FOUNDRY: Host-Owned Trust and Memory for Long-Horizon Agent Swarms

Anonymous Author(s)
Affiliation
Address
email

Abstract

Long-horizon LLM-agent swarms promise to turn frontier models into discovery engines, but they fail when agents are allowed to own the two states that determine progress: trusted evaluation and persistent memory. Without an external evaluator, agents reward-hack local reports, trust stale or incomplete validation, and claim improvements that do not survive independent checking. Without shared memory, independent agents waste budget re-discovering hypotheses that earlier agents have already falsified. We introduce FOUNDRY, a host-coordinated control plane for scientific and engineering discovery swarms. Its design follows one principle: agents propose, while the host verifies and remembers. The host owns an authoritative evaluator, an established-facts registry that compounds evidence across independent agents, and a hypervisor $\rightarrow$ orchestrator $\rightarrow$ solver hierarchy that routes work through file-based mailboxes under token budgets. This host-owned trust-and-memory substrate is model-agnostic, supports both phased waves and long-lived swarms, and can resume runs across hundreds of millions of tokens.

Across combinatorial mathematics, GPU-kernel engineering, and computational biology, FOUNDRY instantiates without per-domain harness changes and improves the public state of the art under host-verified scoring. It tightens the best-known upper bound on the Erdős minimum-overlap constant, achieves the fastest GPUMode TriMul kernel runtime on H100, and matches the state-of-the-art on the OpenProblems single-cell denoising benchmark. These results suggest that progress in agentic discovery depends not only on stronger agents, but on the system boundary that separates untrusted proposal generation from trusted verification and memory.

1 Introduction

Agentic AI is shifting from merely pursuing stronger base models toward asking how intelligence should be organized. Recent systems increasingly translate ideas from human organizations, biological collective behavior, and systems engineering into computational substrates for agents: roles, delegation, verification, memory, routing, and feedback. Swarm intelligence offers one such blueprint: classical swarm systems show how simple agents can produce adaptive collective behavior through local interaction, self-organization, and environmental feedback [Beni and Wang, 1993]. The same organizing principle is now reappearing in LLM-agent discovery systems, where populations of agents search over programs, constructions, kernels, and experimental procedures, then use archives, validators, and feedback to turn individual attempts into collective progress. Systems such as FunSearch, AlphaEvolve, TTT-Discover, OpenEvolve, and ShinkaEvolve have already advanced mathematical bounds, GPU kernels, matrix-multiplication routines, computational-biology tasks, and combinatorial search problems [Romera-Paredes et al., 2024, Novikov et al., 2025, Yuksekgonul et al., 2026, Sharma, 2025, Lange et al., 2025].

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

Yet as these systems move from isolated attempts to long-horizon swarm runs, their system structure becomes the bottleneck. First, the trusted score is often operationally coupled to the agent: the same agent that proposes a candidate also runs local validation scripts, times the candidate, and reports the resulting number. Second, cross-attempt memory is usually organized as a flat archive of candidate *programs*, rather than a structured record of *hypotheses* that have been supported, contradicted, or falsified. It does not preserve the boundary of what has already been tried and ruled out. Actually, for long-horizon swarms, this negative knowledge is often the scarce resource.

These two design choices result in two compounding failure modes. **(1)** Collapse of the trust boundary: When the only evaluator the agent can see is its own local report, the agent can optimize the report rather than the underlying quantity. **(2)** Wasted re-derivation: When independent solvers work in parallel without host-owned shared memory, each agent privately rediscovers the same negative results, such as *"annealing past the local minimum does not help"* or *"step-like perturbations are unstable past resolution $n = 64$"*. A few thousand dollars of frontier-model agent time can therefore evaporate into redundant computation that the next agent will repeat. Existing multi-agent frameworks [Wu et al., 2023, Hong et al., 2024, Qian et al., 2024] pass messages between agents, but the message a discovery swarm most needs is not merely *"what should I work on next"*; it is *"what has every prior agent already shown does not work"*.

The field has spent the last year asking *"how do we train better discovery agents?"* The TTT-Discover, EvoTune, and ThetaEvolve lineages answer by adding training-time signal at inference time [Yuksekgonul et al., 2026, Surina et al., 2025, Wang et al., 2025b]. We argue this is the second question. The first is structural: *"how do we organize frozen agents so they compound their own progress?"* Both failure modes above share a root cause: the agent owns the evaluator and the shared memory. Civilization's progress-compounding institutions take the opposite stance. Peer review uses parties with no interest in the result; open-source code review merges commits by reviewers who did not write them; replicated experiments are the only ones that count. The collapse of that separation into the agent itself is what produces both reward-hacking and wasted re-derivation. The training-free evolutionary lineage [Romera-Paredes et al., 2024, Novikov et al., 2025, Sharma, 2025, Lange et al., 2025] grounds candidates against an external scorer, which addresses the first half of this collapse, but its archive carries forward candidate programs, not falsified hypotheses, and the trusted scoring code typically lives inside the same process the agent runs in. The structural fix is to move both invariants out of the agent: the trusted score, and the cross-agent memory of what has already been ruled out.

We propose FOUNDRY, a host-coordinated metaharness for long-horizon agent swarms. FOUNDRY treats agents as frozen proposal generators: they may search, edit, test, and propose findings, but they do not own the trusted score or the persistent memory. Both are host-owned invariants. The host process owns three mechanisms that agents do not. First, it owns a trusted evaluator that re-derives every accepted score from first principles and never trusts the agent's reported scores or findings. The same evaluator is mounted read-only inside the agent container so that agents can self-check, but modifying local code cannot affect the authoritative run. Second, the host manages the findings of the agent swarm, driving strategy diversity across the swarm by injecting select findings and instructions into subsequent agent prompts according to swarm-level signals. Third, the host owns a hypervisor $\rightarrow$ orchestrator $\rightarrow$ solver hierarchy that routes work through file-based mailboxes under a run-level token budget. Across both phased waves and long-lived swarms, the same invariant holds: *agents propose; the host verifies and remembers*.

In summary, we make the following contributions:

1. **From the trust perspective,** we identify a structural failure mode in long-horizon discovery swarms: agents are routinely allowed to own both the evaluator and the memory, collapsing the trust boundary and preventing progress from compounding across attempts. This motivates a strict separation in which the host re-derives every accepted score from first principles and never trusts the agent's reported number (Section 3).
2. **From the system perspective,** we introduce FOUNDRY, a host-coordinated metaharness for frozen frontier coding agents such as Claude Code and Codex, organized as a hypervisor $\rightarrow$ orchestrator $\rightarrow$ solver hierarchy that routes work through file-based mailboxes under a run-level token budget (Section 3). Its central design principle is that trust and memory live at the host, not in the agents.
3. **From the memory perspective,** we design a confidence-weighted established-facts registry with explicit promotion states (`candidate/supported/established/retired`) driven by

independent support and contradiction, allowing future agents to inherit not only successful programs but also falsified hypotheses (Section 3).

4. Across three discovery benchmarks spanning combinatorics (Erdős minimum-overlap C_5), GPU kernel optimization (H100 TriMul), and scientific machine learning (OpenProblems Denoising), FOUNDRY delivers improvements without per-domain harness changes: a tighter upper bound on C_5, a faster TriMul kernel, and matches the current state of the art on OpenProblems Denoising (Section 4).

2 Related work

Context and memory management for long-horizon LLM agents. Recent LLM systems iteratively improve outputs by incorporating feedback from prior attempts. Editable-context and hierarchical-memory approaches [Zhang et al., 2026, Packer et al., 2024] organize past information across iterations; reflective methods [Shinn et al., 2023, Madaan et al., 2023] feed model-generated feedback into subsequent prompts; and test-time search and scaling [Yao et al., 2023, Snell et al., 2024] structure inference and the rollout-verifier mix. These approaches improve organization and reuse within a single session, but neither cross independent agent sessions nor separate the trusted scorer from the agent.

LLM-driven evolutionary search and discovery. A second line pairs LLMs with cross-attempt archives. FunSearch [Romera-Paredes et al., 2024] uses island-based program archives; AlphaEvolve [Novikov et al., 2025] adds MAP-Elites and multi-objective scoring across roughly fifty open mathematics problems; open-source replications and extensions [Sharma, 2025, Lange et al., 2025] build on AlphaEvolve with novelty-driven selection. AlphaTensor and AlphaDev [Fawzi et al., 2022, Mankowitz et al., 2023] train AlphaZero-style policies on specialized architectures for matrix multiplication and sorting. A learning-based thread adapts the generator via test-time RL or online DPO [Yuksekgonul et al., 2026, Surina et al., 2025, Wang et al., 2025b]. K-Search [Cao et al., 2026] replaces the flat archive with a structured world-model search tree for GPU kernels. Across these threads, cross-attempt memory is an archive of candidate *programs* and the trusted scorer is typically a Python function inside the agent's own process.

Multi-agent orchestration frameworks. Multi-agent LLM frameworks [Wu et al., 2023, Li et al., 2023, Hong et al., 2024, Qian et al., 2024, Wang et al., 2025a] provide message-passing, role-assignment, and conversation-management primitives. They organize how agents talk to each other but neither contribute SOTA-advancing mechanisms nor enforce a host-resident trust boundary; they sit underneath FOUNDRY rather than competing with it.

FOUNDRY differs from these lines on two structural axes. The trusted evaluator is host-resident and unreadable from inside the container (§3); the agent's reported score is replaced by an independent host-side recomputation. The cross-attempt memory is a confidence-weighted hypothesis registry with promotion semantics (§3), not a flat archive: it carries forward falsified hypotheses, demotes them on cross-source contradiction, and re-injects survivors into every subsequent agent's prompt. The same coordinator drives three unrelated benchmarks under three `ProblemSpecs` with no per-domain code changes (§3), under both phased waves and long-lived swarms; the result is a training-free system that compounds progress across independent agents.

3 FOUNDRY

FOUNDRY implements three design choices that move trust and memory out of the agents. The trusted evaluator $\mathcal{E}$ is host-resident, with only a read-only mirror $\mathcal{E}_{\text{ro}}$ exposed to each policy π_i. The cross-attempt memory $\mathcal{M}_t$ is host-curated: only the host writes to it, and only the injection ι exposes it to a future policy. The policies themselves are realized as two roles: a pool of M frozen LLM coding agents that act as *solvers*, and a single *orchestrator* that reads solver transcripts and curates $\mathcal{M}_t$. Algorithm 1 states the resulting discovery loop.

Trust boundary. The host owns the authoritative copy of $\mathcal{E}$ and re-runs it on every accepted candidate; the agent sees only $\mathcal{E}_{\text{ro}}$, which lets a policy self-check its work but cannot affect what the host accepts.

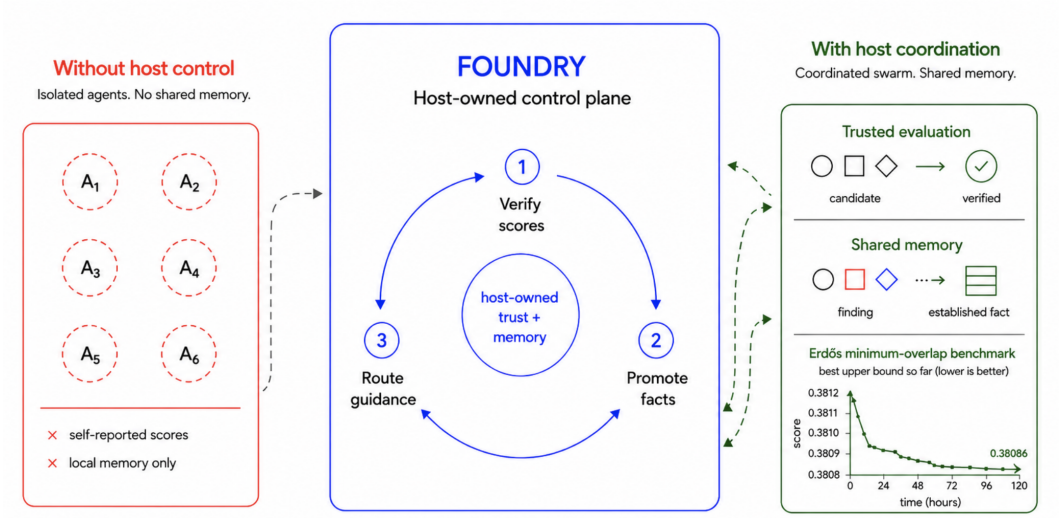

Figure 1. FOUNDRY's host-owned control plane. The coordinator loads state and launches the swarm; the problem adapter layer supplies schemas, prompt builders, a program bank, and the trusted-evaluate entry point. The host-owned control plane sits outside every solver container and exposes three components to the run: a trusted evaluator that recomputes scores, a facts registry that promotes findings from candidate to established, and an orchestrator that routes guidance. Containerized LLM coding-agent solvers receive prompts and guidance from the host and return artifacts and findings. The invariant that trust and shared memory remain host-owned is enforced by the dashed shared-state checkpoint boundary; the durability layer (process supervision and a run-level budget sentinel) keeps long swarm runs recoverable.

For each submission $a_i^{(t)} = (x_i^{(t)}, \hat{s}_i^{(t)}, F_i^{(t)})$, the host accepts the candidate iff

$$\Phi\big(x_i^{(t)}\big) = 1 \quad \text{and} \quad \big|\, \mathcal{E}(x_i^{(t)}) - \hat{s}_i^{(t)} \,\big| \le \varepsilon, \tag{1}$$

where Φ is a problem-specific schema validator. Mismatches are logged and discarded; findings $F_i^{(t)}$ are processed independently of acceptance, so a rejected candidate may still contribute hypotheses to $\mathcal{M}_t$.

Host-curated cross-attempt memory. The memory $\mathcal{M}_t$ preserves findings across agents and across rounds, so independent solvers do not re-derive the same negative results (e.g., *"warm starts upsampled by non-integer ratios inflate C_5"*). Because ι is the only inter-policy channel, no solver observes another's intermediate state; cross-agent influence flows entirely through findings the host has admitted to memory. FOUNDRY uses two realizations of the update operator U. In *swarm* mode, the orchestrator distills cross-solver evidence into broadcast messages that enter $\mathcal{M}_t$ when it promotes them. In *phased* mode, agents emit hypotheses tagged with per-agent endorsement weight $\sigma_+(h, v) \in [0.25, 3.0]$ and contradiction weight $\sigma_-(h, v) \in [0.25, 3.0]$, and the host applies the rule

$$\text{label}(h) = \begin{cases} \text{retired} & \text{if previously deprecated,} \\ \text{candidate} & \text{else if } w_{\text{challenge}}(h) \ge w_{\text{support}}(h), \\ \text{established} & \text{else if } c_{\text{score}}(h) \ge 3.0 \text{ and } |\mathcal{V}_+(h)| \ge 2, \\ \text{supported} & \text{else if } c_{\text{score}}(h) \ge 1.5, \\ \text{candidate} & \text{otherwise,} \end{cases} \tag{2}$$

where $\mathcal{V}_+(h) = \{v : \sigma_+(h, v) > 0\}$ and $c_{\text{score}} = w_{\text{support}} - w_{\text{challenge}}$. Both realizations share a structural invariant: a hypothesis enters the next policy's context only after independent corroboration by at least two distinct agents, which prevents any single confident agent from anchoring the run on a finding it cannot independently corroborate.

Per-problem specification. The same coordinator drives all three benchmarks; each problem supplies Φ, $\mathcal{E}$, prompt builders, and a metric direction, and no code path branches on the problem identity.

Algorithm 1 The FOUNDRY discovery loop.

Require: policies $\{\pi_i\}_{i=1}^{M}$, evaluator $\mathcal{E}$, schema validator Φ, baseline x_0, horizon T, tolerance ε

$\mathcal{M}_0 \leftarrow \emptyset$; $x^\star \leftarrow x_0$
for $t = 0, 1, \ldots, T-1$ **do**
  for policy i in parallel **do**
    $c_i^{(t)} \leftarrow \iota(\mathcal{M}_t, i)$; $a_i^{(t)} = (x_i^{(t)}, \hat{s}_i^{(t)}, F_i^{(t)}) \sim \pi_i(\cdot \mid c_i^{(t)})$
    accept $a_i^{(t)}$ iff $\Phi(x_i^{(t)}) = 1$ and $|\mathcal{E}(x_i^{(t)}) - \hat{s}_i^{(t)}| \le \varepsilon$ {Eq. 1}
  end for
  $\mathcal{M}_{t+1} \leftarrow U\big(\mathcal{M}_t, \{a_i^{(t)}, \mathcal{E}(x_i^{(t)})\}_i\big)$
  $x^\star \leftarrow \operatorname{argmin}\{\mathcal{E}(x) : x \in \{x^\star\} \cup \text{accepted}\}$
end for
return $x^\star$

4 Results

We evaluate FOUNDRY on three problems in three independent domains: discrete mathematics (Erdős minimum-overlap), GPU-kernel engineering (GPUMode TriMul on H100), and computational biology (OpenProblems single-cell denoising). For each problem, we report the best known human result and the best known AI result, and we compare FOUNDRY's host-verified score against both. Every accepted candidate in this section is independently re-evaluated outside the agent's container by the trusted evaluator (§3), so the column we report is directly comparable to the cited prior numbers when those were also produced under independent re-evaluation on the same hardware.

4.1 Erdős minimum-overlap

Task and setup. This is a classic problem in combinatorial number theory, posed by Erdős in 1955. Following AlphaEvolve and TTT-Discover, we work in the relaxed setting that asks for the smallest C_5 such that every step function $h : [0,2] \to [0,1]$ with $\int h\,dx = 1$ satisfies $\sup_k \int h(x)\,(1 - h(x+k))\,dx \ge C_5$. A candidate is a numerical array of length n representing such a step function, produced by an agent's thinking tokens followed by Python code that either constructs a new step function or modifies an existing one. The host parses and executes the code, renormalizes the result to satisfy $h(x) \in [0,1]$ and $\sum_i h_i = n/2$, and recomputes the reward as $\max(\texttt{np.correlate}(h, 1-h, \texttt{mode=full}) \cdot 2/n)$, or returns ∞ if the artifact fails schema validation or contains no `generator_code`.

Method	Model	C_5 (↓)	Notes
Best human			
Haugland 2016	—	0.380927	pre-AI human upper bound
Best known AI			
AlphaEvolve	Gemini 2.0 Pro+Flash	0.380924	training-free evolutionary search
TTT-Discover	gpt-oss-120b	0.380876	test-time RL on open model
FOUNDRY	Claude Code	**0.380863**	training-free, host-verified

Table 1. Erdős minimum-overlap upper bounds. FOUNDRY sets the new state-of-the-art at $C_5 = 0.380863$, advancing TTT-Discover's prior best by 1.3×10^{-5}.

Result. Bounds before AlphaEvolve were $0.379005 < c < 0.380927$, with the upper bound due to Haugland [2016] and the lower bound due to White [2023]; AlphaEvolve improved the upper bound to 0.380924 via verifier-based evolutionary search, and TTT-Discover improved it further to 0.380876 via test-time RL on an open model. FOUNDRY improves the upper bound to $\mathbf{0.380863}$, surpassing TTT-Discover's 0.380876 by an absolute margin of 1.3×10^{-5}. This advance is achieved with frozen frontier-model coding agents and no per-problem fine-tuning.

Discovered construction. The winning step function exhibits two structural features that FOUNDRY's registry promoted to `established` during the run: approximate antisymmetry $h(x) \approx 1 - h(2-x)$,

and low small-lag autocorrelation. Both were derived independently by multiple solver agents before promotion and were re-injected into the prompt of every subsequent solver. The discovery recipe alternates between sequential linear programming on a $\sigma = 0.10$ Gaussian-shaken warm-start and trust-constr polish at larger shake amplitudes; §5 walks through the trajectory in detail.

4.2 GPU-kernel engineering

Task and setup. We apply FOUNDRY to the GPUMode TriMul benchmark, a single-H100 optimization of the triangular multiplicative-update kernel used in AlphaFold's Evoformer block [Jumper et al., 2021]. The benchmark suite is the upstream `gpu-mode/reference-kernels` problem set with seven (sequence-length, batch, dim, hidden-dim) shapes; the score is the geometric mean of per-shape runtimes in microseconds. A candidate is GPU-kernel code in Triton or CUDA, produced by an agent's thinking tokens followed by kernel code that compiles and runs under the upstream evaluator on a single H100. The host reward is the negative geometric mean across the seven benchmark shapes, in microseconds; the kernel fails-closed if correctness checks fail or it times out.

Method	**Model**	**H100** (↓, μs)	**Std** (μs)
`zeyushen`	—	1365	±13
TTT-Discover	gpt-oss-120b	1208	±19
`shiyegao`	—	1092	±2
K-Search	GPT-5.2 + Gemini-3-Pro	1014	±3
FOUNDRY	Claude Code	**989**	**±4**

Table 2. GPUMode TriMul runtimes on H100, geometric mean across the seven benchmark shapes; standard deviation across outer iterations of the same evaluator. FOUNDRY advances K-Search by $\sim 2.5\%$, the strongest public submission (`shiyegao`) by $\sim 9.4\%$, and TTT-Discover by $\sim 18.1\%$.

Result. TTT-Discover targets the same kernel via test-time RL on gpt-oss-120b [Yuksekgonul et al., 2026], and K-Search uses a structured world-model search tree [Cao et al., 2026]; `shiyegao` and `zeyushen` are public GPUMode H100 submissions. To make the margins directly comparable, we re-evaluate every comparator under a single H100 environment with the upstream TriMul evaluator; only kernels re-evaluated by us appear in Table 2. FOUNDRY's best kernel achieves a geometric-mean runtime of $\mathbf{989 \pm 4}\,\mu$s on H100, a $\sim 18.1\%$ reduction against TTT-Discover ($1208 \pm 19\,\mu$s), a $\sim 9.4\%$ reduction against the strongest re-evaluated public submission (`shiyegao`, $1092 \pm 2\,\mu$s), and a $\sim 2.5\%$ reduction against K-Search ($1014 \pm 3\,\mu$s). Because the agent has shell access and could in principle bias its own runtime measurement (warm caches, JIT artifacts), only reproducible runtimes under an evaluator the agent cannot read are directly comparable; FOUNDRY exposes that margin by design (§3).

Novelty. Three structural choices distinguish FOUNDRY's kernel from the comparators in Table 2.

Shape-aware fusion topology. The kernel selects a different fusion topology depending on the channel-to-hidden-dim ratio. When $C >$ `hidden_dim`, the dim$= 384$ regime that dominates the geometric mean, it fuses the left projection, left gate, and out gate into a single three-output kernel that shares one x_{norm} load across all three, with a separate two-output kernel handling the right projection and right gate. When $C \leq$ `hidden_dim`, it instead uses two two-output projection-and-gate kernels and recomputes the out-gate dot product inline from x_{norm} in the final fused kernel, avoiding a third intermediate write. TTT-Discover and K-Search compile a single fixed proj-and-gate kernel that is reused on every shape; `shiyegao` ships a C++/CUDA extension that does not branch on shape; `zeyushen`'s leaderboard kernel is a single Triton LayerNorm targeting one seqlen path. The conditional fusion saves one full x_{norm} read on the geomean-dominating shape (~ 770 MB, $\sim 256\,\mu$s of bandwidth) without regressing the smaller-C shapes.

End-to-end shape-keyed buffer cache. All intermediate fp16 tensors, including the output-projection buffer, are cached by a tuple-keyed shape signature, and the weights are cached fp32→fp16 through a weakref-keyed dictionary, so that repeated calls on the same shape skip every allocation and every type conversion. K-Search caches a subset of these buffers; TTT, `shiyegao`, and `zeyushen` allocate intermediate buffers fresh on each call. Adding the output-projection buffer to the cache (relative to the closest in-suite variant) accounts for the final ~ 0.5–0.8% of geometric-mean reduction.

Pure Triton end to end. The pipeline is implemented entirely in Triton with cuBLAS only for the batched matmul. `shiyegao` relies on a C++/CUDA `torch.utils.cpp_extension` kernel (~ 2245 LOC); the FOUNDRY kernel is ~ 391 LOC. Keeping the kernel in one mutable language preserved the option for solvers to iterate on the fusion topology and tile sizes locally, which is how the kernel arrived at the conditional split above.

4.3 Single-cell denoising

Task and setup. We apply FOUNDRY to the OpenProblems Denoising task from the OpenProblems benchmark suite [Luecken et al., 2025], which evaluates algorithms that recover true expression values from noisy UMI counts in single-cell RNA-sequencing data. Following Batson et al. [2019], the benchmark partitions observed molecules into training and test sets via binomial sampling and scores a denoised training matrix against the held-out test counts; this provides a proxy for accuracy against true expression values without requiring external ground truth. A candidate is a Python callable mapping a sparse count matrix to a dense denoised matrix of the same shape, produced by an agent's thinking tokens followed by code that the host parses and runs. The benchmark reports two complementary metrics: mean-squared error (MSE) in log-normalized space, which measures overall reconstruction accuracy, and Poisson negative log-likelihood, which assesses how well the denoised counts match the statistical properties expected of count data; each is normalized so that no denoising scores 0 and perfect denoising scores 1. The reward is the mean of the two normalized scores, or 0 if the algorithm exceeds the 400-second host time budget on either dataset or fails the Poisson constraint we add to focus search on MSE (normalized Poisson ≥ 0.97). The benchmark provides three datasets: PBMC, Pancreas, and Tabula Muris Senis Lung, in order of size; we use Pancreas for in-loop evaluation and report performance on the held-out PBMC and Tabula datasets.

		PBMC			Tabula		
Method	**Model**	**Score** (↑)	**MSE** (↓)	**Poisson** (↓)	**Score** (↑)	**MSE** (↓)	**Poisson** (↓)
Best human							
MAGIC (A, R)	—	0.64	0.19	0.05	0.64	0.18	0.03
MAGIC (R)	—	0.64	0.19	0.05	0.64	0.18	0.03
ALRA (S, R)	—	0.50	0.26	0.05	0.47	0.27	0.03
Best known AI							
OpenEvolve	gpt-oss-120b	0.70	0.16	0.05	0.71	0.15	0.03
TTT-Discover	gpt-oss-120b	0.71	**0.15**	0.05	0.73	**0.14**	0.03
FOUNDRY	Claude Code	**0.72**	**0.15**	**0.05**	**0.73**	**0.14**	**0.03**

Table 3. OpenProblems Denoising scores on the held-out PBMC and Tabula datasets. Each per-dataset score is the mean of the normalized MSE and Poisson metrics (higher is better). FOUNDRY matches the prior state-of-the-art set by TTT-Discover within reporting precision, tying it on per-dataset MSE (0.15 on PBMC, 0.14 on Tabula). Scores cited for prior work follow Yuksekgonul et al. [2026]; (A) approximate solver, (R) reversed normalization, (S) sqrt normalization.

Result. The strongest pre-AI human entries on the OpenProblems leaderboard are MAGIC [Van Dijk et al., 2018] with reversed normalization (and an approximate-solver variant) and ALRA [Linderman et al., 2022] with sqrt normalization; TTT-Discover [Yuksekgonul et al., 2026] produced the prior best AI result on this benchmark via test-time RL on `gpt-oss-120b` initialized from MAGIC code, and OpenEvolve provides additional non-RL coding-agent baselines reported in that work. FOUNDRY matches TTT-Discover within reporting precision: PBMC score 0.72 vs. 0.71, Tabula tied at 0.73, and per-dataset MSE tied at 0.15 on PBMC and 0.14 on Tabula, with the Poisson constraint passing on both held-out datasets. This is achieved with frozen Claude Code agents and no per-problem fine-tuning, against a baseline whose discovery recipe was test-time-trained directly on this benchmark.

5 Case Study: Coordinating a 17-agent Erdős swarm

We illustrate the coordination mechanism through one long-swarm run on the Erdős minimum-overlap problem, in which 17 solver agents and one orchestrator agent run in parallel for 119 wall-clock

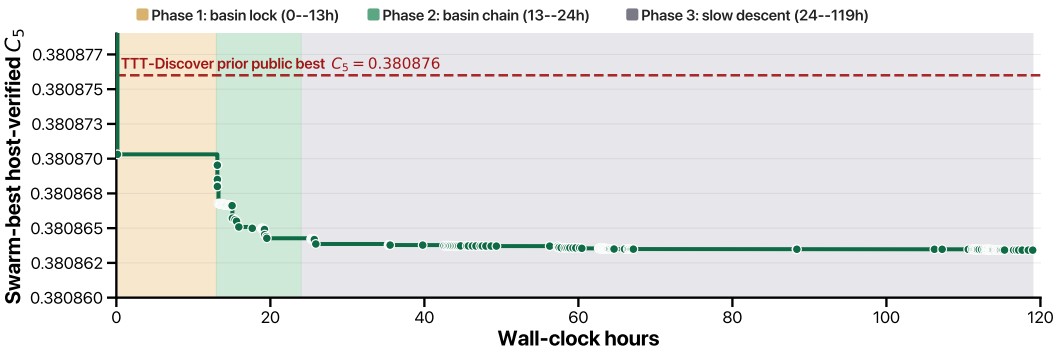

Figure 2. Swarm-best host-verified C_5 over the first 24 hours of an Erdős long-swarm run with 17 solver agents and one orchestrator agent. *Phase 1* (tan) is the basin lock: polishing public reference constructions plateaus at $C_5 \approx 0.380870$. *Phase 2* (green) is the basin chain triggered when one agent's $\sigma = 0.10$ Gaussian shake at hour 13 is broadcast by the orchestrator to all peers; six independent agents reproduce the recipe and the swarm-best descends in discrete steps to 0.380863. The chain saturates by hour ~ 20 and the floor holds for the remaining ~ 95 hours of the run (Phase 3, not shown). The horizontal dashed line marks the prior public best ($C_5 = 0.380876$, TTT-Discover [Yuksekgonul et al., 2026]); the run advances it by 1.3×10^{-5}. This case study illustrates the coordination mechanism behind the headline result in Table 1.

hours and submit 942 host-verified candidates. Solver agents are independent frontier-model coding sessions in separate containers with shell access; the orchestrator is a separate session that reads every solver's events and heartbeats, polls the host's verification feedback, and writes short natural-language guidance into a per-solver inbound channel. No agent is given an explicit search algorithm; each decides its own polish, shake, parameterization, and refinement strategy. Every step-down in the swarm-best trajectory (Figure 2) aligns with a single orchestrator broadcast, and each broadcast aggregates diagnostic signal that no individual agent could produce alone.

Phase 1, basin lock (hours 0–13). Diagnosed problem. Agents warm-start from public reference constructions, reaching $C_5 = 0.380870$ within minutes, then apply a wide repertoire of local optimization techniques (sequential linear programming, mass-transfer descent, projected gradient, smooth-max gradient, active-subgradient cutting plane). None descends below the warm-start floor; the basin appears locally locked. **Rationale.** If the basin is locally locked under continuous polish, the swarm needs structurally distinct perturbations to escape it; the orchestrator should canonicalize this diagnosis so agents stop redundantly polishing the same basin. **Mechanism.** The orchestrator aggregates cross-agent evidence (continuous polish is stationary up to 64K tangent-direction tests with a 21-D null space), publishes a basin-lock diagnosis to the swarm's shared scratch space, and recommends four alternatives: interpolate to higher n then polish; binarize at high n then run swap simulated annealing; random restart via a large-σ Gaussian shake; or a different parameterization (Fourier coefficients, dual on White's spectral lower bound). **Observed effect.** Agents partition across the four directions. Random restart at $n \geq 2400$ reaches $C_5 \in [0.41, 0.49]$; Fourier-CMA at $K = 40, n = 2400$ plateaus at ~ 0.4045. By hour 13 the swarm has eliminated restart-from-random and Fourier-CMA as productive directions.

Phase 2, basin chain (hours 13–24). Diagnosed problem. The remaining alternative is a large-σ Gaussian shake; only one agent has tried it, and only at $\sigma = 0.05$. **Rationale.** If the warm-start basin's attractor radius is small but the basin landscape is rich at larger perturbation scales, increasing the shake amplitude could identify new local minima; once any agent finds a new basin, the orchestrator can amplify the recipe to the swarm. **Mechanism.** At hour 13, one agent runs a Gaussian shake at $\sigma = 0.10$ on the warm-start vector, polishes with SLP, and discovers a sub-basin at $C_5 = 0.380867$ that is structurally distinct from the warm-start basin ($L_2 = 1.10$, $402/600$ coordinates differ, active-lag set differs in $252/437$ entries). The orchestrator broadcasts the recipe and recommends heavier shake at $\sigma \in [0.15, 0.30]$ from the new basin. **Observed effect.** Within the next hour, six independent agents reproduce the recipe. The chain descends in discrete steps from 0.380870 to 0.380863, each step driven by a different agent applying a larger-σ shake on the previous floor.

Phase 3, slow descent (hours 24–119). Diagnosed problem. The basin chain saturates at $C_5 \approx 0.380863$, 1.3×10^{-5} below the prior public best; additional shake iterations at σ up to 0.50 yield no new basins. **Rationale.** The shake-and-polish recipe has exhausted the local basin landscape; the swarm needs a qualitatively different construction family. **Mechanism.** The orchestrator partitions the swarm across (i) high-precision Fourier-CMA at $K \in [40, 60]$, (ii) a dual / Lagrangian formulation against White's spectral lower bound, (iii) deep local search with `trust-constr` from the basin floor, (iv) a genetic algorithm on the discrete h-vector. At hour ~ 28 the host evaluator goes silent for 7.5 hours due to an infrastructure issue; the supervisor keeps all 17 solver processes alive while the orchestrator marks candidates from that interval as unverified and re-issues verification once the host recovers. **Observed effect.** None of the four alternative parameterizations escapes the basin within the wall-clock budget. The run ends with swarm-best $C_5 = 0.380863$, advancing the prior public best by 1.3×10^{-5}.

What the trajectory shows. Every step-down on the trajectory follows a single orchestrator broadcast within minutes, and each broadcast canonicalizes a diagnosis (basin-locked, larger-σ escape, alternative parameterizations) that no individual agent derives from its own state. The supervisor sustains multi-day continuity through a 7.5-hour host-evaluator outage without derailing the run.

6 Limitations

FOUNDRY's guarantees rest on a host-owned evaluator that is faithful to the objective, together with a per-problem adapter that supplies the schema, trusted evaluator, and prompt builders for each new domain; problems without a fast and trustworthy evaluator therefore lie outside its present scope. The promotion rule and prompt design rely on empirical choices we have not formally characterized, and all reported results use frozen frontier coding agents at substantial compute cost. Whether host-owned trust and memory help smaller models close this gap is a question for future work.

7 Conclusion

We argued that the bottleneck in long-horizon agentic discovery is structural, not capability-bound: when agents are allowed to own the trusted score and the cross-attempt memory, the trust boundary collapses and progress fails to compound. FOUNDRY takes the opposite stance. Trust and memory are host-owned invariants: the host re-derives every accepted score from an evaluator the agent cannot modify, and curates a confidence-weighted registry that promotes a hypothesis only after independent corroboration. The same coordinator, registry, and hypervisor $\rightarrow$ orchestrator $\rightarrow$ solver hierarchy drive three unrelated benchmarks without per-domain code changes.

Under this single substrate, frozen frontier coding agents tighten the upper bound on the Erdős minimum-overlap constant to $C_5 = 0.380863$, achieve a $989\,\mu$s geometric-mean runtime on the GPUMode TriMul kernel on H100, and lower the mean-squared error on OpenProblems Denoising. Together, these results suggest that progress in agentic discovery depends not only on stronger proposal generators, but on the system boundary that separates untrusted proposal from trusted verification and memory.

References

Joshua Batson, Loic Royer, and James Webber. Molecular cross-validation for single-cell rna-seq. *BioRxiv*, page 786269, 2019.

Gerardo Beni and Jing Wang. Swarm intelligence in cellular robotic systems. In *Robots and biological systems: towards a new bionics?*, pages 703–712. Springer, 1993.

Shiyi Cao, Ziming Mao, Joseph E. Gonzalez, and Ion Stoica. K-search: Llm kernel generation via co-evolving intrinsic world model, 2026. URL https://arxiv.org/abs/2602.19128.

Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis, and Pushmeet Kohli. Discovering faster matrix

multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022. doi: 10.1038/s41586-022-05172-4. URL https://doi.org/10.1038/s41586-022-05172-4.

Jan Kristian Haugland. The minimum overlap problem revisited, 2016. URL https://arxiv.org/abs/1609.08000.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework, 2024. URL https://arxiv.org/abs/2308.00352.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021. doi: 10.1038/s41586-021-03819-2. URL https://doi.org/10.1038/s41586-021-03819-2.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. Shinkaevolve: Towards open-ended and sample-efficient program evolution, 2025. URL https://arxiv.org/abs/2509.19349.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society, 2023. URL https://arxiv.org/abs/2303.17760.

George C Linderman, Jun Zhao, Manolis Roulis, Piotr Bielecki, Richard A Flavell, Boaz Nadler, and Yuval Kluger. Zero-preserving imputation of single-cell rna-seq data. *Nature communications*, 13 (1):192, 2022.

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*, pages 1–6, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

Daniel J. Mankowitz, Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, Edouard Leurent, Shariq Iqbal, Jean-Baptiste Lespiau, Alex Ahern, Thomas Köppe, Kevin Millikin, Stephen Gaffney, Sophie Elster, Jackson Broshear, Chris Gamble, Kieran Milan, Robert Tung, Minjae Hwang, Taylan Cemgil, Mohammadamin Barekatain, Yujia Li, Amol Mandhane, Thomas Hubert, Julian Schrittwieser, Demis Hassabis, Pushmeet Kohli, Martin Riedmiller, Oriol Vinyals, and David Silver. Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, 618(7964):257–263, 2023. doi: 10.1038/s41586-023-06004-9. URL https://doi.org/10.1038/s41586-023-06004-9.

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

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2024. URL https://arxiv.org/abs/2310.08560.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. Chatdev: Communicative agents for software development, 2024. URL https://arxiv.org/abs/2307.07924.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/s41586-023-06924-6. URL https://doi.org/10.1038/s41586-023-06924-6.

Asankhaya Sharma. Openevolve: an open-source evolutionary coding agent. https://github.com/algorithmicsuperintelligence/openevolve, 2025. URL https://github.com/algorithmicsuperintelligence/openevolve.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Anja Surina, Amin Mansouri, Lars Quaedvlieg, Amal Seddas, Maryna Viazovska, Emmanuel Abbe, and Caglar Gulcehre. Algorithm discovery with llms: Evolutionary search meets reinforcement learning, 2025. URL https://arxiv.org/abs/2504.05108.

David Van Dijk, Roshan Sharma, Juozas Nainys, Kristina Yim, Pooja Kathail, Ambrose J Carr, Cassandra Burdziak, Kevin R Moon, Christine L Chaffer, Diwakar Pattabiraman, et al. Recovering gene interactions from single-cell data using data diffusion. *Cell*, 174(3):716–729, 2018.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for ai software developers as generalist agents, 2025a. URL https://arxiv.org/abs/2407.16741.

Yiping Wang, Shao-Rong Su, Zhiyuan Zeng, Eva Xu, Liliang Ren, Xinyu Yang, Zeyi Huang, Xuehai He, Luyao Ma, Baolin Peng, Hao Cheng, Pengcheng He, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. Thetaevolve: Test-time learning on open problems, 2025b. URL https://arxiv.org/abs/2511.23473.

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

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation, 2023. URL https://arxiv.org/abs/2308.08155.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL https://arxiv.org/abs/2305.10601.

Mert Yuksekgonul, Daniel Koceja, Xinhao Li, Federico Bianchi, Jed McCaleb, Xiaolong Wang, Jan Kautz, Yejin Choi, James Zou, Carlos Guestrin, and Yu Sun. Learning to discover at test time, 2026. URL https://arxiv.org/abs/2601.16175.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models, 2026. URL https://arxiv.org/abs/2510.04618.

NeurIPS Paper Checklist

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 state the four contributions (host-owned trust boundary, the FOUNDRY metaharness, the established-facts registry, and three host-verified benchmark results); each is supported by Section 3, Section 4, and 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: Section 6 discusses the scope of FOUNDRY's guarantees, the empirical design choices in the promotion rule and prompts, and the compute cost of frontier-model swarms.

   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: The paper does not include formal theoretical results. Equation 1 and Equation 2 are operational definitions of the host-side accept rule and the registry promotion rule, not theorems.

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: Algorithm 1 specifies the discovery loop, Section 3 defines the per-problem adapter, and the Environment paragraphs of Section 4 specify the state, action, dynamics, and trusted reward for each benchmark.

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: [No]

   Justification: Code is not released with the submission. The algorithmic content (Section 3), the per-problem adapter specifications (Section 3), and the discovered constructions and kernels (Section 4) are described in sufficient detail to reproduce the main results.

   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: Each Environment paragraph in Section 4 specifies the state, action, dynamics, trusted reward, and constraint validators; the case study in Section 5 provides the coordination details for one long-horizon run.

   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: [Yes]

   Justification: Table 2 reports per-comparator standard deviations across outer iterations of the upstream evaluator on identical hardware. The Erdős and OpenProblems Denoising scores are deterministic functions of a fixed discovered artifact and have no run-to-run variability for a given construction.

   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: Section 5 reports the wall-clock duration and solver count for the Erdős long-horizon run; Section 4.2 specifies the H100 hardware used for kernel evaluation; Section 6 acknowledges the overall compute footprint of frontier-agent swarms.

   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: The work does not involve human subjects, sensitive data, or scraped content, and conforms with the NeurIPS Code of Ethics.

   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: [N/A]

Justification: FOUNDRY is a host-side coordination substrate for frozen LLM coding agents on well-defined discovery benchmarks. It does not introduce new model capabilities or release new models, and we do not see a direct application path to negative outcomes beyond the dual-use considerations general to stronger autonomous coding agents.

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: No high-risk model, dataset, or scraped resource is released with the submission.

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: All baselines and benchmarks are credited to their original publications in Section 2 and Section 4; the upstream TriMul evaluator is the unmodified GPUMode

reference suite. FOUNDRY's solver and orchestrator roles are instantiated using closed-source frontier coding agents (Claude Code, Codex) under their providers' terms of use.

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: [N/A]

Justification: No new assets are released with the submission.

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: The work does not involve crowdsourcing or research with human subjects.

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: The work does not involve research with human subjects.

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: [N/A]

Justification: LLMs were used only for editing during preparation of the manuscript.

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.