown Edges	max score	Online shortest-path queries on a grid with unknown edge weights.	164
FCS-150: Torus Matrix Recovery from Substrings	judge score	Recover a torus-matrix matching given cyclic substrings.	179
FCS-151: Patrol Route Minimization	min time	Patrol-route minimization on weighted road grid (AHC005-style).	182
FCS-152: Food Delivery Route with 50 Orders	min length	Select 50 of 1000 orders on a cyclic delivery route (AHC006-style).	180
FCS-153: Online MST with Edge Length Uncertainty	min cost	Online MST under edge-length uncertainty (AHC003-style).	173
FCS-154: Pet Herding via Partitions	max score	Pet herding on a 30×30 grid via partitions (AHC008-style).	187
FCS-155: Robust Commute String	max score	Robust action-sequence design under stochastic execution (AHC004-style).	174
FCS-156: Loop Line Toy Train	max score	Toy-train loop-line track-laying (AHC009-style).	185
FCS-157: Sliding Puzzle Tree Target	max score	Sliding-tile puzzle to form a target spanning tree (AHC011-style).	184
FCS-158: Cake Cutting with Strawberries	judge score	Cut a circular cake with $\leq K$ lines to match a strawberry-count target (AHC012-style).	176
FCS-159: RectJoin Grid Puzzle	max score	RectJoin grid-puzzle constructive problem (AHC014-style).	174
FCS-160: Halloween Candy Tilt (10x10)	max clusters	Tilt-controlled candy-clustering on a 10×10 box (AHC015-style).	180
FCS-161: TV Network (Planar Graph)	judge score	Planar-graph TV-network construction (AHC010-style).	181
FCS-162: Pyramid Ball Swap	min swaps	Pyramid-ball reorder via adjacent swaps (AHC021-style).	173
FCS-163: Map Tile Coloring Minimization	min size	Map-tile coloring/compression preserving adjacencies (AHC024-style).	175
FCS-164: Cardboard Box Carry-Out	min cost	Cardboard-box carry-out with minimum lifting energy (AHC026-style).	177
FCS-165: Lucky String Keyboard	judge score	Keyboard-layout design for lucky-string typing.	176
FCS-166: Dump Truck Ground Leveling	min cost	Dump-truck ground leveling on an $N\times N$ grid (AHC034-style).	177
FCS-167: Purse Seine Fishing	max score	Rectilinear-polygon mackerel/sardine separation (AHC039-style).	166
FCS-168: Rooted Christmas Tree Beauty	max score	Rooted-tree beauty maximization.	177
FCS-169: Oni wa Soto, Fuku wa Uchi (Setsubun board game)	max score	Setsubun-themed Oni/Fuku push-board game.	172
FCS-170: Cleaning Duty Assignment	judge score	Cleaning-duty assignment under per-employee targets.	181

Continued on next page

Table 19 (continued)

Task Name	Objective	Description	# of Traj.
FCS-171: Skating Rink Block Placement	min ops	Skating-rink block-placement order-of-visit problem (AHC046-style).	177
FCS-174: Graph 3-Coloring	judge score	3-coloring of a given graph.	186
FCS-175: 3-SAT	judge score	3-SAT instance (Frontier-CS variant).	184
FCS-176: 3-SAT	judge score	3-SAT instance (different size class).	190
FCS-177: Graph 3-Coloring	judge score	3-coloring instance (different size class).	188
FCS-178: 3-SAT	judge score	3-SAT instance (third size class).	177
FCS-179: Subset Sum	judge score	Subset-sum instance.	179
FCS-180: Graph Isomorphism	judge score	Graph-isomorphism instance.	176
FCS-181: Binary Quadratic Assignment Problem	judge score	Binary-QAP instance.	193
FCS-182: Vertex Cover Challenge	judge score	Vertex-cover challenge.	171
FCS-183: Maximum Independent Set Challenge	judge score	Maximum-independent-set challenge.	176
FCS-184: Maximum Independent Set Challenge	judge score	Maximum-independent-set challenge (different instance).	177
FCS-185: Maximum Clique Challenge	judge score	Maximum-clique challenge.	175
FCS-186: Graph Coloring Challenge	judge score	Graph-coloring challenge.	185
FCS-187: Clique Cover Challenge	judge score	Clique-cover challenge.	181
FCS-188: LCS Challenge (Approximation)	judge score	Approximate longest-common-subsequence challenge.	186
FCS-189: Edit Distance Challenge (Approximation)	judge score	Approximate edit-distance challenge.	171
FCS-192: Max-Cut	judge score	Max-cut instance.	184
FCS-193: Max-2-SAT	judge score	Max-2-SAT instance.	178
FCS-203: Chameleon	judge score	"Chameleon" constructive problem.	169
FCS-205: Sequence Transformation	judge score	Constrained sequence-transformation problem.	180
FCS-207: Efficient Sorting	judge score	Cost-bounded sorting variant.	182
FCS-209: Hidden Weights	judge score	Recover hidden item weights via comparison queries.	168
FCS-210: Military Exercise: Fighter Scheduling & Base Strikes	judge score	Joint fighter scheduling and base-strike planning.	182
FCS-211: Communication Robots	judge score	"Communication Robots" constructive problem.	185
FCS-212: I Wanna Cross the Grid	judge score	"I Wanna Cross the Grid" constructive problem.	178
FCS-213: Sequence Shift (Moqueve)	judge score	"moqueve" sequence-shift problem.	179
FCS-214: Sequence Reversal (Requese)	judge score	"requese" sequence-reversal problem.	182
FCS-217: Super Dango Maker	judge score	"Super Dango Maker" constructive problem.	171
FCS-220: Playing Around the Table	judge score	"Playing Around the Table" constructive problem.	183
FCS-222: Hedgehog Graph	judge score	"Hedgehog" graph-construction problem.	176
FCS-225: Permutation Set Merging	judge score	Permutation-set merging problem.	169
FCS-227: 4-Way Partition: max LIS+LDS+LIS+LDS	max sum	4-way partition maximizing the combined LIS+LDS+LIS+LDS lengths.	177
FCS-228: 01-String: Substrings with $\|0\| = \|1\|^2$	judge score	Count $\{0,1\}$-string substrings where $\#0 = (\#1)^2$.	180
FCS-229: LIS after Interval Shifts (Cheated LIS)	max LIS	Maximize LIS after ≤ 10 interval shifts.	187
FCS-231: Differentiating Games	judge score	"Differentiating Games" constructive problem.	178
FCS-233: Snake	judge score	"Snake" constructive problem.	186
FCS-239: Shortest 3-Hop DAG Augmentation	min edges	Add minimum edges to a $0 \to n$ chain so every $v < u$ has a ≤ 3-edge path.	174
FCS-241: Boolean Expression Construction (AND/OR from Truth Table)	judge score	Construct an AND/OR Boolean expression matching a given truth table.	183
FCS-243: Hidden Position Determination on Grid Map (interactive)	min queries	Locate a hidden position on a grid map via interactive queries.	188
FCS-245: Asesino	judge score	"Asesino" constructive problem.	180

Continued on next page

Table 19 (continued)

Task Name	Objective	Description	# of Traj.
FCS-247: Sequence Swap to Reach B	judge score	Transform A into B via swap-and-adjust ops $(A_i, A_j) \to (A_j{-}1, A_i{+}1)$.	170
FCS-248: Drone Delivery	judge score	"Drone Delivery" constructive problem.	172
FCS-249: X-OR	judge score	"X-OR" constructive problem.	188
FCS-252: Hotel	judge score	"Hotel" constructive problem.	156
FCS-253: Roads	judge score	"Roads" constructive problem.	175
FCS-254: Pepe Racing	judge score	"Pepe Racing" constructive problem.	183
FCS-255: Magnets	judge score	"Magnets" constructive problem.	186
FCS-256: Palindromic Paths	judge score	"Palindromic Paths" constructive problem.	180
FCS-257: Omkar and Modes	judge score	"Omkar and Modes" constructive problem.	174
FCS-258: Network Synchronization: Finding Dual Anomalies	judge score	Network-synchronization dual-anomaly detection.	178

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer:[Yes]

   Justification: The abstract and Section 1 clearly state our three contributions—(i) the EVOLUTION FINE-TUNING (EFT) mid-training paradigm, (ii) the $\mathcal{F}$inch Collection dataset of 156K trajectories spanning 10 domains and 371 tasks, and (iii) the $\mathcal{F}$inch model family (2B–9B)—and these are substantiated by the held-out evaluations in Section 4 and the analyses in Section 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: In Appendix A

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: This is an empirical paper introducing a fine-tuning paradigm, a dataset, and a model family; it does not contain formal theorems or proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

Justification: Section 3.1 and Appendix E describe the trajectory-collection pipeline (search scaffold, teacher model, domains, tasks); Section 4 and Appendix F list training and inference hyperparameters (e.g.,, T=100, temperature 0.7, top-p 0.95, 30K-token context, scaffold defaults), the held-out task suite, and the nanodiscover variant we use for test-time learning.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will release $\mathcal{F}$inch Collection, the $\mathcal{F}$inch-{2,4,8,9}B model checkpoints, and the training/evaluation code under open licenses upon publication.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 and Appendix F report the train/held-out task split, search-scaffold settings (T, parallel size, temperature, top-p, max tokens, island defaults), and the test-time learning configurations (matching the original TTT-Discover); fine-tuning hyperparameters for $\mathcal{F}$inch-{2,4,8,9}B are listed in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [N/A]

   Justification: N/A

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: In Section 3.3 and Appendix F.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: We have reviewed the NeurIPS Code of Ethics and confirm that the research—training open-source LLMs on programmatically generated optimization-search trajectories—does not involve human subjects, sensitive personal data, or deployment scenarios that conflict with the guidelines; anonymity is preserved in this submission.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Appendix B

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: N/A

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the OpenEvolve scaffold [15], the Qwen3 / Qwen3.5 model families [16] used for trajectory generation and as base LLMs, the TTT-Discover algorithm [14] and the nanodiscover, and each held-out benchmark (CALICO, FrontierCS, AlphaEvolve mathematics tasks, ALE-Bench, CO-Bench, etc.); the corresponding licenses are listed in Appendix E.2.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

Justification: $\mathcal{F}$inch Collection (156K trajectories, 10 domains, 371 tasks) and the $\mathcal{F}$inch-{2,4,8,9}B model family are the two new assets; their construction, schema (per-step program, score, evolution-type annotations), training recipe, and intended use are documented in Section 3.1 and Appendix E, and the released artifacts will ship with a datasheet/model card.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: N/A

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: N/A

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs are central to this work: Qwen3.5-397B-A17B is the teacher model whose evolutionary-search trajectories form $\mathcal{F}$inch Collection, Qwen3-{2,4,8,9}B are the open-source backbones that we fine-tune to obtain $\mathcal{F}$inch, and the held-out evaluations themselves use LLMs as mutation operators inside the OpenEvolve and nanodiscover scaffolds; all of these usages are described in Sections 3.1 and 4.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.