GÖDEL FOREST: BALANCING SEARCH DEPTH AND BREADTH FOR DATA-CENTRIC RECURSIVE SELF-IMPROVEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Recursive self-improvement (RSI) aims to achieve compounding gains by having models improve themselves. While most existing RSI systems optimize external agent harnesses or prompts around a frozen base model, *data-centric RSI* directly updates the model's own parameters by training on agent-generated data. However, because validating data strategies requires expensive model training, existing methods face a fundamental dilemma: a single agent gets trapped in narrow directions and lacks exploration breadth, while naive parallel search or heavy trace sharing sacrifices long-horizon search depth. To address this challenge, we introduce **Gödel Forest**, a multi-agent framework that organizes recursive self-improvement as an ensemble of co-evolving search trees. In Gödel Forest, each agent autonomously grows a persistent tree, deepening, branching, or pruning data strategies based on model feedback to secure depth, while parallel trees explore distinct regions of the data space to expand breadth. Crucially, rather than leaving trees isolated or flooding them with heavy execution logs, a dynamically co-evolving memory connects the forest: agents continuously distill their successes and failures into compact procedural lessons anchored to a global leaderboard. Through this forest ecosystem, a dead-end in one tree instantly warns the whole forest against unpromising paths, while an empirical breakthrough quickly seeds new exploration branches in neighboring trees. Evaluated on RSIBench-Data across six diverse domains, Gödel Forest outperforms the single-agent baseline by an average of 10.70% while reducing wall-clock time on five tasks. Ablations confirm that co-evolving shared memory yields a +7.00% gain over independent parallel search, demonstrating that collective distillation is key to scalable self-improvement. The code is available at `https://anonymous.4open.science/r/godel-forest-review/`.

1 INTRODUCTION

Recursive self-improvement (RSI) refers to a process in which a system repeatedly produces improved versions of itself under a given evaluator, thereby improving its own ability to improve and yielding compounding gains (Li et al., 2026a;b; Yu et al., 2026; Zhao et al., 2026b). The original Gödel machine casts RSI as a proof-guided search over self-modifications, accepting a candidate rewrite only after proving that it increases expected utility. Because this proof search is intractable for realistic agents, recent empirical methods (Robeyns et al., 2025; Zhang et al., 2026a; Wang et al., 2026) replace formal proof with validation based on observed downstream task utility. Recent extensions such as the Mendel Gödel Machine (MGM)(Liu et al., 2026) further enrich this loop by contrasting execution traces across tasks and parallel runs to evolve executable agent harnesses.

However, RSI systems remain confined to the external harness or prompts around a frozen base model, rather than entering the model itself through parameter updates. Data-centric RSI offers a complementary route in which an LLM agent repeatedly proposes data and training strategies, trains candidate target models from fixed initial parameters, and evaluates them on downstream tasks (Meng et al., 2026; Kulikov et al., 2026; Rank et al., 2026; Yano et al., 2025; Kessler et al., 2025). The resulting evidence updates the agent's persistent search state and guides subsequent

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

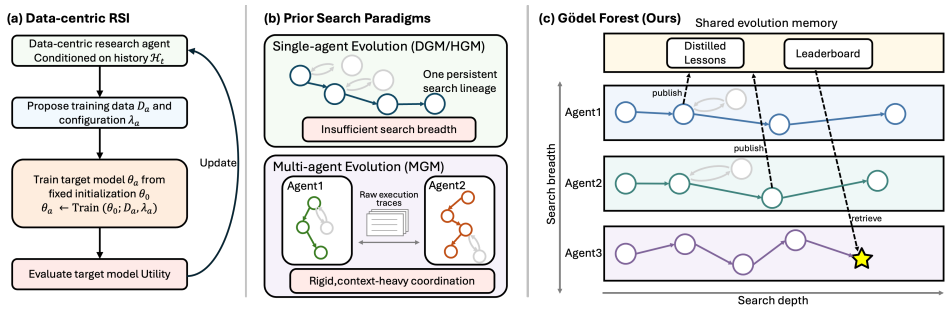


Figure 1: Comparison of search paradigms in recursive self-improvement (RSI).

proposals. Thus, training changes the parameters of each candidate target model, whereas recursion lies in the agent's evidence-driven evolution of data and training strategies (Figure 1a).

Unlike agent harness optimization, data evolution navigates an explosive combinatorial space spanning individual sample synthesis and dataset-level distribution shifts. Crucially, validating each candidate hypothesis incurs full-loop model training and downstream evaluation, rendering naive exploration prohibitively expensive under constrained compute budgets. This training-in-the-loop bottleneck traps existing RSI paradigms in a stark dilemma (Figure 1b): single-agent frameworks (Zhang et al., 2026a; Wang et al., 2026) easily stagnate in myopic, narrow trajectories lacking directional breadth, whereas multi-agent variants (Liu et al., 2026) rely on raw trace exchange across identical tasks, resulting in rigid, context-saturated coordination that fails to scale to open-ended data discovery. How to sustain deep, long-horizon hypothesis pursuit while unlocking expansive exploration breadth through lightweight coordination remains the central challenge of data-centric RSI.

To resolve this, we propose **Gödel Forest** (Figure 1c), a multi-agent framework that structures RSI as an ensemble of co-evolving search trees. Gödel Forest reconciles the depth-versus-breadth tradeoff through a structured duality: locally, each agent autonomously cultivates a deep, persistent search tree, using target-model feedback to continuously deepen, refine, or backtrack its data strategies to secure long-horizon search depth; globally, parallel trees explore distinct regions of the combinatorial data space, opening expansive directional breadth without discarding their accumulated search context. Crucially, rather than leaving trees isolated or choking communication with raw execution traces, agents coordinate asynchronously via a dynamically co-evolving shared memory through a read-evolve-publish protocol. By distilling empirical breakthroughs and failure modes into compact, actionable data lessons anchored to a global leaderboard, Gödel Forest establishes a collective feedback loop: a dead-end hit by one tree instantly acts as an inductive guardrail across the forest, while an empirical leap swiftly seeds new exploration branches in neighboring trees.

We evaluate Gödel Forest on RSIBench-Data across six diverse benchmarks spanning repository-level software engineering (SWE-bench Verified, Multilingual, and Pro), long-horizon terminal interaction (Terminal-Bench 2.0), scientific reasoning (GPQA Diamond), and competition mathematics (AIME 2026). Across all tasks, Gödel Forest consistently outperforms the single-agent baseline, achieving an average gain of 10.70% (e.g., boosting SWE-bench Verified from 35.00% to 54.00%) while reducing end-to-end wall-clock time on five of the six benchmarks. Ablations further confirm that co-evolving via shared evolution memory provides a +7.00% gain over running independent parallel agents, demonstrating that dynamic experience sharing is key to scalable self-improvement.

The contributions of this paper are as follows:

- We formulate data-centric RSI as an open-ended data-evolution search, identifying the fundamental tension between long-horizon search depth and directional exploration breadth under expensive training-in-the-loop validation.
- We propose **G"odel Forest**, a co-evolutionary multi-agent framework that structures recursive improvement as an ensemble of persistent search trees, coupling private long-horizon tree refinement with an asynchronous read-evolve-publish shared memory for lightweight coordination.
- We evaluate **Gödel Forest** across six diverse benchmarks in RSIBench-Data, demonstrating consistent gains over single-agent and independent parallel baselines, superior wall-clock efficiency, and significant empirical advantages from shared-memory co-evolution.

2 Related Work

Empirical Gödel Machines and Harness Evolution. Originating from Schmidhuber's theoretical vision (Schmidhuber, 2006), empirical Gödel machines replace intractable formal proof searches with observed task utility to iteratively rewrite agent programs (Robeyns et al., 2025; Yin et al., 2025). Subsequent efforts generalize this paradigm through archive-based exploration (Zhang et al., 2026a; Wang et al., 2026), risk-controlled acceptance (Wu et al., 2025), evaluator co-evolution (Zhang et al., 2026b; Iacob et al., 2026), and multi-trajectory comparative evolution (Liu et al., 2026). Despite these advances, existing empirical Gödel systems operate under a shared constraint: their self-modifications are strictly restricted to external executable harnesses or prompt wrappers around frozen base models, precluding internal parameter updates. Furthermore, exchanging verbose execution traces induces severe context bloat, preventing effective scaling in open-ended research spaces.

Data-Centric Recursive Self-Improvement. A parallel line of research pursues compounding gains by directly updating model parameters. Early frameworks bootstrap reasoning capabilities from self-generated rationales (Zelikman et al., 2022; Chen et al., 2024; Yuan et al., 2024) or zero-data environment self-play (Zhao et al., 2026a; Huang et al., 2026). Building on interactive environments (Khan et al., 2025; Kessler et al., 2025), recent post-training frameworks delegate dataset curation, recipe synthesis, and execution monitoring to autonomous LLM agents (Yano et al., 2025; Li et al., 2026b; Meng et al., 2026). However, existing data-centric RSI systems predominantly rely on single-agent search that easily gets trapped in localized data distributions under constrained budgets, while uncoordinated parallel sampling repeatedly wastes compute on redundant failure modes.

Unlike prior work confined to prompt-level harnesses or uncoordinated sampling, we establish *asynchronous memory co-evolution* as the coordination paradigm for data-centric RSI. Gödel Forest realizes this through a search duality: private search trees maintain local selective pressure for deep hypothesis refinement, while a shared memory substrate prevents redundant exploration and propagates breakthrough recipes across workers. Grounded in real parameter updates, Gödel Forest ultimately bridges empirical Gödel search with scalable post-training collective intelligence.

3 Gödel Forest

To overcome the training-in-the-loop bottleneck in data-centric recursive self-improvement (RSI), we present **Gödel Forest**, a co-evolutionary multi-agent framework that structures search as an ensemble of persistent search trees coupled through a dynamically shared evolution memory. Gödel Forest resolves the fundamental tension between search depth and exploration breadth through a structured duality: locally, individual agents cultivate deep, persistent search trees with budget-aware diagnostic gating; globally, parallel trees explore distinct regions of the combinatorial data space and coordinate asynchronously via compact, distilled empirical lessons.

3.1 Problem Formulation and Single-Tree Foundations

Empirical Gödel search as tree search. The original Gödel machine (Schmidhuber, 2006) executes self-modifications only after formally proving that they yield higher expected utility than continuing the search. Because formal proof search is intractable for complex generative agents, empirical Gödel frameworks guide exploration using observed downstream utility over an archive of agent versions. Let $\mathcal{T}_t$ denote the agent archive at step t, initialized with a seed agent as $\mathcal{T}_0 = \{a_0\}$. Each non-root agent retains the identity of the parent from which it was derived, forming a directed tree. A search policy selects an archived agent $a \in \mathcal{T}_t$ and applies a modification operator $a' \leftarrow \Phi(a, \mathcal{H}_t)$ conditioned on the accumulated evaluation evidence $\mathcal{H}_t$. The successor a' is appended to the archive as a child of a, giving $\mathcal{T}_{t+1} = \mathcal{T}_t \cup \{a'\}$. Because any archived node can be selected for modification, the search can either deepen a promising branch or backtrack to an earlier node to pursue an alternative direction.

Data-centric recursive self-improvement. Unlike prior empirical Gödel systems that optimize external prompt templates or agent harnesses around frozen base models, data-centric RSI seeks compounding gains by directly updating target model parameters. Here, each archived agent a is a

specialized data-engineering program. Executing a synthesizes training data D_a, trains candidate parameters θ_a from a fixed base model θ_0, and evaluates downstream utility $U(a)$ on dataset $\mathcal{B}$:

$$(D_a, \lambda_a) = \text{Generate}(a), \tag{1}$$

$$\theta_a = \text{Train}(\theta_0; D_a, \lambda_a), \tag{2}$$

$$U(a) = \text{Eval}_{\mathcal{B}}(\theta_a). \tag{3}$$

Although every candidate model is trained from the same initial parameters θ_0, the process remains strictly recursive: the empirical performance and failure diagnostics of earlier target models update the agent's persistent state, which in turn governs subsequent data generation and curation decisions.

The depth-versus-breadth dilemma. Validating each data hypothesis requires full-loop model training and downstream evaluation, rendering naive exploration prohibitively expensive. This training-in-the-loop bottleneck creates a stark dilemma: a single tree easily stagnates in path-dependent trajectories lacking directional breadth, whereas parallel search either redundantly repeats shared failures or chokes on verbose trace exchange (Liu et al., 2026). Reconciling deep, multi-round hypothesis pursuit with broad, lightweight collective coordination is the central objective of Gödel Forest.

3.2 Persistent Tree Search: Local Deep Hypothesis Pursuit

To enable long-horizon hypothesis pursuit without premature abandonment, each worker $i \in \{1, \ldots, K\}$ maintains a private search tree $\mathcal{T}_t^{(i)}$ and evidence ledger $\mathcal{H}_t^{(i)}$. Each worker LLM autonomously selects parents, designs data and training configurations, and makes other search decisions under fixed interfaces and budgets. Within each tree, the worker executes a skill-enhanced cognitive loop comprising four distinct stages:

Stage 1: Diagnose. The worker analyzes the failure modes of previous attempts in $\mathcal{H}_t^{(i)}$ (e.g., runaway completion, ungrounded actions, or skipped verification) to localize specific behavioral defects in the candidate model.

Stage 2: Propose. Conditioned on both its private trajectory $\mathcal{H}_t^{(i)}$ and the globally shared memory M_t, the worker selects an existing parent agent $a \in \mathcal{T}_t^{(i)}$ and generates an evolved successor program a' (realized as executable data synthesis and curation code):

$$a' \leftarrow \Phi(a, \mathcal{H}_t^{(i)}, M_t). \tag{4}$$

The successor a' is archived in $\mathcal{T}_t^{(i)}$, executing $(D_{a'}, \lambda_{a'}) = \text{Generate}(a')$ followed by candidate model training $\theta_{a'} = \text{Train}(\theta_0; D_{a'}, \lambda_{a'})$.

Stage 3: Validate & Evaluate. Full-benchmark evaluation on long-horizon tasks is computationally demanding. To preserve the evaluation budget, the worker first validates the generated data and evaluates the candidate on a lightweight diagnostic subset $\mathcal{B}^{\text{diag}} \subset \mathcal{B}$, whose composition is determined autonomously by the worker:

$$U_{\text{diag}}(a') = \text{Eval}_{\mathcal{B}^{\text{diag}}}(\theta_{a'}). \tag{5}$$

If $U_{\text{diag}}(a')$ surpasses the gating threshold, the worker proceeds to evaluate the remaining tasks in $\mathcal{B}$ using the same checkpoint $\theta_{a'}$ without retraining, obtaining the complete utility $U(a')$. Otherwise, evaluation is terminated early, pruning unviable candidates before incurring full-set overhead while recording the diagnostic evidence.

Stage 4: Review. The worker appends the evaluation outcome to $\mathcal{H}_t^{(i)}$ and determines the next search action: exploiting the current direction through further modification, revising hyperparameters, or backtracking to an earlier ancestor node.

3.3 Forest Co-Evolution via Shared Evolution Memory

To reconcile independent exploration with collective synergy, Gödel Forest organizes K concurrent workers into a co-evolving ensemble $\mathcal{F}_t = \{\mathcal{T}_t^{(i)}\}_{i=1}^{K}$. Workers interact asynchronously through a shared evolution memory $M_t = (\Lambda_t, \mathcal{L}_t)$ that acts as a non-blocking public blackboard.

Algorithm 1 Gödel Forest Asynchronous Co-Evolution Protocol

Initialize: Archives $\mathcal{T}_0^{(i)} \leftarrow \{a_0\}$, evidence $\mathcal{H}_0^{(i)} \leftarrow \emptyset$ for $i \in \{1, \ldots, K\}$, shared memory $M_0 \leftarrow (\emptyset, \emptyset)$.

while elapsed wall-clock $W \leq W_{\max}$ **and** cumulative expenditure $C \leq C_{\max}$ **do**
  for each worker $i \in \{1, \ldots, K\}$ **in parallel, asynchronously do**
    $\mathcal{L}_{\text{rel}}^{(i)} \leftarrow \text{Retrieve}(\mathcal{L}_t, \mathcal{H}_t^{(i)})$ {**Read**: Query relevant shared lessons}
    Select parent $a \in \mathcal{T}_t^{(i)}$ and propose $a' \leftarrow \Phi(a, \mathcal{H}_t^{(i)}, \mathcal{L}_{\text{rel}}^{(i)}, \Lambda_t)$ {**Propose**}
    $(D_{a'}, \lambda_{a'}) \leftarrow \text{Generate}(a')$; $\theta_{a'} \leftarrow \text{Train}(\theta_0; D_{a'}, \lambda_{a'})$ {**Train**}
    $U_{\text{diag}}(a') \leftarrow \text{Eval}_{\mathcal{B}^{\text{diag}}}(\theta_{a'})$ {**Diagnostic Screening**}
    if $U_{\text{diag}}(a')$ passes gating threshold **then**
      $U_{\text{rem}}(a') \leftarrow \text{Eval}_{\mathcal{B}\setminus\mathcal{B}^{\text{diag}}}(\theta_{a'})$; $U(a') \leftarrow \text{Merge}(U_{\text{diag}}(a'), U_{\text{rem}}(a'))$
      $\Lambda_t \leftarrow \Lambda_t \cup \{(a', U(a'))\}$ {Full eval & leaderboard update}
    end if
    Update private evidence $\mathcal{H}_{t+1}^{(i)}$; Distill lesson $\ell_{a'}^{(i)} \leftarrow \text{Synthesize}(\mathcal{H}_{t+1}^{(i)})$ {**Review**}
    if $\ell_{a'}^{(i)}$ contains non-trivial empirical findings **then**
      $\mathcal{L}_{t+1} \leftarrow \mathcal{L}_t \cup \{\ell_{a'}^{(i)}\}$ {**Publish**: Broadcast lesson to blackboard}
    end if
  end for
end while
return Optimal agent $a^\star = \arg\max_{a \in \Lambda_{t_{\text{end}}}} U(a)$ with target checkpoint $\theta_{a^\star}$.

Outcome-level anchor: shared leaderboard Λ_t. The leaderboard aggregates verified utilities across all fully evaluated agents across workers, $\Lambda_t = \{(a, U(a)) \mid a \in \bigcup_{j=1}^{K} \mathcal{T}_t^{(j)},\ a \text{ fully evaluated}\}$, establishing a transparent global performance anchor for each worker to gauge whether its local branch remains competitive against the collective frontier.

Procedural cognition: structured empirical lessons $\mathcal{L}_t$. While numeric scores indicate performance, cross-tree coordination requires understanding *why* a data intervention succeeded or failed. Instead of broadcasting raw data samples or execution traces, worker i synthesizes each significant trial into a compact, structured lesson ℓ:

$$\ell = \big\langle \mathcal{H}_{\text{intent}},\ \Delta U,\ \text{Attribution},\ \rho \big\rangle, \tag{6}$$

where $\mathcal{H}_{\text{intent}}$ specifies the tested data hypothesis (e.g., all-turn vs. terminal-turn supervision), ΔU records the empirical utility shift, Attribution denotes the causal diagnosis of the outcome, and ρ articulates a distilled, actionable operational rule (e.g., "restrict supervision loss to final assistant turns to prevent run-on completion"). This structured schema decouples high-level procedural takeaways from heavy execution logs, keeping cross-tree communication lightweight ($\sim$100 tokens per lesson) and circumventing the severe context bloat that impedes raw-trace sharing (Liu et al., 2026). The published lessons dynamically form an asynchronous shared blackboard: $\mathcal{L}_{t+1} = \mathcal{L}_t \cup \{\ell_a^{(i)}\}$.

Asynchronous Read-Evolve-Publish loop. Communication in Gödel Forest is entirely pull-based and non-blocking, operating without global synchronization barriers:

- **Read**: Prior to formulating a new candidate, worker i retrieves contextually relevant lessons based on its current failure symptoms: $\mathcal{L}_{\text{rel}}^{(i)} = \text{Retrieve}(\mathcal{L}_t, \mathcal{H}_t^{(i)}) \subseteq \mathcal{L}_t$.
- **Evolve**: The retrieved lessons fulfill two complementary functions: negative lessons serve as *inductive guardrails*, pruning paths that caused failure in peer trees; positive lessons act as *catalytic seeds*, inspiring peer workers to explore creative variants in distinct domains.
- **Publish**: Upon concluding its Review stage, the worker broadcasts any distilled lesson to $\mathcal{L}_t$, immediately enriching the collective memory for all subsequent queries.

Algorithm 1 summarizes the complete Gödel Forest co-evolutionary protocol. Upon reaching budget termination, the manager returns the globally optimal checkpoint $\theta_{a^\star}$ satisfying $a^\star = \arg\max_{a \in \Lambda} U(a)$. This architecture ensures that individual trees maintain deep, uninterrupted search continuity, while shared memory continuously drives collective discovery across the forest.

Table 1: Official performance and resource use with Claude Code + `Sonnet-5`. Time is end-to-end wall-clock hours for data generation, training, validation, and review; cost is cumulative TINKER expenditure across all workers. Here the official scores S_{off} are reported.

Benchmark	Method	Score ↑	Time (h) ↓	Cost ($) ↓
SWE-bench Verified	Vanilla	35.00%	14.91	**181.70**
	Gödel Forest	**54.00%**	**14.06**	239.67
SWE-bench Multilingual	Vanilla	22.00%	14.20	363.77
	Gödel Forest	**30.00%**	**11.02**	**336.55**
SWE-bench Pro	Vanilla	4.00%	9.54	**45.63**
	Gödel Forest	**15.00%**	**7.49**	419.87
Terminal-Bench 2.0	Vanilla	5.62%	**8.87**	**156.93**
	Gödel Forest	**17.98%**	15.50	295.06
GPQA Diamond	Vanilla	52.00%	6.43	16.03
	Gödel Forest	**55.00%**	**5.76**	**15.22**
AIME 2026	Vanilla	49.17%	9.29	121.21
	Gödel Forest	**60.00%**	**8.61**	**49.94**

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Benchmarks and task interface. We evaluate Gödel Forest on RSIBench-Data (Meng et al., 2026) across six diverse, high-difficulty benchmarks: SWE-bench Verified (Jimenez et al., 2024), SWE-bench Multilingual (Zan et al., 2026), SWE-bench Pro (Deng et al., 2025), Terminal-Bench 2.0 (Merrill et al., 2026), GPQA Diamond (Rein et al., 2023), and AIME 2026. Together, these cover repository-level software engineering, long-horizon interactive bash environments, multidisciplinary scientific reasoning, and competition mathematics. The three SWE-bench suites use the Mini-SWE-Agent runner (Yang et al., 2024). All candidates are evaluated by Harbor in isolated E2B sandboxes under identical interfaces.

Models, training infrastructure, and budgets. Every candidate starts from the base model `Qwen3.5-35B-A3B-Base` (Yang et al., 2025) and is fine-tuned with LoRA SFT through the shared TINKER backend; Claude Opus 4.8 serves as the fixed external rollout generator. Each run is strictly governed by a 16-hour wall-clock limit and a $500 cumulative TINKER budget, enforced across the entire forest.

Official evaluation and baselines. During search, $U(a)$ denotes feedback from the RSIBench-Data selection evaluator, which guides data generation and checkpoint selection without exposing gold solutions. Post-search, the selected checkpoint $\theta_{a^\star}$ is frozen and officially evaluated in an isolated canonical run, yielding $S_{\text{off}} = \text{Eval}^{\text{off}}(\theta_{a^\star})$. Because both phases share task identities, S_{off} serves as a standardized post-selection verification rather than a held-out generalization benchmark. Our primary baseline is vanilla Claude Code[1] with `Sonnet-5` operating as a single-agent search without forest coordination or shared memory. To demonstrate framework generality across underlying LLM backbones, we additionally instantiate Gödel Forest with OpenAI Codex[2] (`gpt-5.6-sol`). Claude Code runs at high reasoning effort and Codex at maximum reasoning effort, following RSIBench-Data standards.

4.2 MAIN RESULTS

Table 1 presents the official benchmark performance, end-to-end wall-clock time, and cumulative TINKER expenditure for Gödel Forest ($K = 3$) versus the vanilla Claude Code baseline.

Universal gains across diverse scenarios. Gödel Forest consistently outperforms the single-agent baseline on all six benchmarks, yielding an average absolute gain of 10.70%. The most pronounced

[1] `https://github.com/anthropics/claude-code`

[2] `https://github.com/openai/codex`

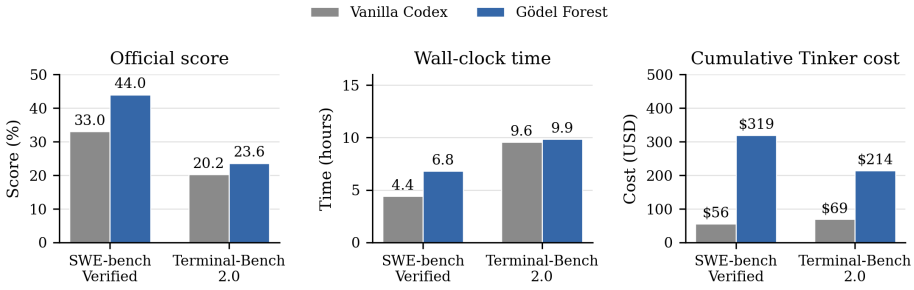


Figure 2: Cross-backbone generalization with Codex (`gpt-5.6-sol`). Left: official score. Center: wall-clock time. Right: cumulative TINKER expenditure.

improvement emerges on SWE-bench Verified, soaring from 35.00% to 54.00% (+19.00%). Substantial gains also extend across challenging executable domains (SWE-bench Pro improves from 4.00% to 15.00%; Terminal-Bench 2.0 improves from 5.62% to 17.98%) and complex reasoning tasks (GPQA Diamond reaches 55.00%; AIME 2026 reaches 60.00%). These comprehensive improvements confirm that co-evolving persistent search trees consistently unlocks superior data synthesis strategies across heterogeneous problem distributions.

The efficiency paradox: faster completion via asynchronous synergy. Despite conducting broader and deeper multi-tree exploration, Gödel Forest achieves *lower* end-to-end wall-clock time on five of the six benchmarks (e.g., reducing SWE-bench Multilingual from 14.20h to 11.02h, and SWE-bench Pro from 9.54h to 7.49h). This efficiency stems directly from two algorithmic properties: asynchronous parallel execution maximizes hardware concurrency, while shared evolution memory rapidly propagates negative findings, enabling workers to prune unviable exploration branches before committing expensive training cycles.

Understanding inference expenditure. While wall-clock time decreases, cumulative expenditure is higher on long-horizon agentic benchmarks (e.g., SWE-bench Pro and Terminal-Bench 2.0). This expenditure disparity is an artifact of candidate capability rather than search inefficiency: weaker baseline models frequently crash, loop, or terminate prematurely during early turns, truncating their interaction trajectories. In contrast, models evolved by Gödel Forest sustain coherent, multi-turn reasoning and tool use, executing complex actions that incur higher inference cost but successfully solve challenging tasks.

4.3 CROSS-BACKBONE GENERALIZATION: RESULTS WITH CODEX

To confirm that Gödel Forest functions as a general meta-architecture independent of any specific agent implementation, we evaluate worker agents driven by Codex (`gpt-5.6-sol`) on SWE-bench Verified and Terminal-Bench 2.0 (Figure 2). With target model, training backend, and evaluation interfaces held fixed, Gödel Forest secures marked improvements over the vanilla Codex baseline: official scores rise from 33.00% to 44.00% (+11.00%) on SWE-bench Verified and from 20.22% to 23.60% (+3.38%) on Terminal-Bench 2.0. Furthermore, elapsed wall-clock latency remains modest (6.81h vs. 4.41h on SWE-bench Verified, and 9.85h vs. 9.56h on Terminal-Bench 2.0). As in Section 4.2, the higher expenditure ($319.22 vs. $55.61) reflects the target model's enhanced ability to sustain extended, successful problem-solving interactions. These consistent gains demonstrate that the benefits of asynchronous tree co-evolution generalize across distinct frontier LLMs.

4.4 ABLATION STUDY: DISSECTING EXPLORATION BREADTH AND COORDINATION

We conduct an ablation study on SWE-bench Verified to isolate the contributions of tree concurrency and shared memory coordination (Figure 3).

Concurrency scaling and the depth-breadth trade-off. Scaling the number of persistent workers from $K = 1$ to $K = 3$ broadens initial exploration diversity, steadily elevating official performance from 45.00% to 51.00% and $\mathbf{54.00}\%$. Here, the single-worker setting ($K = 1$) isolates our local persistent tree search (§3.2) from forest coordination. Unlike the vanilla baseline operating

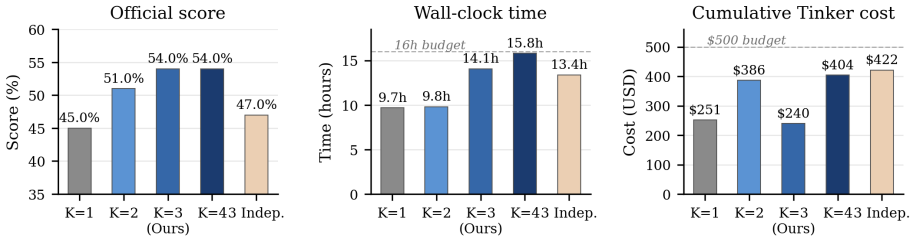


Figure 3: Ablation study on SWE-bench Verified. We evaluate worker scaling ($K \in \{1, 2, 3, 4\}$) with full shared memory, and compare against 3 Independent Workers without shared memory. Dashed lines denote 16-hour and $500 budget limits.

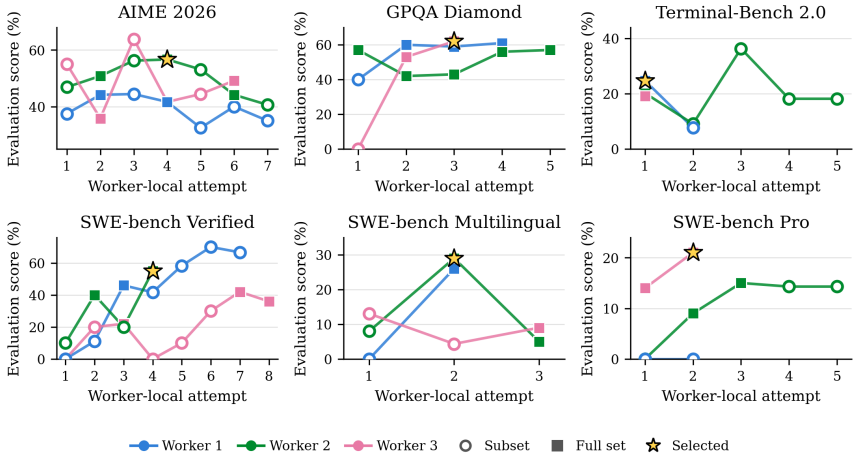


Figure 4: Candidate search trajectories across six benchmarks. Colors trace independent trees maintained by the three workers ($K = 3$). Open circles denote diagnostic evaluations ($\mathcal{B}^{\text{diag}}$); filled squares denote full benchmark evaluations ($\mathcal{B}$). Yellow stars mark the selected checkpoints.

over an unconstrained flat conversational loop, a worker in our framework incorporates a structured evidence ledger $\mathcal{H}$ for systematic backtracking, cognitive stage skills for grounded candidate proposal, and diagnostic gating ($\mathcal{B}^{\text{diag}} \rightarrow \mathcal{B}$) to prune unviable recipes early. However, scaling further to $K = 4$ plateaus at 54.00% while substantially increasing wall-clock time (15.80h) and cost ($404.43). Under a fixed total budget, excessive worker concurrency splinters compute across too many concurrent branches, diminishing the search depth achievable along any individual tree. Thus, $K = 3$ establishes an optimal Pareto balance between exploration breadth, refinement depth, and resource expenditure.

Collective synergy vs. uncoordinated concurrency. Contrasting Gödel Forest with the **3 Indep.** baseline (three concurrent workers evolving without shared memory) isolates the effect of cross-tree communication. Deprived of shared lessons and the global leaderboard, three independent workers achieve only 47.00% (a steep 7.00% performance drop) despite consuming comparable budget ($421.63 and 13.38h). Without shared memory, parallel workers repeatedly squander training resources on identical dead-ends that were already diagnosed by peers. This confirms that Gödel Forest's superior performance stems from collective cognitive synergy rather than naive parallel compute scaling.

4.5 Search Dynamics and Cross-Worker Knowledge Synthesis

To understand how persistent tree search and shared memory interact in practice, we examine both the macroscopic trajectory dynamics across all benchmarks (Figure 4) and a microscopic case study on SWE-bench Verified (Figure 5).

Macroscopic search dynamics: volatility and multi-tree resilience. As illustrated in Figure 4, data-centric RSI trajectories are inherently non-monotonic: high-performing checkpoints are frequently followed by sharp performance drops when workers explore aggressive data mixtures or

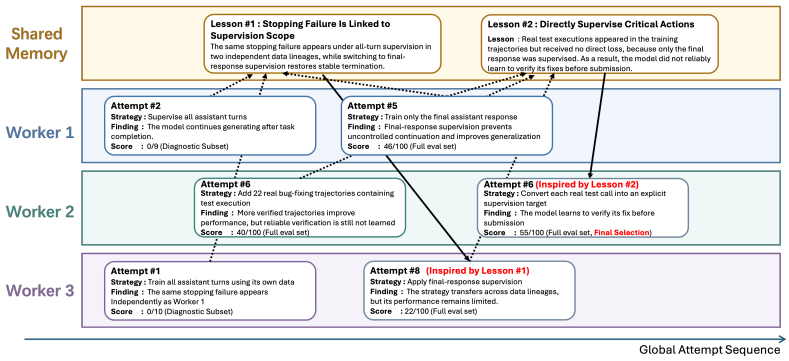


Figure 5: Cross-worker knowledge synthesis on SWE-bench Verified. Workers cultivate private search trees (horizontal lanes) and coordinate asynchronously via shared evolution memory (top lane). Dotted arrows trace lesson distillation from empirical findings; dashed arrows show lessons inspiring subsequent data interventions. Candidate scores reflect within-run evaluations.

novel supervision formats. Rather than guaranteeing monotonic ascent, each attempt acts as an empirical hypothesis test. In this volatile optimization landscape, Gödel Forest provides critical structural resilience: if an individual worker hits a severe regression or local dead-end, parallel workers continue advancing complementary branches; meanwhile, the persistent global archive decouples speculative exploration from checkpoint selection, guaranteeing that temporary failures never forfeit prior performance peaks.

Microscopic knowledge synthesis: turning failures into collaborative leaps. Figure 5 illustrates how workers coordinate without copying raw data or sharing verbose logs, but by synthesizing empirical breakdowns into compact, actionable lessons.

Lesson #1 (Stopping Boundary Guardrail): Under dense all-turn supervision, Worker 1 (Attempt #2, $0/9$) and Worker 3 (Attempt #1, $0/10$) independently encounter severe runaway generation where the model fails to terminate after finishing tasks. Diagnosing that multi-turn loss degrades termination tokens, Worker 1 restricts supervision strictly to the final assistant turn in Attempt #5, restoring termination boundaries and achieving a 46% score. This finding crystallizes into **Lesson #1**: *loss masking to the final assistant turn restores stable task termination*. Reading this lesson, Worker 3 immediately adopts final-turn masking in Attempt #8, boosting its score from near-zero to 22% and preventing further compute waste.

Lesson #2 (Supervising Verification Actions): Although final-turn supervision stabilizes completion, it exposes an unexpected limitation: the model ceases executing self-verification tests because intermediate bash calls receive zero training loss. Worker 2 notes this in Attempt #6 (40%), distilling **Lesson #2**: *critical intermediate actions, such as test executions, require explicit supervision*. To reconcile this with Lesson #1, Worker 2 synthesizes *prefix-truncated trajectories*, converting intermediate verification commands into pseudo-terminal targets. In revised Attempt #6, the model actively executes test suites prior to submission, achieving the peak score of $\mathbf{55}\%$ and earning final checkpoint selection.
Together, these empirical dynamics show how Gödel Forest turns noisy trial-and-error into compounding, self-correcting scientific discovery.

5 Conclusion

We presented **Gödel Forest**, a multi-agent framework for data-centric recursive self-improvement that resolves the depth-versus-breadth search dilemma. Locally, agents cultivate persistent search trees with diagnostic gating for deep hypothesis pursuit; globally, parallel trees explore distinct data regions and coordinate asynchronously via shared memory. By distilling outcomes into compact shared lessons anchored to a leaderboard, Gödel Forest establishes a collective feedback loop where failures prune dead-ends and breakthroughs seed new branches. Across six RSIBench-Data benchmarks, Gödel Forest achieves consistent gains over single-agent and parallel baselines (+10.70% average score, +7.00% ablation gain) with lower wall-clock latency, validating asynchronous tree co-evolution as a scalable self-improvement paradigm.

AI USE STATEMENT

In this work, we used generative AI tools for implementing methods and assisting with translation. We have not used generative AI tools for proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, cleaning and reformatting datasets, supporting qualitative and thematic data analysis, or interpreting results. Generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, and assisting in the writing of proofs are not applicable to this work. Additionally, we used generative AI tools for finding related work. We have reviewed all AI-assisted work: all AI-assisted code implementations were manually inspected, debugged, and verified through unit and integration testing by the authors, and all suggested citations were independently read and confirmed. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, et al. Swe-bench pro: Can ai agents solve long-horizon software engineering tasks? *arXiv preprint arXiv:2509.16941*, 2025.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-zero: Self-evolving reasoning llm from zero data. In *International Conference on Learning Representations*, volume 2026, pp. 130770–130790, 2026.

Alex Iacob, Andrej Jovanović, William F Shen, Daniel Burkhardt, Meghdad Kurmanji, Nurbek Tastan, Lorenzo Sani, Niccolò Alberto Elia Venanzi, Ambroise Odonnat, Zeyu Cao, et al. The red queen g\" odel machine: Co-evolving agents and their evaluators. *arXiv preprint arXiv:2606.26294*, 2026.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Samuel Kessler, Menglin Xia, Daniel Madrigal Diaz, Dongge Han, Helia Heshemi, Saravan Rajmohan, Victor Ruehle, and Jordan T Ash. Towards active synthetic data generation for finetuning language models. *arXiv preprint arXiv:2512.00884*, 2025.

Zaid Khan, Elias Stengel-Eskin, Jaemin Cho, and Mohit Bansal. Dataenvgym: Data generation agents in teacher environments with student feedback. In *International Conference on Learning Representations*, volume 2025, pp. 53480–53507, 2025.

Ilia Kulikov, Chenxi Whitehouse, Tianhao Wu, Yixin Nie, Swarnadeep Saha, Eryk Helenowski, Weizhe Yuan, Olga Golovneva, Jack Lanchantin, Yoram Bachrach, et al. Autodata: An agentic data scientist to create high quality synthetic data. *arXiv preprint arXiv:2606.25996*, 2026.

Hanjing Li, Qiguang Chen, Chenyuan Zhang, Qionglin Qiu, Fanqing Meng, Mengkang Hu, Libo Qin, and Min Zhang. Towards ai that improves itself: A survey of recursive self-improvement. 2026a.

Qizheng Li, Yifei Zhang, Xiao Yang, Xu Yang, Zhuo Wang, Weiqing Liu, and Jiang Bian. Ft-dojo: Towards autonomous llm fine-tuning with language agents. *arXiv preprint arXiv:2603.01712*, 2026b.

Changzhi Liu, Yilun Liu, Sikuan Yan, Volker Tresp, and Yunpu Ma. Mendel g\" odel machine: Recursive self-improving coding agents via comparative evolution. *arXiv preprint arXiv:2608.07645*, 2026.

Fanqing Meng, Lingxiao Du, Qiguang Chen, Ziqi Zhao, Haocheng Lu, Mengkang Hu, and Michael Qizhe Shieh. Rsibench-data: Benchmarking data-centric research for recursive self-improvement. *arXiv preprint arXiv:2607.25886*, 2026.

Mike Merrill, Alexander Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations*, volume 2026, pp. 40903–40986, 2026.

Ben Rank, Hardik Bhatnagar, Ameya Prabhu, Shira Eisenberg, Karina Nguyen, Matthias Bethge, and Maksym Andriushchenko. Posttrainbench: Can llm agents automate llm post-training? *arXiv preprint arXiv:2603.08640*, 2026.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Maxime Robeyns, Martin Szummer, and Laurence Aitchison. A self-improving coding agent. *arXiv preprint arXiv:2504.15228*, 2025.

Juergen Schmidhuber. Goedel machines: Self-referential universal problem solvers making provably optimal self-improvements, 2006. URL https://arxiv.org/abs/cs/0309048.

Wenyi Wang, Piotr Piękos, Li Nanbo, Firas Laakom, Yimeng Chen, Mateusz Ostaszewski, Mingchen Zhuge, and Jürgen Schmidhuber. Huxley-g\" odel machine: Human-level coding agent development by an approximation of the optimal self-improving machine. In *International Conference on Learning Representations*, volume 2026, pp. 80356–80386, 2026.

Xuening Wu, Shenqin Yin, Yanlan Kang, Xinhang Zhang, Qianya Xu, Zeping Chen, and Wenqiang Zhang. Sgm: A statistical godel machine for risk-controlled recursive self-modification. *arXiv preprint arXiv:2510.10232*, 2025.

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.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

Taro Yano, Yoichi Ishibashi, and Masafumi Oyamada. Lamdagent: An autonomous framework for post-training pipeline optimization via llm agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 30066–30083, 2025.

Xunjian Yin, Xinyi Wang, Liangming Pan, Li Lin, Xiaojun Wan, and William Yang Wang. Gödel agent: A self-referential agent framework for recursively self-improvement. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27890–27913, 2025.

Zhaojian Yu, Penghao Yin, Shuzheng Gao, Shilin He, Kai Cai, and Xiao-Ping Zhang. Autotrainess: Teaching language models to improve language models autonomously. *arXiv preprint arXiv:2606.31551*, 2026.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024.

Daoguang Zan, Zhirong Huang, Wei Liu, Hanwu Chen, Shulin Xin, Linhao Zhang, Qi Liu, Li Aoyan, Lu Chen, Xiaojian Zhong, et al. Multi-swe-bench: A multilingual benchmark for issue resolving. *Advances in Neural Information Processing Systems*, 38, 2026.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin gödel machine: open-ended evolution of self-improving agents. In *International Conference on Learning Representations*, volume 2026, pp. 104223–104294, 2026a.

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

Andrew Zhao, Yiran Wu, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *Advances in Neural Information Processing Systems*, 38:105816–105879, 2026a.

Zhengyang Zhao, Shengjie Ye, Lu Ma, Hao Liang, Hengyi Feng, and Wentao Zhang. Andes: Agent native data evolving synthesis tool for autonomous instruction alignment. *arXiv preprint arXiv:2606.01279*, 2026b.