STELLAR COLOSSEUM: A MANY-AGENT HARNESS FOR LONG-HORIZON RESEARCH IN MATHEMATICS AND THEORETICAL COMPUTER SCIENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Language models can produce plausible short proofs, but may still be unreliable on long-horizon research problems, where progress depends on a sequence of uncertain and interdependent decisions. We introduce STELLAR COLOSSEUM, a model-agnostic harness for allocating inference across research in mathematics and theoretical computer science. COLOSSEUM explores alternative strategies before proof construction, uses a readiness gate to decide when a route is mature enough to decompose, represents the proof plan as interdependent section-level subproblems, and routes verifier findings back to the affected part of the argument. Across these stages, it generates candidates in parallel, attacks them with targeted falsification, and combines candidates and their critiques into a single research artifact through overlapping random-sample tree aggregation.

We demonstrate the capabilities of COLOSSEUM through open-ended research and evaluations on theorem-proving and competitive programming benchmarks. Using COLOSSEUM with Gemini 3.1 Pro, we obtain several new results that address open problems arising from papers published at top venues such as FOCS and JMLR. On TCS-BENCH (Cohen-Addad et al., 2026), a benchmark of research-level theorem-proving tasks drawn from papers published at FOCS, STOC, and SODA, COLOSSEUM achieves 71.0% accuracy using Gemini 3.1 Pro and Gemini 3.7 Flash. In a separate Codeforces evaluation using Gemini 3.1 Pro, the proof-oriented pipeline with execution feedback solves 218 of 222 problems.

1 INTRODUCTION

Language models have made substantial progress in mathematical reasoning. Systems built around them have solved olympiad-level problems (Trinh et al., 2024; Hubert et al., 2026). Recent studies also describe how language models have helped obtain new results in mathematics and theoretical computer science (Feng et al., 2026b; Woodruff et al., 2026). These successes motivate a closer look at how to organize research over many rounds of exploration, proof construction, and revision.

Work on a research problem often starts with a literature search and small examples. Researchers may try several representations and conjectures before finding lemmas worth proving. Even a promising approach can fail if a key lemma turns out to be false or a later step exposes a missing assumption. As the proof grows, definitions and assumptions must remain consistent across dependent lemmas; changing one part may alter what later sections need to establish. A failed approach may still yield a useful restriction, counterexample, or alternative formulation. Progress depends on deciding what to develop, what to repair, and when to change direction, using the evidence gathered along the way.

Inference-time scaling gives a system more opportunities to explore possible solutions and check them. Sampling multiple reasoning paths, searching over intermediate states, and iteratively critiquing an answer can improve reasoning quality (Wang et al., 2023; Yao et al., 2023; Madaan et al., 2023). Agreement among candidates can still hide a shared error, so a flat vote over final answers offers limited guidance for proofs. The challenge is to use this computation to choose between approaches, uncover specific gaps, and preserve what remains useful after an attempt fails.

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.

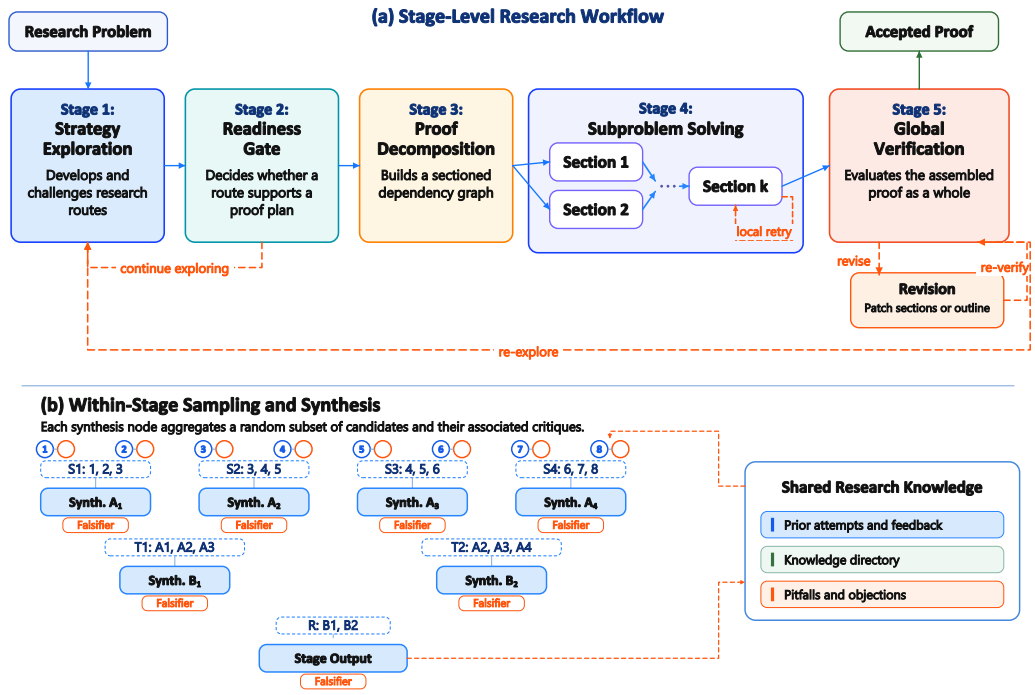


Figure 1: Overview of the COLOSSEUM architecture. Panel (a) shows the stage-level workflow from strategy exploration through global verification, including local revision and re-exploration. Panel (b) expands inference within a stage. Synthesis nodes aggregate random subsets of candidates and their associated critiques, and each is paired with a falsifier.

We introduce STELLAR COLOSSEUM (COLOSSEUM for short), a model-agnostic system for long-horizon research in mathematics and theoretical computer science. COLOSSEUM builds on and substantially extends the parallel exploration and iterative verification architecture described by Woodruff et al. (2026), adding explicit control over research stages, dependency-aware proof construction, and aggregation that preserves critiques. The system keeps track of proposed strategies, partial proofs, and verification findings as the work proceeds. This shared research state allows exploration, proof construction, and revision to build on one another.

COLOSSEUM organizes inference at two levels (Figure 1). At the workflow level, it develops and challenges alternative strategies until one is concrete enough to decompose into a sectioned plan with explicit dependencies. Independent subproblems can then be developed in parallel, and the completed sections are assembled into a single proof. Verification feedback guides repairs to the argument and reopens exploration when the underlying strategy is no longer viable.

At the stage level, COLOSSEUM generates a population of candidate artifacts for tasks such as strategy exploration, proof planning, and proof construction. It subjects these candidates to targeted falsification and combines them through overlapping random-sample tree aggregation. Critiques remain attached to the proposals they address, so aggregation can combine useful ideas while retaining objections and evidence of failure. The resulting artifact, together with any unresolved objections, becomes part of the research state used in later stages.

Our Contributions.

1. We formulate long-horizon research as a pipeline that separates strategy exploration, proof decomposition, subproblem solving, and global verification. Decomposition produces a sectioned plan with explicit dependencies. Independent subproblems can then be solved in parallel; rejected proofs are revised or returned to exploration (Section 3).

2. We develop a stage-level adversarial inference procedure. It combines diverse candidate generation, targeted falsification, and overlapping tree-structured aggregation while retaining both synthesized content and evidence against it (Section 3.3).
3. We study how many initial copies are needed for useful evidence to survive repeated aggregation with constant probability. In an idealized model, retention and reuse determine whether this number grows polynomially, logarithmically, or stays constant as the number of initial candidates grows (Section 4).
4. We demonstrate COLOSSEUM on open-ended research problems. The workflow contributed to several research results, and we present case studies on long-form proof construction and independent strategy rediscovery. We also evaluate COLOSSEUM on research-level theorem proving and competitive programming (Section 5).

2 RELATED WORK

We review work on inference-time search and critique, discovery guided by verifiers and evaluators, and systems that support sustained research. Appendix F provides the extended review.

Search, Critique, and Aggregation. Sampling diverse reasoning paths and aggregating their answers (Wang et al., 2023), searching over intermediate states (Yao et al., 2023), revising against model-generated feedback (Madaan et al., 2023), and organizing multi-agent debate (Du et al., 2024) provide complementary ways to use additional inference. Wu et al. (2025) compare inference strategies under fixed compute budgets and introduce REBASE, which allocates tree expansions using intermediate rewards. COLOSSEUM uses search, critique, and aggregation throughout the research workflow. Within each stage, proposals remain paired with their falsification reports during aggregation. The resulting artifact, unresolved objections, and repair targets remain available during strategy selection, proof construction, and verification.

Research Workflows and Verification. Research workflows must decide how to reuse intermediate work and when to ask for further verification. Aletheia maintains a generate–verify–revise loop over long natural-language solutions (Feng et al., 2026b). Other systems assign these tasks to different roles: QED separates decomposition, proof generation, and verification (An et al., 2026); ProofCouncil iterates between an author and a critic (Schmitt et al., 2026); and RMA coordinates initializer, proposer, and verifier agents through shared structured memory (Zhao et al., 2026). These systems use verification repeatedly as an argument develops. Danus admits verifier-approved claims into a fact graph together with their proofs and logical dependencies (Liu et al., 2026). COLOSSEUM combines these ideas to connect strategy exploration with the development and repair of a long proof. Its readiness gate controls when a route is decomposed into subproblems, and its aggregation procedure carries critiques alongside the proposals they address. Shared drafts and research knowledge keep that evidence available during both local revision and renewed exploration.

3 METHOD

3.1 WORKFLOW OVERVIEW

Research Pipeline. Figure 1 summarizes the architecture. Panel (a) shows the overall research workflow, while panel (b) expands inference within a stage. Exploration develops and tests candidate strategies until the readiness gate selects a route that is concrete enough to support proof construction. Decomposition turns that route into a sectioned proof skeleton and a dependency graph over its subproblems. Eligible subproblems are then solved, with independent sections processed in parallel. A section that fails its local review is retried without restarting unaffected subproblems, and the completed sections are assembled into a single proof.

The global verifier evaluates the assembled proof. A rejected proof either enters revision or reopens exploration. Revision may replace one or more section bodies or modify the proof outline before the argument is assembled and verified again. Re-exploration is reserved for cases in which the current strategy no longer supports a credible path to the target. Sections 3.2 and 3.4 develop these stages and return paths in detail.

Inference within a Stage. The research pipeline determines which task is currently being addressed, but a difficult stage is not entrusted to a single model continuation. COLOSSEUM generates a population of candidates, subjects them to targeted falsification, and iteratively synthesizes random subsets of candidates and their associated critiques through the overlapping sampling tree shown in panel (b). The candidate type changes with the stage—a strategy, a proof skeleton, a section body, a verifier assessment, or a revision—but the inference pattern is reused. The root artifact and any unresolved objections are returned to the research pipeline. In this way, stage-level decisions determine where additional inference is applied, while aggregation determines how a difficult decision within that stage is resolved. Section 3.3 describes this inner procedure in detail.

Context across Stages. Information produced earlier in the pipeline remains available to later stages. Within a round, this includes the current strategy, sectioned proof plan, completed section bodies, and verifier findings. Across rounds, a rejected proof draft and its verifier feedback are passed directly into the next attempt. A shared knowledge directory further retains reusable results from the wider search. Section 3.4 describes these two forms of cross-round memory. Appendix B.4 discusses the design challenges that motivate the workflow.

3.2 STRATEGY EXPLORATION AND PROOF CONSTRUCTION

Strategy Exploration and Readiness. COLOSSEUM begins by exploring proof strategies rather than drafting a proof around the first plausible idea. Parallel attempts develop routes based on different reformulations, intermediate claims, and connections to known results. The purpose of this stage is to expose what each route would require, which parts remain conjectural, and where the main technical difficulties lie.

The readiness gate asks whether one of these routes is concrete enough to support a proof plan. It tests whether the remaining uncertainty can be localized within a stable proof architecture, not whether the proof has already been completed. A route passes when its central reduction or mechanism is stable, its unresolved claims are precise enough to assign to proof sections, and no unresolved bridge is likely to change the target or the architecture of the argument. A central lemma may remain difficult; what matters is that its statement, role, and expected path to proof or verification are explicit. If these conditions are not met, exploration continues; otherwise, the selected route is passed to decomposition.

Decomposition and Parallel Proof Construction. The decomposer turns the selected route into a numbered, sectioned proof skeleton. Each section is associated with a subproblem specifying the mathematical content that must be established there. Dependency edges record which completed sections a subproblem may use, producing a directed acyclic graph. The document order controls the intended exposition, while the dependency graph controls the order in which the mathematical work can proceed.

A subproblem becomes eligible once its dependencies have been completed, so independent sections can be solved in parallel. Each solver receives the assigned task together with the relevant completed sections. Difficult subproblems can use the adversarial inference procedure of Section 3.3 to develop and test alternative solutions.

Before a section is committed to the skeleton, a local reviewer checks it against the assigned subproblem and the completed material on which it depends. If the section remains incomplete or the review identifies an unresolved defect, the same subproblem is run again with the failed section and the review as additional input. The retry is local to the affected subproblem: completed work elsewhere in the dependency graph is preserved. Once a section is accepted locally, its body replaces the corresponding placeholder and becomes available to downstream subproblems. Filling the skeleton in this way produces the candidate proof submitted to global verification.

3.3 ADVERSARIAL GENERATION AND TREE-STRUCTURED AGGREGATION

COLOSSEUM generates a population of candidates in parallel. Each candidate is paired with a falsifier, which checks it for counterexamples, unsupported steps, and missing assumptions. The falsifier's feedback remains attached to the candidate throughout aggregation. Appendix B.1 gives further details on candidate generation and targeted falsification.

COLOSSEUM organizes aggregation as a tree, with the initial candidates at the leaves and the final candidate at the root. At each level, aggregation nodes independently sample small groups of candidates and combine their content and falsifier feedback into new candidates. The groups may overlap, giving each candidate multiple opportunities to contribute. Newly generated candidates are checked by falsifiers before further aggregation. This process continues until a single candidate remains, together with any unresolved objections.

Tree-Structured Aggregation. Comparing every candidate and critique in one prompt becomes unwieldy at large sample counts. COLOSSEUM uses a reduction tree instead. Let z_i denote a candidate bundled with all of its falsification records, and let $\mathcal{C}^{(0)} = \{z_1, \dots, z_n\}$. The tree shape is specified by the population width at every level, $(m_0, m_1, \dots, m_L)$, with $m_0 = n$ and $m_L = 1$. For the transition from level ℓ to level $\ell + 1$, each of the $m_{\ell+1}$ aggregation nodes independently draws k_ℓ distinct inputs uniformly from the current population, where k_ℓ is the configured sample size capped by m_ℓ. Thus

$$G_j^{(\ell)} \sim \mathrm{Unif}\Big\{ G \subseteq \mathcal{C}^{(\ell)} : |G| = k_\ell \Big\}, \qquad j = 1, \dots, m_{\ell+1}.$$

Sampling is without replacement within one group, but groups for different aggregation nodes are drawn independently and may overlap. They do not form a partition. Given the stage-specific aggregator A, the current input artifact x, and contextual evidence K, each node computes

$$s_j^{(\ell+1)} = A\Big(x, K, G_j^{(\ell)}\Big), \qquad \mathcal{C}^{(\ell+1)} = \{z_j^{(\ell+1)}\}_{j=1}^{m_{\ell+1}},$$

where $z_j^{(\ell+1)}$ bundles $s_j^{(\ell+1)}$ with its subsequent falsification records. If $R_i^{(\ell)}$ is the number of next-level groups containing a fixed node from level ℓ, then

$$\mathbb{E}\Big[R_i^{(\ell)}\Big] = \frac{m_{\ell+1} k_\ell}{m_\ell}. \tag{1}$$

Ordinary contraction layers choose widths and sample sizes that give an expected reuse of roughly two to three. A transition from 128 nodes to 64 with sample size five, for example, gives expected reuse 2.5. Mixing layers and the final root may use a different rate. Overlap gives a candidate several chances to contribute while keeping each aggregation context small.

Intermediate aggregation is constructive rather than a vote or ranking: it may merge compatible components, retain competing branches, repair a localized flaw, or declare an unresolved conflict. Substantive disagreements and falsification evidence are carried forward rather than averaged away. The root returns a synthesized candidate together with any unresolved objections. Because groups may overlap, the dependency graph is a DAG rather than a disjoint tree. Selected prompt templates and their structured output interfaces are provided in Appendix C.

3.4 VERIFICATION, REVISION, AND RESEARCH MEMORY

The global verifier reads the original problem and the completed document as a single argument, with the local subproblem reviews available as audit context. Global verification is itself carried out by multiple independent reviews and tree-structured aggregation. The aggregation merges genuinely duplicate criticisms while retaining distinct substantive objections. It is not a majority vote: a concrete fatal defect is sufficient to reject the proof, and generic acceptance judgments do not resolve it.

When the verifier rejects a proof, the pipeline proceeds by revision or re-exploration. Revision is used when the current strategy remains viable; it may replace section bodies or modify the proof outline before the entire argument is verified again. If the verifier's findings undermine the central strategy, the system returns to exploration instead.

Research Memory. COLOSSEUM carries information across research rounds in two complementary forms: the latest proof attempt is retained in full, while a shared knowledge directory collects reusable findings from the wider search. When a round does not produce an accepted proof, COLOSSEUM passes the resulting draft and the verifier feedback directly into the next round.

Appendices B.2 and B.3 give further details on verification and research memory; Table 1 gives the inference configurations. To quantify the evidence-preservation role of overlapping aggregation,

Section 4 characterizes how many initial copies of a useful item are needed to reach the final synthesis with constant probability. The analysis isolates the effects of reuse and retention in an idealized geometric-width model of the aggregation DAG.

4 Theoretical Analysis of Tree Aggregation

A useful item may appear in only a few initial candidates. Overlapping aggregation gives it several chances to contribute, but each synthesis may also lose it. We thus ask how many initial copies are needed for the item to reach the final candidate with constant probability as the population grows. The constant may depend on the fixed model parameters.

In this section, we study this question using an idealized aggregation DAG with geometric layer widths. Index the layers backward from a terminal block as $N_t = mc^t$, $t = 0, \dots, L$, so that layer L contains the $n = mc^L$ initial candidates. Here $c \geq 2$ and $m, k \geq 1$ are fixed integers with $k \leq mc$. Each aggregation node uniformly samples k distinct parents from the next backward layer. If any sampled parent contains the item, the node retains it with probability $\alpha \in (0, 1]$. All sampling and retention decisions are independent. The terminal merge reads a fixed set of $h \in \{1, \dots, m\}$ nodes in layer 0 and retains the item if any of them contains it. We place r copies uniformly among distinct initial candidates, independently of the graph and retention decisions, and write $P_L(r)$ for the probability that the item reaches the output. The model tracks retention of an item; α is not a measure of proof correctness.

The interaction between reuse and retention is described by

$$\beta = \frac{k}{c}, \qquad \lambda = \alpha\beta, \qquad \mu = \alpha k = c\lambda. \tag{2}$$

Here β is the expected reuse of a candidate. When only one candidate contains the item, λ is the expected number of copies retained in the next layer. Looking backward, a node contributes either its k parents or none, depending on its retention decision. Their mean number is μ, before accounting for parents shared by different nodes.

Theorem 1 (Rare-evidence copy complexity)**.** *Fix the model parameters above and suppose $\mu > 1$. There are constants $0 < C_- \leq 1 \leq C_+ < \infty$ such that, for all sufficiently large L and every integer $1 \leq r \leq n$,*

$$C_- \min\{1, r/b_L\} \leq P_L(r) \leq C_+ \min\{1, r/b_L\}, \tag{3}$$

where

$$b_L = \begin{cases} \lambda^{-L} = \Theta(n^{\log_c(1/\lambda)}), & 1/c < \lambda < 1, \\ L + 1 = \Theta(\log n), & \lambda = 1, \\ 1, & \lambda > 1. \end{cases} \tag{4}$$

The theorem shows why expected reuse alone does not determine whether evidence survives: its effective multiplier is $\lambda = \alpha\beta$. For illustration, take $c = 2$, $k = 5$, and $m \geq 3$. Each candidate is reused 2.5 times on average, so the transition occurs at retention probability $\alpha = 0.4$. With $\alpha = 0.3$, the required copy count is $\Theta(n^{\log_2(4/3)})$; at 0.4 it is $\Theta(\log n)$; above 0.4 it is $\Theta(1)$.

Proof Intuition. For a fixed graph and retention decisions, we trace backward from the terminal nodes read by the output. Whenever a node's retention decision succeeds, we follow its sampled parents, counting shared parents only once. These positions form the influence set: the item would reach the output if it were present at any of them. Let Z_t be its size in layer t and $Q_t = Z_t/N_t$ its density. Success occurs exactly when an initial copy lies in this set at layer L, and we have

$$P_L(r) = \mathbb{E}\left[1 - \frac{\binom{n-Z_L}{r}}{\binom{n}{r}}\right], \qquad \mathbb{E}[1 - e^{-rQ_L}] \leq P_L(r) \leq \min\{1, r\mathbb{E}Q_L\}. \tag{5}$$

For a fixed influence set of density $q > 0$ in the initial layer, $\Theta(1/q)$ copies give a constant probability of success. To estimate this density, condition on $Z_t = z$. Each traced node includes a given parent with probability $\alpha k/N_{t+1} = \mu/N_{t+1}$. By independence, all z nodes miss that parent with probability $(1 - \mu/N_{t+1})^z$. Summing over parents gives

$$\mathbb{E}[Z_{t+1} \mid Z_t = z] = N_{t+1}\left[1 - \left(1 - \frac{\mu}{N_{t+1}}\right)^z\right]. \tag{6}$$

For large layer widths, a current density q gives an expected next density of approximately $1 - e^{-\lambda q}$. When $1/c < \lambda < 1$, the layers expand faster than the influence set, giving the density scale $(\mu/c)^L = \lambda^L$. At $\lambda = 1$, the linear term preserves density, but shared parents produce a quadratic loss: $1 - e^{-q} = q - q^2/2 + O(q^3)$. In this approximation, the loss makes $1/q$ grow linearly with depth, yielding the $1/(L+1)$ density scale. For $\lambda > 1$, the expected density exceeds q when q is small and positive, allowing a positive density to persist.

In Appendix A, we control the random fluctuations and prove $\mathbb{E}Q_L = O(1/b_L)$ and $\mathbb{P}(Q_L \geq d/b_L) \geq p$ for constants $d, p > 0$ independent of L. Together with (5), these bounds establish the theorem.

The same backward construction and exact one-step transition apply to arbitrary admissible layer widths. We use geometric widths to obtain the three explicit rates above; for nonuniform widths, the bounds depend on the full sequence of layer sizes.

5 Evaluation

We demonstrate the capabilities of Colosseum through open-ended research and evaluations on theorem-proving and competitive programming benchmarks.

Inference Configurations. For open-ended research problems (Section 5.1), we use a range of tree configurations during strategy exploration, some with slightly over 100 leaf nodes. These larger trees are used only for exploration; all remaining stages share a fixed configuration with 16 leaf nodes. Table 1 summarizes these configurations. These configurations specify population widths and aggregation fan-in rather than the exact total number of model calls, which also depends on the number of proof sections, local retries, and global revision rounds.

Table 1: Stage-level inference configurations. At level ℓ, the effective sample size is $\min\{k, m_\ell\}$.

Setting	Stage	Tree widths $\mathbf{m}$	Sample size k
Open-problem research	Strategy exploration	Varies	Varies
TCS-Bench	Strategy exploration	$(32, 16, 8, 5, 1)$	5
Codeforces	Strategy exploration	$(32, 16, 8, 5, 1)$	5
All three settings	All remaining stages	$(16, 8, 5, 1)$	5

5.1 Open-Ended Research Results

We first present five selected research problems spanning approximation, sparse optimization, vector representations, quantization, and streaming. Using Gemini 3.1 Pro, the workflow contributed to advances on these problems, including sharper guarantees from existing sampling methods, simpler estimators with improved error bounds, and lower bounds on computational and representational resources. Together, they illustrate the range of mathematical tasks to which the workflow has contributed. Table 2 summarizes the principal advances. The mathematical arguments were checked by the authors of the respective papers, which provide the complete statements, parameter conditions, attribution, and proofs. Appendix D expands the mathematical context. The two cases below illustrate long-form proof construction and strategy development across research rounds.

Long-Form Proof Construction: Knuth's Cycles. For an integer $m > 2$, Knuth's cycles problem asks whether the directed edges of the Cayley graph $\Gamma_m = \mathrm{Cay}(\mathbb{Z}_m^3, \{e_1, e_2, e_3\})$ can be partitioned into three directed Hamiltonian cycles (Knuth, 2026). Brenner et al. (2026) give proofs for two simple even-case constructions for Knuth's cycles, together with 46- and 75-page proof drafts produced by the workflow. Although each construction is specified by a compact set of local routing rules, proving that it yields three Hamiltonian cycles requires a substantially more involved global argument. These examples illustrate how a long proof can be developed as a persistent, revisable document organized around an explicit dependency structure.

Strategy Rediscovery: The Erdős Unit-Distance Problem. Following the reported counterexample to the unit-distance conjecture (Alon et al., 2026), we ran Colosseum on the same problem using Gemini 3.1 Pro as the base model, with internet access disabled and only the original problem

Table 2: Five research results to which the workflow contributed. Symbols are local to each problem; full hypotheses and statements appear in the cited papers and Appendix D.

Research problem	Principal advance
Strong ℓ_p coresets, $p > 2$	Improves the coreset size from $\widetilde{O}_p(k^{p/2}\varepsilon^{-p})$ to $\widetilde{O}_p(k^{p/2}\varepsilon^{-2})$ using the same sampling rule (Lin et al., 2026b).
Sparse least squares	Establishes a condition-number barrier under the randomized exact-volume Small-Set Expansion Hypothesis, ruling out any fixed sublinear power of the restricted condition number in the stated regime (Lin et al., 2026a).
Maximum inner product embeddings	Proves a dimension lower bound $D \geq m^{c_\delta/\varepsilon^{2-2\delta}}$, nearly closing the gap in the exponent and allowing fully data-dependent representations (Jayaram et al., 2026a).
Hadamard quantization	Obtains single-stage $1/(d4^b)$ mean-squared-error scaling, removes a residual-stage $O(d)$-bit payload, and reduces the leading upper-bound constant by a factor of approximately 5.93 (Lin et al., 2026c).
Prefix-matrix factorizations	Proves $\gamma_{2,1}(Q) = \Omega(\log^{3/2} n/(\log\log n)^{3/2})$. Concurrent work improves the dependence on $\log\log n$ (Lin et al., 2026d).

statement as the initial prompt. The model's documented knowledge cutoff is January 2025 (Google, 2026), before the reported counterexample. The resulting 22-page draft (Colosseum Research Harness, 2026), available here, developed a number-theoretic approach based on unramified towers and relative unit groups, independently arriving at the central architecture of the OpenAI solution (Alon et al., 2026). Notably, the run proceeded through 15 exploration rounds. Across these rounds, the accumulated-knowledge layer carried partial results, objections, and failed attempts forward across successive rounds, allowing the strategy-selection loop to synthesize the accumulated evidence into subsequent research directions. This provides a concrete example of the architecture sustaining a coherent long-horizon research process. The full case studies and research results are given in Appendix D.

5.2 Research-Level Theorem Proving: TCS-Bench

The preceding results and case studies examine individual problems in depth. To evaluate the breadth of the system on a common set of research-level problems, we also test Colosseum on TCS-Bench (Cohen-Addad et al., 2026). All Colosseum runs in this evaluation have internet access disabled. The benchmark contains 300 theorem-proving tasks derived from papers published at FOCS, STOC, and SODA between 2020 and 2026. Each task supplies the mathematical context needed to state the problem and asks the model to produce a self-contained proof of a target theorem. Candidate proofs are scored by a reference-assisted automated grader that also receives the benchmark's ground-truth proof. Its prompt was optimized on a separate set of 100 expert-labeled proofs, on which it reported more than 90% accuracy. All benchmark accuracies below are measured by this grader.

We run Colosseum separately with Gemini 3.1 Pro and Gemini 3.7 Flash, producing one candidate proof from each run for every problem. To select between the two candidates, Gemini 3.7 Flash produces eight independently sampled critiques of the Gemini 3.1 Pro proof. If at least five critiques judge that proof correct, it is submitted; otherwise, the Gemini 3.7 Flash proof is submitted. The benchmark grader is used only for evaluation and plays no role in proof selection.

The two individual runs have nearly identical overall accuracy, but their errors are sufficiently complementary for cross-model selection to solve 213 problems, an improvement of 48 problems over the stronger individual run. The critique signal distinguishes proofs labeled correct and incorrect by the benchmark grader with an AUC of 0.896; routing with Gemini 3.1 Pro's internal verifier alone yields 64.7% accuracy. Among the evaluated non-oracle methods on this dataset, cross-model selection gives the highest accuracy.

The table reports accuracy under the stated inference configurations, not a compute-matched comparison. Cross-model selection uses two full harness runs and eight additional critiques.

Table 3: TCS-BENCH results on 300 tasks. Cross-model selection uses critiques without access to the reference proof. The oracle selects a grader-correct proof whenever either run supplies one and is an upper bound on selection from these two candidates.

Direct model evaluation

Model	Accuracy
Gemini 3.1 Pro	30.3%
Gemini 3.1 DeepThink	52.0%
GPT-5.6 Pro (max)	68.0%

COLOSSEUM **evaluation**

Model / selector	Accuracy
Gemini 3.1 Pro	54.0%
Gemini 3.7 Flash	55.0%
Cross-model selection	71.0%
Oracle best-of-two	77.3%

5.3 TRANSFER TO COMPETITIVE PROGRAMMING

To test whether the same architecture transfers from theorem proving to executable algorithmic tasks, we follow the Codeforces evaluation described for Gemini 3 Deep Think: all 222 problems with clist.by difficulty estimates above 1500 from contests held between April and October 2025 (Google DeepMind, 2026). In our evaluation snapshot, these problems come from 52 Codeforces contests numbered 2084–2162. Each submitted C++ solution is compiled and run against the complete hidden test set using the problem's original checker, and is accepted only if every test passes. Strict as-submitted grading yields the same score, so no accepted solution depends on an output repair.

Architecture Transfer. When applied to competitive programming, COLOSSEUM retains the proof-oriented architecture used for mathematical research. Exploration searches over solution strategies and passes the selected route to the decomposer. Rather than partitioning a program into software components, the decomposer represents that route as a directed acyclic graph of mathematical and algorithmic subproblems. Each subproblem is solved and aggregated as described in Section 3. The decomposition differs only at its endpoint: a terminal implementation subproblem realizes the algorithm established by the preceding nodes as a single C++ program. Appendix E gives an example decomposition for Codeforces 2084F.

Execution Feedback. The one addition to this reasoning pipeline is a C++ execution probe. Once a candidate implementation is available, the probe compiles and runs it on public samples and model-generated stress inputs, returning checker outcomes together with time and memory measurements to the existing verification–revision loop. Hidden tests remain inaccessible to the workflow and are used only for final grading. The probe makes implementation-stage failures available to the existing verification–revision loop.

Results. With Gemini 3.1 Pro as the base model, the execution-enabled configuration records 218 accepted solutions on the 222-problem suite. The comparison configuration without the execution probe records 213 accepted solutions (Table 4). Appendix E reports the difficulty distribution and the corpus-level performance-rating calculation.

Table 4: Execution feedback within the proof-oriented pipeline.

Configuration	Accepted / total
Without execution probe	213/222
With execution probe	218/222

6 CONCLUSION

We presented COLOSSEUM, a model-agnostic many-agent harness for long-horizon research in mathematics and theoretical computer science. It connects strategy exploration, proof construction, and revision through a shared research state, while tree aggregation combines candidates and their critiques within each stage. The harness contributed to several new research results and achieved strong performance on TCS-BENCH and in competitive programming. Appendix G discusses limitations and directions for future work.

AI USE STATEMENT

We used generative language models throughout the COLOSSEUM research workflow and in the evaluations reported in Section 5. We also used these models to help develop and refine the workflow.

AI tools helped develop and check the theoretical analysis in Section 4 and assisted with writing and editing this manuscript. The mathematical arguments for the research results in Section 5.1 were checked by the authors of the respective papers. We take responsibility for the final text, claims, and results reported in this manuscript.

REPRODUCIBILITY STATEMENT

The workflow is described in Section 3, with further details on candidate generation, falsification, verification, and research memory in Appendix B. To specify how it is run, Table 1 lists the inference configurations, and Appendix C provides abridged prompt templates and their information flow. For the theoretical analysis, Appendix A gives the full model assumptions and proof of Theorem 1. The evaluation procedures are described in Section 5, including proof grading, cross-model selection, and program testing; Appendix E provides further details on the programming benchmark and its rating calculation. Complete statements and proofs of the open-ended research results appear in the papers cited in Section 5.1.

REFERENCES

Noga Alon, Thomas F. Bloom, W. T. Gowers, Daniel Litt, Will Sawin, Arul Shankar, Jacob Tsimerman, Victor Wang, and Melanie Matchett Wood. Remarks on the disproof of the unit distance conjecture. *arXiv preprint arXiv:2605.20695*, 2026. URL https://arxiv.org/abs/2605.20695.

AlphaCode Team. AlphaCode 2 technical report. Technical report, Google DeepMind, December 2023. URL https://storage.googleapis.com/deepmind-media/AlphaCode2/AlphaCode2_Tech_Report.pdf.

Levent Alpöge. Hello there the Jacobian conjecture is false. X post, July 2026. URL https://x.com/__alpoge__/status/2079028340955197566. Published July 20, 2026.

Chenyang An, Qihao Ye, Minghao Pan, and Jiayaun Zhang. QED: An open-source multi-agent system for generating mathematical proofs on open problems. *arXiv preprint arXiv:2604.24021*, 2026. doi: 10.48550/arXiv.2604.24021. URL https://arxiv.org/abs/2604.24021.

Anthropic. Discovering cryptographic weaknesses with Claude. Anthropic Frontier Red Team, July 2026a. URL https://www.anthropic.com/research/discovering-cryptographic-weaknesses. Published July 28, 2026.

Anthropic. Learning more about Claude's mathematical capabilities. Anthropic Science, August 2026b. URL https://www.anthropic.com/research/riemann-zeta. Published August 10, 2026; updated August 13, 2026.

Keston Aquino-Michaels. Completing Claude's cycles: Multi-agent structured exploration on an open combinatorial problem, March 2026. URL https://github.com/no-way-labs/residue. Preprint.

Kyriakos Axiotis and Maxim Sviridenko. Sparse convex optimization via adaptively regularized hard thresholding. *Journal of Machine Learning Research*, 22(122):1–47, 2021. URL https://jmlr.org/papers/v22/20-661.html.

Federico Bianchi, Yongchan Kwon, Aneesh Pappu, and James Zou. Harnessing the collective intelligence of AI agents in the wild for new discoveries. *arXiv preprint arXiv:2606.10402*, 2026. doi: 10.48550/arXiv.2606.10402. URL https://arxiv.org/abs/2606.10402.

Michael P. Brenner, Honghao Lin, Vahab Mirrokni, and David P. Woodruff. Simple even-case constructions in Knuth's cycles and their Gemini-generated proofs. Manuscript and accompanying proof artifacts, 2026. URL https://github.com/dpwoodru/knuthCycles.

Jan Bulánek, Ravi Kumar, Raghu Meka, Jelani Nelson, and Tamás Sarlós. A matrix factorization approach in turnstile streaming. *arXiv preprint arXiv:2607.28819*, 2026. URL https://arxiv.org/abs/2607.28819.

Yichuan Cao, Ruichen Qiu, Junqi Liu, Jiaqi Wang, Dakai Guo, Ruyong Feng, Lihong Zhi, and Xiao-Shan Gao. MechMath agent team: LLM driven agents for mathematical research. *arXiv preprint arXiv:2607.04394*, 2026. doi: 10.48550/arXiv.2607.04394. URL https://arxiv.org/abs/2607.04394.

Stephen Chung, Wenyu Du, and William J. Wesley. Autonomous mathematical discovery in an open-world multi-agent environment. *arXiv preprint arXiv:2608.23691*, 2026. doi: 10.48550/arXiv.2608.23691. URL https://arxiv.org/abs/2608.23691.

Vincent Cohen-Addad, Dimitris Paparas, Ernest van Wijland, Max Springer, Julien Canitrot-Paradis, Honghao Lin, David P. Woodruff, Adarsh Kumarappan, Rajesh Jayaram, Rudrajit Das, Lalit Jain, Ola Svensson, Silvio Lattanzi, Mislav Balunovic, Theophane Weber, and Vahab Mirrokni. TCS-BENCH: Benchmarking state-of-the-art generative AI theoretical computer science research ability. *arXiv preprint arXiv:2608.09538*, 2026. URL https://arxiv.org/abs/2608.09538.

Colosseum Research Harness. A constructive lower bound for C_{84} via unramified mixed-signature towers. Generated proof artifact, 2026. URL https://drive.google.com/file/d/1krFP97U45LjRs9_Ah-TKDoc8LlnVI832/view?usp=sharing. Generated by a Gemini 3.1 Pro research harness.

Xingyu Dang, Rohit Agarwal, Rodrigo Porto, Anirudh Goyal, Liam H. Fowl, and Sanjeev Arora. Escaping the cognitive well: Efficient competition math with off-the-shelf models. *arXiv preprint arXiv:2602.16793*, 2026. doi: 10.48550/arXiv.2602.16793. URL https://arxiv.org/abs/2602.16793.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11733–11763. PMLR, 2024. URL https://proceedings.mlr.press/v235/du24e.html.

Ahmed El-Kishky, Alexander Wei, Andre Saraiva, Borys Minaiev, Daniel Selsam, David Dohan, Francis Song, Hunter Lightman, et al. Competitive programming with large reasoning models. *arXiv preprint arXiv:2502.06807*, 2025. doi: 10.48550/arXiv.2502.06807. URL https://arxiv.org/abs/2502.06807.

Tony Feng, Junehyuk Jung, Sang-hyun Kim, Carlo Pagano, Sergei Gukov, Chiang-Chiang Tsai, David P. Woodruff, Adel Javanmard, Aryan Mokhtari, Dawsen Hwang, Yuri Chervonyi, Jonathan N. Lee, Garrett Bingham, Trieu H. Trinh, Vahab Mirrokni, Quoc V. Le, and Thang Luong. Aletheia tackles FirstProof autonomously. *arXiv preprint arXiv:2602.21201*, 2026a. URL https://arxiv.org/abs/2602.21201.

Tony Feng, Trieu H. Trinh, Garrett Bingham, Dawsen Hwang, Yuri Chervonyi, Junehyuk Jung, Joonkyung Lee, Carlo Pagano, Sang-hyun Kim, Federico Pasqualotto, Sergei Gukov, Jonathan N. Lee, Junsu Kim, et al. Towards autonomous mathematics research. *arXiv preprint arXiv:2602.10177*, 2026b. URL https://arxiv.org/abs/2602.10177.

Ying Feng, Piotr Indyk, Michael Kapralov, Dmitry Krachun, and Boris Prokhorov. Provable quantization with randomized Hadamard transform. *arXiv preprint arXiv:2605.13810*, 2026c. URL https://arxiv.org/abs/2605.13810.

Shuhong Gao. Counterexamples to the Jacobian conjecture in dimensions greater than two. *arXiv preprint arXiv:2608.00222*, 2026. doi: 10.48550/arXiv.2608.00222. URL https://arxiv.org/abs/2608.00222.

Bogdan Georgiev, Javier Gómez-Serrano, Terence Tao, and Adam Zsolt Wagner. Mathematical exploration and discovery at scale. *arXiv preprint arXiv:2511.02864*, 2025. doi: 10.48550/arXiv.2511.02864. URL https://arxiv.org/abs/2511.02864.

Ting Gong, Michael Ruofan Zeng, and Yong Yang. Albilich: Steerable proof-state orchestration for LLM-based mathematical research with CAS integration. *arXiv preprint arXiv:2607.27705*, 2026. doi: 10.48550/arXiv.2607.27705. URL https://arxiv.org/abs/2607.27705.

Google. Gemini 3 developer guide. Gemini API documentation, 2026. URL https://ai.google.dev/gemini-api/docs/generate-content/gemini-3. Accessed September 25, 2026.

Google DeepMind. Gemini 3 Deep Think: Model evaluation—approach, methodology & results. Technical report, Google DeepMind, February 2026. URL https://storage.googleapis.com/deepmind-media/gemini/gemini_3_deep_think_model_evaluation.pdf.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Petar Sirkovic, Artiom Myaskovsky, Grzegorz Glowaty, Felix Weissenberger, Alessio Orlandi, Dan Popovici, et al. Accelerating scientific discovery with Co-Scientist. *Nature*, 655:487–496, 2026. doi: 10.1038/s41586-026-10644-y. URL https://www.nature.com/articles/s41586-026-10644-y.

GPT-5.4 Pro. A proof for the even closed-form rule in Claude's cycles, 2026. URL https://cs.stanford.edu/~knuth/even_closed_form_proof_final.pdf. March 6, 2026.

Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z. Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, et al. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, 651:607–613, 2026. doi: 10.1038/s41586-025-09833-y. URL https://www.nature.com/articles/s41586-025-09833-y.

Rajesh Jayaram. Multi-vector embeddings are provably more expressive than single vector embeddings. *arXiv preprint arXiv:2606.23475*, 2026. URL https://arxiv.org/abs/2606.23475.

Rajesh Jayaram, Honghao Lin, Vahab Mirrokni, and David P. Woodruff. Near-optimal dimension lower bounds for single-vector embeddings of maximum inner product similarity. *arXiv preprint arXiv:2607.20393*, 2026a. URL https://arxiv.org/abs/2607.20393.

Rajesh Jayaram, Drew Tyler, David Woodruff, Corinna Cortes, Yossi Matias, Vahab Mirrokni, and Vincent Cohen-Addad. Towards automating scientific review with Google's paper assistant tool. *arXiv preprint arXiv:2606.28277*, 2026b. URL https://arxiv.org/abs/2606.28277.

Haocheng Ju, Guoxiong Gao, Jiedong Jiang, Bin Wu, Zeming Sun, Shurui Liu, et al. Automated conjecture resolution with formal verification. *arXiv preprint arXiv:2604.03789*, 2026. doi: 10.48550/arXiv.2604.03789. URL https://arxiv.org/abs/2604.03789.

Donald E. Knuth. Claude's cycles. Stanford Computer Science Department technical note, 2026. URL https://cs.stanford.edu/~knuth/papers/claude-cycles.pdf. Revised April 14, 2026.

Han Li, Jinyu Tian, Rili Feng, Yuqiao Du, Chong Zheng, Chenyu Wang, Chenchen Liu, Shihao Li, Xinping Lei, Yifan Yao, et al. Solvita: Enhancing large language models for competitive programming via agentic evolution. *arXiv preprint arXiv:2605.15301*, 2026a. doi: 10.48550/arXiv.2605.15301. URL https://arxiv.org/abs/2605.15301.

Xiaoya Li, Guoyin Wang, Songqiao Su, Chris Shum, and Jiwei Li. GrandCode: Achieving grandmaster level in competitive programming via agentic reinforcement learning. *arXiv preprint arXiv:2604.02721*, 2026b. doi: 10.48550/arXiv.2604.02721. URL https://arxiv.org/abs/2604.02721.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with AlphaCode. *Science*, 378(6624):1092–1097, 2022. doi: 10.1126/science.abq1158. URL https://arxiv.org/abs/2203.07814.

Honghao Lin, Vahab Mirrokni, and David P. Woodruff. The condition-number barrier in sparse least squares. *arXiv preprint arXiv:2608.02588*, 2026a. URL https://arxiv.org/abs/2608.02588.

Honghao Lin, Vahab Mirrokni, and David P. Woodruff. Nearly optimal strong coresets for ℓ_p subspace approximation. *arXiv preprint arXiv:2608.26047*, 2026b. URL https://arxiv.org/abs/2608.26047.

Honghao Lin, Vahab Mirrokni, and David P. Woodruff. Pairwise-independent dithering for single-stage Hadamard quantization. *arXiv preprint arXiv:2608.02564*, 2026c. URL https://arxiv.org/abs/2608.02564.

Honghao Lin, Vahab Mirrokni, and David P. Woodruff. A near-optimal lower bound for prefix-matrix factorizations. *arXiv preprint arXiv:2608.08238*, 2026d. URL https://arxiv.org/abs/2608.08238.

Jihao Liu, Guoxiong Gao, Zeming Sun, Bin Wu, Shurui Liu, Jiedong Jiang, Haocheng Ju, Leheng Chen, Ronnie Cheng, Xiping Zhang, and Bin Dong. Danus: Orchestrating mathematical reasoning agents with fact-graph memory. *arXiv preprint arXiv:2607.06447*, 2026. doi: 10.48550/arXiv.2607.06447. URL https://arxiv.org/abs/2607.06447.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024. URL https://arxiv.org/abs/2408.06292.

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. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-2019. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025. URL https://arxiv.org/abs/2506.13131.

OpenAI. Our First Proof submissions. OpenAI Research, February 2026a. URL https://openai.com/index/first-proof-submissions/. Published February 20, 2026.

OpenAI. Ten advances in mathematics and theoretical computer science. OpenAI Publication, August 2026b. URL https://openai.com/index/ten-advances-in-mathematics/. Published August 1, 2026.

Shanghaoran Quan, Jiaxi Yang, Bowen Yu, Bo Zheng, Dayiheng Liu, An Yang, Xuancheng Ren, Bofei Gao, et al. CodeElo: Benchmarking competition-level code generation of LLMs with human-comparable Elo ratings. *arXiv preprint arXiv:2501.01257*, 2025. doi: 10.48550/arXiv.2501.01257. URL https://arxiv.org/abs/2501.01257.

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*, 625:468–475, 2024. doi: 10.1038/s41586-023-06924-6. URL https://www.nature.com/articles/s41586-023-06924-6.

Johannes Schmitt, Tim Gehrunger, Jasper Dekoninck, Gergely Bérczi, Uri Kreitner, Liam Price, and David Holmes. ProofCouncil: An LLM agent for solving open mathematical problems. *arXiv preprint arXiv:2607.09474*, 2026. doi: 10.48550/arXiv.2607.09474. URL https://arxiv.org/abs/2607.09474.

Shiven Sinha, Shashwat Goel, Ponnurangam Kumaraguru, Jonas Geiping, Matthias Bethge, and Ameya Prabhu. Can language models falsify? evaluating algorithmic reasoning with counterexample creation. In *Second Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2502.19414.

Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625:476–482, 2024. doi: 10.1038/s41586-023-06747-5. URL https://www.nature.com/articles/s41586-023-06747-5.

George Tsoukalas, Anton Kovsharov, Sergey Shirobokov, Anja Surina, Moritz Firsching, Gergely Bérczi, et al. Advancing mathematics research with AI-driven formal proof search. *arXiv preprint arXiv:2605.22763*, 2026. doi: 10.48550/arXiv.2605.22763. URL https://arxiv.org/abs/2605.22763.

Peisong Wang, Bowen Liu, Zehua Li, Yuyao Wang, Zhiwei Ma, Yuhan Li, and Jia Li. CP-Agent: A calibrated risk-controlled agent for feedback-driven competitive programming. *arXiv preprint arXiv:2605.24693*, 2026. doi: 10.48550/arXiv.2605.24693. URL https://arxiv.org/abs/2605.24693.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

David P. Woodruff and Taisuke Yasuda. Root ridge leverage score sampling for ℓ_p subspace approximation. In *Proceedings of the 66th Annual IEEE Symposium on Foundations of Computer Science*, 2025. URL https://arxiv.org/abs/2407.03262.

David P. Woodruff, Vincent Cohen-Addad, Lalit Jain, Jieming Mao, Song Zuo, MohammadHossein Bateni, Simina Branzei, Michael P. Brenner, Lin Chen, Ying Feng, et al. Accelerating scientific research with Gemini: Case studies and common techniques. *arXiv preprint arXiv:2602.03837*, 2026. URL https://arxiv.org/abs/2602.03837.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=VNckp7JEHn.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025. doi: 10.48550/arXiv.2504.08066. URL https://arxiv.org/abs/2504.08066.

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. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2305.10601.

Yuanhe Zhang, Yuekai Sun, Taiji Suzuki, Jason D. Lee, and Fanghui Liu. LeanMarathon: Toward reliable AI co-mathematicians through long-horizon Lean autoformalization. *arXiv preprint arXiv:2606.05400*, 2026. doi: 10.48550/arXiv.2606.05400. URL https://arxiv.org/abs/2606.05400.

Zelin Zhao, Bo Yuan, Jaemoo Choi, and Yongxin Chen. RMA: An agentic system for research-level mathematical problems. *arXiv preprint arXiv:2605.22875*, 2026. doi: 10.48550/arXiv.2605.22875. URL https://arxiv.org/abs/2605.22875.

Daniel Zheng, Ingrid von Glehn, Yori Zwols, Iuliya Beloshapka, Lars Buesing, Daniel M. Roy, Martin Wattenberg, Bogdan Georgiev, Tatiana Schmidt, Andrew Cowie, et al. AI co-mathematician: Accelerating mathematicians with agentic AI. *arXiv preprint arXiv:2605.06651*, 2026. doi: 10.48550/arXiv.2605.06651. URL https://arxiv.org/abs/2605.06651.

Zihan Zheng, Zerui Cheng, Zeyu Shen, Shang Zhou, Kaiyuan Liu, Hansen He, Dongruixuan Li, Stanley Wei, et al. LiveCodeBench Pro: How do olympiad medalists judge LLMs in competitive programming? In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3502. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/9712b78386cebdc3db7f1a48c2d20edb-Abstract-Datasets_and_Benchmarks_Track.html.

Shang Zhou, Wenhao Chai, Kaiyuan Liu, Huanzhi Mao, Qiuyang Mang, and Jingbo Shang. OpenDeepThink: Parallel reasoning via Bradley–Terry aggregation. *arXiv preprint arXiv:2605.15177*, 2026. doi: 10.48550/arXiv.2605.15177. URL https://arxiv.org/abs/2605.15177.

A Proof of the Copy-Complexity Theorem

A.1 Model and notation

Fix integers $c \geq 2$ and $m, k, h \geq 1$ with $h \leq m$ and $k \leq mc$, and let $\alpha \in (0, 1]$ be the probability that a node retains the evidence. All these parameters are held fixed as the depth L grows.

Index layers *backward* from the terminal block:

$$W_0, W_1, \ldots, W_L, \qquad |W_t| = N_t = mc^t.$$

Thus W_L is the source layer and W_0 is the final layer before the terminal merge. Each node $v \in W_t$, $0 \leq t < L$, selects k distinct nodes from W_{t+1}, with every such subset equally likely. We call these nodes its parents and denote their set by $\mathcal{N}(v)$. Also draw a bit B_v, equal to one with probability α and zero otherwise. All parent sets and retention bits are mutually independent. A node v contains the evidence exactly when $B_v = 1$ and at least one of its sampled parents contains it. Fix a set $\mathcal{T} \subseteq W_0$ of size h. The terminal merge contains the evidence if at least one node of $\mathcal{T}$ contains it.

Initially, the evidence is placed on a uniformly random r-subset $R \subseteq W_L$, independently of the graph and retention bits. Set

$$n = N_L = mc^L, \qquad P_L(r) = \mathbb{P}(\text{the terminal merge contains the evidence}).$$

A fixed parent is sampled by $N_t k / N_{t+1} = k/c$ nodes in expectation. As in the main text, write

$$\beta = \frac{k}{c}, \qquad \lambda = \alpha\beta = \frac{\alpha k}{c}, \qquad \mu = \alpha k = c\lambda. \tag{7}$$

Thus β counts how often a node is used on average. Multiplying by the retention probability gives λ, the expected number of retained copies in the next forward layer when only one node carries the evidence. Looking backward, μ is the expected number of parents contributed by one node after its retention bit is drawn. Several nodes may select the same parent; we call these repeated selections collisions.

A.2 Backward influence sets

For a fixed graph and its retention bits, let I_t be the set of positions in layer t from which evidence would reach the terminal merge. These sets can be found one layer at a time:

$$I_0 = \mathcal{T}, \qquad I_{t+1} = \bigcup_{v \in I_t : B_v = 1} \mathcal{N}(v), \qquad Z_t = |I_t|, \quad Q_t = \frac{Z_t}{N_t}.$$

We call I_t the influence set. Its size is Z_t, and its density Q_t is the fraction of layer t that it occupies. In particular, $Z_0 = h$, and success is exactly the event $R \cap I_L \neq \varnothing$.

The construction of I_t uses only randomness in layers closer to the terminal block. The bits and parent sets of its nodes are therefore still independent, with their original distributions. Conditional on $Z_t = z$, each of the z nodes is retained independently with probability α. Their number S therefore has the binomial distribution $\mathrm{Bin}(z, \alpha)$. Then Z_{t+1} is the size of the union of S independent uniform k-subsets of W_{t+1}. Taking this union accounts for collisions between parent sets.

Lemma 2 (Hitting the influence set)**.** *With the convention* $\binom{a}{r} = 0$ *for integers* $0 \leq a < r$*,*

$$P_L(r) = \mathbb{E}\left[1 - \frac{\binom{n - Z_L}{r}}{\binom{n}{r}}\right]. \tag{8}$$

For every $1 \leq r \leq n$*,*

$$\mathbb{E}[1 - e^{-rQ_L}] \leq P_L(r) \leq \min\{1, r\mathbb{E}Q_L\}. \tag{9}$$

Proof. Condition on the graph and its retention bits. A uniform r-subset misses the fixed set I_L with probability $\binom{n - Z_L}{r} / \binom{n}{r}$, proving (8). Write $q = Z_L/n$. The probability of missing I_L is at most $(1 - q)^r \leq e^{-rq}$: conditional on all previous draws missing I_L, the next draw is at least as likely to hit it as the first. For the upper bound, the expected number of sampled positions in I_L is rq, so the probability of at least one hit is at most $\min\{1, rq\}$. Taking expectations gives (9). □

Lemma 3 (One-step estimates). *For $0 \le t < L$, write $N = N_{t+1} = cN_t$ and $\sigma^2 = k^2\alpha(1-\alpha)$. Conditional on $Z_t = z$,*

$$\mathbb{E}[Z_{t+1} \mid Z_t = z] = N\left[1 - \left(1 - \frac{\mu}{N}\right)^z\right], \tag{10}$$

$$\mu z - \frac{\mu^2 z(z-1)}{2N} \le \mathbb{E}[Z_{t+1} \mid Z_t = z] \le \mu z, \tag{11}$$

$$\mathbb{E}[Z_{t+1}^2 \mid Z_t = z] \le \mu^2 z^2 + \sigma^2 z, \tag{12}$$

$$\mathrm{Var}(Z_{t+1} \mid Z_t = z) \le k^2 z. \tag{13}$$

In density form, with $G_\lambda(x) = 1 - e^{-\lambda x}$,

$$\mathbb{E}[Q_{t+1} \mid Z_t = z] \ge G_\lambda(z/N_t), \tag{14}$$

$$\mathrm{Var}(Q_{t+1} \mid Z_t = z) \le \frac{k^2}{c^2 N_t}. \tag{15}$$

Proof. A given vertex of W_{t+1} is reached by each node of I_t with probability $\alpha k/N = \mu/N$, independently across the z nodes. It therefore belongs to I_{t+1} with probability $1-(1-\mu/N)^z$. Summing over the N vertices gives (10). Adding the individual hit probabilities gives an upper bound on the probability of at least one hit. Subtracting the probabilities of all pairwise hits gives a lower bound. These two counts yield (11).

Since a union has at most kS elements,

$$\mathbb{E}Z_{t+1}^2 \le \mathbb{E}(kS)^2 = (\alpha k z)^2 + k^2\alpha(1-\alpha)z,$$

where expectations in this display are conditional on $Z_t = z$.

To bound the variance, write the independent retained parent sets as $A_1, \dots, A_z$, each either empty or a k-subset, and let $Y = |A_1 \cup \cdots \cup A_z|$. After seeing the first i sets, our conditional estimate of Y is

$$M_i = \mathbb{E}[Y \mid A_1, \dots, A_i], \qquad M_0 = \mathbb{E}Y, \quad M_z = Y.$$

Changing one set changes Y by at most k. The same bound holds after averaging over the sets still unseen, so $|M_i - M_{i-1}| \le k$. Write $D_i = M_i - M_{i-1}$. Given the first $i-1$ sets, D_i has mean zero. For $i < j$, conditioning on the first $j-1$ sets therefore gives $\mathbb{E}[D_i D_j] = 0$. Expanding the square now gives

$$\mathrm{Var}(Y) = \mathbb{E}\left(\sum_{i=1}^{z} D_i\right)^2 = \sum_{i=1}^{z} \mathbb{E}D_i^2 \le zk^2,$$

which proves (13). All expectations in this argument are conditional on $Z_t = z$.

Finally, $(1-\mu/N)^z \le e^{-\mu z/N}$ and $N = cN_t$ imply (14). Dividing (13) by N_{t+1}^2 and using $z \le N_t$ gives (15). □

We will use the variance bound to control the probability of falling below a threshold. For any random variable X with finite variance and any $d > 0$,

$$\mathbb{P}(X \le \mathbb{E}X - d) \le \frac{\mathrm{Var}(X)}{d^2}. \tag{16}$$

Indeed, $(X - \mathbb{E}X)^2 \ge d^2$ on the event on the left, so its expectation is at least d^2 times the probability of that event. The same argument applies to a conditional distribution.

The lower bounds will start from one of two configurations at a fixed depth.

Lemma 4 (Initial configurations). *Fix an integer $t_0 \ge 0$ and let $L \ge t_0$. If $t_0 \ge 1$, then $\mathbb{P}(Z_{t_0} = k) > 0$. If $k \ge c$, then*

$$\mathbb{P}(Q_{t_0} \ge h/m) > 0. \tag{17}$$

These probabilities depend only on t_0 and the fixed model parameters, not on L.

Proof. For the first assertion, at each of the first t_0 steps require every node in the current influence set to retain and to choose the same prescribed k-subset. The influence set then has size k after the first step and at every step through time t_0. There are only finitely many prescribed choices, each with positive probability.

For the second assertion, require all z nodes in the current influence set to retain and choose parent sets whose union has size $\min\{kz, N_{t+1}\}$. Such a choice is possible because $k \le N_{t+1}$: choose disjoint sets if $kz \le N_{t+1}$; otherwise cover all N_{t+1} positions, filling each set to size k with distinct positions. Since $k \ge c$, induction gives, on this event,

$$Z_t = \min\{hk^t, N_t\}, \qquad Q_t = \min\{(h/m)(k/c)^t, 1\} \ge h/m.$$

Imposing these choices for the first t_0 steps has positive probability. Both events involve only the first t_0 layers, so their probabilities are the same for every $L \ge t_0$. □

Lemma 5 (From influence size to success probability)**.** *Suppose $b_L \ge 1$ and, for constants $U, d, p > 0$ independent of L,*

$$\mathbb{E}Q_L \le \frac{U}{b_L}, \qquad \mathbb{P}\left(Q_L \ge \frac{d}{b_L}\right) \ge p.$$

Then for all $1 \le r \le n$,

$$p(1 - e^{-d}) \min\{1, r/b_L\} \le P_L(r) \le \max\{1, U\} \min\{1, r/b_L\}. \tag{18}$$

Proof. Lemma 2 gives the upper bound and also implies $P_L(r) \ge p(1 - e^{-dr/b_L})$. For $0 \le u \le 1$, the function $1 - e^{-du}$ lies above the line joining its values at zero and one, since its second derivative is $-d^2 e^{-du} \le 0$. Hence $1 - e^{-du} \ge u(1 - e^{-d})$. For $u \ge 1$, we have $1 - e^{-du} \ge 1 - e^{-d}$. Taking $u = r/b_L$ proves the lower bound. □

A.3 Polynomially many copies: $1/c < \lambda < 1$

Here $1 < \mu < c$. Iterating the upper bound in (11) gives

$$\mathbb{E}Z_L \le h\mu^L, \qquad \mathbb{E}Q_L \le \frac{h}{m}\lambda^L. \tag{19}$$

For the lower bound, we control the accumulated loss due to collisions. Fix a time $t_0 \ge 1$, to be chosen below, and condition on $Z_{t_0} = z = k$. Write $H = N_{t_0}$. Until stated otherwise, expectations are taken under this conditioning and $0 \le s \le L - t_0$. Set

$$D_z = z^2 + \frac{\sigma^2 z}{\mu(\mu - 1)}.$$

The bounds on the mean and the expected square in Lemma 3 imply

$$\mathbb{E}Z_{t_0+s} \le z\mu^s, \qquad \mathbb{E}Z_{t_0+s}^2 \le D_z \mu^{2s}. \tag{20}$$

Indeed, dividing the recursion (12) by $\mu^{2(s+1)}$ and summing gives

$$\frac{\mathbb{E}Z_{t_0+s}^2}{\mu^{2s}} \le z^2 + \frac{\sigma^2 z}{\mu^2} \sum_{j=0}^{s-1} \mu^{-j} \le z^2 + \frac{\sigma^2 z}{\mu(\mu - 1)}.$$

Let $a_s = \mathbb{E}Z_{t_0+s}/\mu^s$. Using the lower bound in (11), $N_{t_0+s+1} = Hc^{s+1}$, and (20), we obtain, for $s < L - t_0$,

$$\begin{aligned} a_{s+1} &\ge a_s - \frac{\mu^2 \mathbb{E}Z_{t_0+s}^2}{2Hc^{s+1}\mu^{s+1}} \\ &\ge a_s - \frac{D_z \lambda}{2H}\lambda^s. \end{aligned}$$

Since $\lambda < 1$, the losses on the right sum to at most $D_z\lambda/[2H(1 - \lambda)]$. Choose t_0 large enough that

$$H = N_{t_0} \ge \frac{D_z \lambda}{z(1 - \lambda)}. \tag{21}$$

Summing from $a_0 = z$ gives

$$a_s \geq z - \frac{D_z \lambda}{2H(1-\lambda)} \geq \frac{z}{2}, \qquad \mathbb{E}Z_{t_0+s} \geq \frac{z}{2}\mu^s. \tag{22}$$

To turn this mean bound into a probability bound, recall that every nonnegative random variable X with $0 < \mathbb{E}X^2 < \infty$ satisfies

$$\mathbb{P}(X \geq \tfrac{1}{2}\mathbb{E}X) \geq \frac{(\mathbb{E}X)^2}{4\mathbb{E}X^2}. \tag{23}$$

To see this, put $A = \{X \geq \mathbb{E}X/2\}$ and let $\mathbf{1}_A$ be one on A and zero elsewhere. The contribution from outside A is at most $\mathbb{E}X/2$, and therefore

$$\frac{\mathbb{E}X}{2} \leq \mathbb{E}[X\mathbf{1}_A] \leq \sqrt{\mathbb{E}X^2}\sqrt{\mathbb{P}(A)}.$$

For the last inequality, expand $\mathbb{E}[(X - u\mathbf{1}_A)^2] \geq 0$ and take $u = \mathbb{E}[X\mathbf{1}_A]/\mathbb{P}(A)$; here $\mathbb{P}(A) > 0$ because $\mathbb{E}X > 0$. Squaring gives (23). Applying (23) with (20)–(22) gives

$$\mathbb{P}\Big(Z_{t_0+s} \geq \frac{z}{4}\mu^s \,\Big|\, Z_{t_0} = z\Big) \geq \frac{z^2}{16D_z}.$$

Now remove the conditioning. Lemma 4 gives $\eta = \mathbb{P}(Z_{t_0} = z) > 0$, independently of L. Taking $s = L - t_0$ and dividing by $N_L = mc^L$, we obtain, for every $L \geq t_0$,

$$\mathbb{P}\big(Q_L \geq d\lambda^L\big) \geq p, \qquad d = \frac{z}{4m\mu^{t_0}}, \quad p = \eta\frac{z^2}{16D_z} > 0. \tag{24}$$

Applying Lemma 5 to (19) and (24) proves (3) with $b_L = \lambda^{-L}$.

A.4 Logarithmically many copies: $\lambda = 1$

Here $\mu = c$. Without repeated parents, the expected density would stay the same from one layer to the next. We show that the loss from repeated parents makes it decrease at rate $1/(t+1)$.

A.4.1 Upper bound

Since $\mu/N_{t+1} = 1/N_t$, the exact conditional mean is

$$\mathbb{E}[Q_{t+1} \mid Z_t = z] = 1 - (1 - 1/N_t)^z.$$

For integers $0 \leq z \leq M$, count individual hits, subtract pairwise hits, and add triple hits. This gives the upper bound

$$\begin{aligned} 1 - (1 - 1/M)^z &\leq \frac{z}{M} - \frac{z(z-1)}{2M^2} + \frac{z(z-1)(z-2)}{6M^3} \\ &\leq \frac{z}{M} - \frac{z(z-1)}{3M^2}. \end{aligned} \tag{25}$$

For $z = 0, 1$ the inequality holds directly; for $z \geq 2$, use $(z-2)/M \leq 1$. Writing $u_t = \mathbb{E}Q_t$, substituting $M = N_t$, and using $\mathbb{E}Q_t^2 \geq (\mathbb{E}Q_t)^2$, we obtain

$$u_{t+1} \leq \left(1 + \frac{1}{3N_t}\right) u_t - \frac{1}{3}u_t^2. \tag{26}$$

To remove the factor $1 + 1/(3N_t)$ from the linear term, set

$$A_t = \prod_{j=0}^{t-1}\left(1 + \frac{1}{3N_j}\right), \qquad v_t = \frac{u_t}{A_t}.$$

Since $1 + x \leq e^x$ and $\sum_{j \geq 0} 1/N_j = c/[m(c-1)]$,

$$1 \leq A_t \leq A_* := \exp\left(\frac{c}{3m(c-1)}\right). \tag{27}$$

Using $A_t \geq 1$ and $1/(3N_t) \leq 1/3$, equation (26) becomes

$$v_{t+1} \leq v_t - \frac{A_t}{3(1+1/(3N_t))} v_t^2 \leq v_t - \frac{1}{4} v_t^2.$$

For each finite t, all nodes in the first t layers retain with positive probability, giving $Z_t > 0$ on this event and hence $v_t > 0$; also $v_t \leq 1$. Taking reciprocals yields

$$\frac{1}{v_{t+1}} \geq \frac{1}{v_t(1 - v_t/4)} \geq \frac{1}{v_t} + \frac{1}{4}.$$

Since $v_0 = h/m$, it follows that

$$\mathbb{E}Q_t = u_t \leq \frac{A_*}{m/h + t/4} \leq \frac{4A_*}{t+1}. \tag{28}$$

A.4.2 Lower bound

Set $a = h/(2m) \in (0, 1/2]$. Since $\alpha k = c$ and $\alpha \leq 1$, we have $k \geq c$, and Lemma 4 gives a positive probability of reaching density at least a at any fixed time t_0. We will choose t_0 so that, conditional on reaching this density, the process has probability at least $1/2$ of satisfying

$$Q_{t_0+s} \geq \frac{a}{s+1} \tag{29}$$

for all $0 \leq s \leq L - t_0$. For a single step with $s < L - t_0$, put $w = s + 1$. If (29) holds, $G_1(Q_{t_0+s}) \geq G_1(a/w)$ because G_1 is increasing. Using $1 - e^{-x} \geq x - x^2/2$ gives

$$\begin{aligned} \mathbb{E}[Q_{t_0+s+1} \mid Z_{t_0+s}] - \frac{a}{w+1} &\geq \frac{a}{w} - \frac{a^2}{2w^2} - \frac{a}{w+1} \\ &= \frac{a}{w(w+1)} - \frac{a^2}{2w^2} \\ &\geq \frac{a}{4w^2}. \end{aligned}$$

The last inequality uses $w + 1 \leq 2w$ and $a \leq 1/2$. Thus the conditional mean exceeds the next threshold by at least $a/(4w^2)$. Applying (16) with this gap and the variance bound (15) gives

$$\mathbb{P}\left(Q_{t_0+s+1} < \frac{a}{s+2} \,\middle|\, Z_{t_0+s} \right) \leq \frac{16k^2(s+1)^4}{c^2 a^2 N_{t_0+s}} \tag{30}$$

uniformly over states satisfying (29). The bound also holds conditional on all earlier influence sets, since the next step uses independent sampling and retention decisions. Because $N_{t_0+s} = N_{t_0} c^s$ and the series $\sum_{s \geq 0} (s+1)^4 c^{-s}$ converges, choose t_0 large enough that

$$\frac{16k^2}{c^2 a^2 N_{t_0}} \sum_{s=0}^{\infty} (s+1)^4 c^{-s} \leq \frac{1}{2}. \tag{31}$$

Let $\eta = \mathbb{P}(Q_{t_0} \geq a) > 0$. Conditional on $Q_{t_0} \geq a$, consider the first step at which (29) fails. Up to that step the bound (30) applies, so summing over the possible first failures gives a total probability at most $1/2$. This choice of t_0 is independent of L. It follows that, for every $L \geq t_0$,

$$\mathbb{P}\left(Q_L \geq \frac{a}{L - t_0 + 1} \right) \geq \frac{\eta}{2}, \qquad \mathbb{P}\left(Q_L \geq \frac{a}{L+1} \right) \geq \frac{\eta}{2}. \tag{32}$$

Combining (28) and (32) with Lemma 5 proves (3) with $b_L = L + 1$.

A.5 A constant number of copies: $\lambda > 1$

When $\lambda > 1$, the same argument works with a fixed density threshold. Choose

$$a = \min\left\{ \frac{h}{2m}, \frac{\lambda - 1}{\lambda^2} \right\} > 0, \qquad \delta = \frac{(\lambda - 1)a}{2} > 0.$$

Using $1 - e^{-x} \geq x - x^2/2$ again,

$$G_\lambda(a) \geq \lambda a - \frac{\lambda^2 a^2}{2} \geq a + \delta. \tag{33}$$

For $Q_t \geq a$, the conditional mean of Q_{t+1} is therefore at least $a + \delta$. Applying (16) with gap δ and using (15) gives

$$\mathbb{P}(Q_{t+1} < a \mid Z_t) \leq \frac{k^2}{c^2 \delta^2 N_t}. \tag{34}$$

As above, this bound holds conditional on all earlier influence sets because the next step uses independent randomness. Choose t_0, independently of L, so that

$$\frac{k^2}{c^2 \delta^2 N_{t_0}} \sum_{s=0}^{\infty} c^{-s} \leq \frac{1}{2}.$$

Since $\lambda > 1$ implies $k > c$, Lemma 4 gives $\eta = \mathbb{P}(Q_{t_0} \geq a) > 0$. Conditional on this event, summing (34) over the possible first steps at which the density falls below a gives

$$\mathbb{P}(Q_L \geq a) \geq \frac{\eta}{2} \qquad (L \geq t_0). \tag{35}$$

Since $\mathbb{E}Q_L \leq 1$, Lemma 5 proves (3) with $b_L = 1$.

A.6 Minimum number of initial copies

We have proved (3) in each case, with constants depending only on the fixed parameters. Fix $0 < \varepsilon < C_-$, and let $r_\varepsilon(L)$ be the smallest number of copies for which $P_L(r) \geq \varepsilon$. Here C_- is a guaranteed success level, while loss at all h terminal nodes gives $P_L(r) \leq 1 - (1-\alpha)^h < 1$ when $\alpha < 1$ and $L \geq 1$.

In all three cases, $b_L \geq 1$ and $n/b_L \to \infty$. When $1/c < \lambda < 1$,

$$\frac{n}{b_L} = mc^L \lambda^L = m\mu^L \longrightarrow \infty;$$

in the other two cases this follows from exponential growth of n. Hence $\lceil b_L \rceil \leq n$ for all sufficiently large L. Equation (3) gives

$$P_L(\lceil b_L \rceil) \geq C_- > \varepsilon, \qquad P_L(r) \leq C_+ r / b_L.$$

Thus $\lceil b_L \rceil$ copies suffice, while any number achieving success probability at least ε must satisfy $r \geq \varepsilon b_L / C_+$. Hence

$$\frac{\varepsilon}{C_+} b_L \leq r_\varepsilon(L) \leq \lceil b_L \rceil \leq 2 b_L. \tag{36}$$

Finally, $L = \log_c(n/m)$, so

$$\lambda^{-L} = (n/m)^{\log_c(1/\lambda)}, \qquad L + 1 = \Theta(\log n).$$

Since m is fixed, these are the three orders in (4), proving Theorem 1. □

B Extended Method

B.1 Candidate Generation and Targeted Falsification

At a difficult stage, COLOSSEUM generates a population of candidates, subjects each candidate to targeted falsification, and combines the candidates and their critiques through a tree, as described in Section 3.3.

B.1.1 PARALLEL CANDIDATE GENERATION

Parallel generation seeks differences that can change the mathematical outcome: representation, principal lemma, proof technique, case split, interpretation of evidence, or location of the main bottleneck. Seeds, temperatures, prompt perspectives, tool access, and assumed lines of attack provide additional sources of variation.

Each candidate follows a typed schema appropriate to its stage. A strategy proposal, for example, states its mechanism, required lemmas, expected bottleneck, and a falsifiable test. A proof proposal states its assumptions and marks any gaps. The schemas make candidates comparable and give reviewers specific claims to attack.

Selected prompt templates and their structured output interfaces are provided in Appendix C.

B.1.2 TARGETED FALSIFICATION

Each candidate is paired with a falsifier. The falsifier concentrates on finding defects and proposes an alternative solution only when it helps establish one. The review covers:

- counterexamples and boundary cases;
- invalid implications or silently strengthened hypotheses;
- circularity and undeclared dependencies;
- misuse of a theorem, computation, or external reference;
- a mismatch between the proved statement and the target claim; and
- missing assumptions or results needed by later sections.

Falsification records remain attached to the candidate. A precise objection can make another candidate easier to repair. Each candidate produced during aggregation is also paired with a falsifier, as in Section 3.3.

The full tree-structured aggregation procedure, including the sampling rule, population update, and expected reuse, is presented in Section 3.3.

B.2 GLOBAL VERIFICATION AND FEEDBACK

The global verifier reads the original problem and the completed document as a single argument, with the local subproblem reviews available as audit context. A collection of individually plausible sections may still fail as a proof because a dependency is used with the wrong assumptions, notation or definitions drift across sections, a case is omitted, or the final conclusion does not match the original target. The verifier also checks that a conditional or attacked claim from a local review has not been silently inherited by the assembled proof.

Global verification is itself carried out by multiple independent reviews and tree-structured aggregation. The aggregation merges genuinely duplicate criticisms while retaining distinct substantive objections. It is not a majority vote: a concrete fatal defect is sufficient to reject the proof, and generic acceptance judgments do not resolve it.

The final review gives an overall verdict together with the material defects found in the argument. Each defect is tied to the section or claim where it arises; for a cross-section dependency error, the review identifies both the supporting section and the section that uses it. This localization makes the global review actionable without reducing it to a collection of independent section checks.

When the verifier rejects a proof, the pipeline proceeds by revision or re-exploration. Revision is used when the current strategy remains viable; it may replace section bodies or modify the proof outline before the entire argument is verified again. If the verifier's findings undermine the central strategy, the system returns to exploration instead.

B.3 SHARED RESEARCH KNOWLEDGE

COLOSSEUM carries information across research rounds in two complementary forms: the latest proof attempt is retained in full, while a shared knowledge directory collects reusable findings from

the wider search. The former grounds the next round in the argument that was actually attempted; the latter preserves useful results that may otherwise be lost as candidates are aggregated or discarded.

B.3.1 Retaining Prior Attempts

When a round does not produce an accepted proof, COLOSSEUM passes the resulting draft and the verifier feedback directly into the next round. Revision can address the concrete defects identified in the draft, while re-exploration can reconsider the strategy in light of the argument that failed. The next attempt therefore does not begin from the original problem alone.

Retaining a draft does not endorse its claims. The verifier feedback remains attached to it, making clear which steps or strategic assumptions require further work.

B.3.2 Knowledge Directory

The latest draft is only one product of a much wider search. A knowledge curator reads strategy proposals and falsification reports and records four kinds of reusable knowledge:

1. **theorems and lemmas**: mathematical results developed during the search, together with their hypotheses, supporting arguments, and possible applications;
2. **failed approaches**: attempted routes, their precise failure points, and conditions under which a variant might still work;
3. **references**: relevant literature, including the statements and hypotheses needed for the current problem; and
4. **observations**: structural properties or computational findings, together with their evidence and implications for subsequent work.

Entries retain their source and relevant caveats. The directory is updated across research rounds and made available to subsequent agents, allowing them to reuse earlier results and avoid repeating approaches whose failure has already been identified.

B.4 Design Motivation

The design of COLOSSEUM is motivated by four challenges in long-horizon research:

- **Strategic Uncertainty.** A precise problem statement may leave the route to a proof unclear. Finding useful representations, reductions, or intermediate targets often requires substantial exploration, and a promising route may remain uncertain until its central claims are tested.
- **Distributed Technical Difficulty.** A proof may contain several interdependent bottlenecks, from constructing a new object to establishing a delicate estimate. Resolving one difficulty can expose another, and a local revision may require changes to the surrounding argument.
- **Long Outputs and Error Accumulation.** Long proofs can exceed the output budget of a single response, leading to omitted details or unfinished arguments. Definitions and assumptions must also remain consistent across distant sections, since an unsupported step can be reused and propagate errors through the rest of the proof.
- **Failure and Partial Progress.** Failed attempts may still yield useful counterexamples, restrictions, or intermediate results. To guide later work, this information must be recorded as specific mathematical claims or objections; a generic failure judgment does not explain what remains valid or what should change.

C Selected Prompt Templates and Information Flow

The templates below are shortened versions of the prompts used by the harness. They show what each stage does, what information it receives, and what it passes to later stages. Repeated instructions and implementation details are omitted. Items in braces are filled in by the harness at runtime. The prompts use *falser* for the falsifier described in Section 3.3.

C.1 STRATEGY EXPLORATION

Strategy exploration combines an explorer, a paired falser, an exploration aggregator, and a readiness gate. The gate approves a strategy for decomposition, allows a stable route to proceed with explicit proof obligations, or returns it for further exploration.

Explorer

```
Input:
  {problem}
  {knowledge directory, when available}
  {exploration round}
  {previous strategy, falser report, and readiness-gate report, when
   available}
  {previous proof and verifier feedback, when available}

Develop one promising high-level solution strategy before
    decomposition.
Return one strategy card, not a proof, section plan, or LaTeX document.

Normalize the target, state a candidate reduction and core mechanism,
    and
identify the first hard obstruction.

Maintain exactly one concrete primary route and at most two genuinely
distinct backup routes. Preserve an affirmative route unless a
    reproducible
counterexample or symbolic contradiction already refutes the target.
    For any
gateway test, state both the proof route if it succeeds and the
counterexample route if it fails.

Distinguish proved facts from partial, heuristic, or unverified claims.

Record risky lemmas, adversarial notes, sanity checks, and every
    blocking
doubt as an active obligation. Treat the strategy as ready for
decomposition only when no unresolved fatal obligation remains.

Return a strategy card that records the normalized problem, the
    primary
and backup routes, their supporting evidence and risky lemmas, the
    first
hard obstruction, active obligations, a concrete next task, and a
readiness assessment.
```

Exploration Falser

```
Input:
  {problem}
  {knowledge directory, when available}
  {strategy card, including executed code-probe result when available}
  {previous proof and verifier feedback, when available}
  {falser reports inherited from child nodes, when available}

Attack the supplied strategy card. Do not rewrite, polish, or defend
    it,
except to name a minimal weakening. Examine the primary route first
    and
then each backup route.
```

Exploration Falser (continued)

```
Test the weakest central lemmas, hidden assumptions, overstrong claims,

boundary or counterexample regimes, dropped objections, theorem
hypotheses, inequality directions, and any mismatch between a code
    probe
and the claim it is said to test. For each serious issue, explain how
    the
route could fail and give a cheap decisive test when possible. A
    failed
program is missing evidence, not a mathematical counterexample;
    reserve
falsified for a valid argument or reproducible test.

Return an overall assessment, categorized objections, cheap
    falsification
tests, unresolved objections that later stages must preserve, minimal
weakenings to try, and a verdict indicating whether the route survives,

requires weakening or repair, should be rejected, or has been
    falsified.
```

Exploration Aggregator

```
Input:
  {problem}
  {knowledge directory, when available}
  {(strategy card, falser report) pairs sampled at this tree node}
  {previous strategy, falser report, and readiness-gate report, when
   available}
  {previous proof and verifier feedback, when available}

Produce one new strategy card, not a list and not a decomposition. Do
    not
average the children or merely select one of them. Preserve the
    narrowest
concrete, repairable route as the single primary route unless its core
mechanism has been killed. Retain at most two nonduplicative backup
    routes,
including a useful minority route when warranted.

Address every serious falser objection: reject, weaken, or repair the
attacked claim; refute the objection with a specific mathematical
    reason;
or carry the unresolved objection forward. Preserve unresolved
    blocking
obligations without recursively retelling the full history. Normalize
notation, preserve both branches of conditional gateways, and treat
computational results only at the strength actually tested.

Return a fresh strategy card together with an updated readiness
    judgment.
A route with an unresolved fatal obligation is not ready for
    decomposition.
```

Readiness Gate

```
Input:
  {problem}
```

Readiness Gate (continued)

```
  {knowledge directory, when available}
  {final strategy card}
  {attached falser report, when available}
  {previous proof and verifier feedback, when available}

Do not solve the problem or repair the strategy. Decide whether the
    card
is safe to turn into a proof decomposition. Audit every fatal bridge
claim, cited theorem and hypothesis, target equivalence, induction or
construction invariant, compatibility condition, hidden hard step,
bound direction, unresolved falser objection, and central code result.

Classify the strategy as ready for decomposition, eligible for
    decomposition
with explicit obligations, in need of further exploration, or to be
    rejected.

A strategy with an unresolved fatal bridge cannot be marked ready.
A strategy without such a bridge may proceed to decomposition, but
    only
if the route and proof architecture are stable and every remaining
obligation has a concrete proof or verification path. Treat these as
minor obligations and assign each to an explicit proof or certificate
section in the proof plan.

An obligation is major if resolving it could require changing the
    route,
reduced target, or core mechanism, or if it has no clear repair path.
In such cases, return the strategy for further exploration.

Return the blocking issues, fatal bridge claims, theorem and criterion
audits, hidden hard steps, minimal remaining obligations, obligation
severity, whether decomposition is allowed, and the recommended next
    action.
```

C.2 Decomposition into Subproblems

The decomposer turns an approved strategy into a LaTeX skeleton and a DAG of section-level subproblems. Each subproblem lists its earlier dependencies, allowing eligible independent sections to be solved in parallel.

Decomposer

```
Input:
  {problem}
  {knowledge directory, when available}
  {approved strategy card}
  {falser and readiness-gate reports}
  {previous decomposition, proof, and verifier feedback, when
    available}

Break the original problem into a short list of subproblems in
    topological
order. For each subproblem, list the indices of all earlier
    subproblems on
which it depends. Each subproblem should be mathematically meaningful,
nontrivial, and useful for reaching the final proof. Avoid tiny
    bookkeeping
steps.
```

Decomposer (continued)

```
Produce a standalone LaTeX master-document skeleton with a title, a
concise abstract describing the proof plan, and one section per
subproblem. Inside each section, write a short, mathematically
    specific
roadmap and reserve a unique location for the future section body.

Return the master-document skeleton together with the same ordered
    list
of subproblem titles, tasks, and dependency indices. Do not silently
    build
on unresolved attacked claims. Isolate them as explicit subproblems,
    weaken
them, or route around them.
```

A decomposition aggregator combines candidate skeletons and their subproblem DAGs into one plan. Unresolved bridge lemmas and certificate obligations must remain explicit in that plan.

C.3 Subproblem Solving and Aggregation

For each subproblem, multiple solvers produce candidate sections, paired falsers critique them, and aggregators combine the candidates through a tree. On retry, the previous attempt and its criticism are supplied so useful progress is retained without hiding unresolved objections.

Subproblem Solver

```
Input:
  {problem and assigned subproblem}
  {knowledge directory, when available}
  {current master document}
  {previous proof and global verifier feedback, when available}
  {previous attempt and attached criticism, on retry}
  {executed code-probe result, when available}

Write the full LaTeX body that replaces this section's placeholder. Do
not write a standalone document or repeat earlier sections. You may
    rely
on prior sections only to the extent justified by their text. Address
every relevant error already identified by global feedback or a
    previous
falser. If the section remains incomplete, mark the gap explicitly.

State exactly what theorem, lemma, reduction, or partial result the
section establishes. Separately identify the bridge claims on which it
depends and every obligation that remains unproved. Mark the section
    as
solved, partial, or blocked, and request a retry only for a concrete
remaining defect.

Only an executed code probe counts as computational evidence. If a
necessary certificate has not been executed, state only the reduction,
conditional result, or certificate plan and retain the stronger claim
    as
an open obligation.
```

Subproblem Falser

```
Input:
```

Subproblem Falser (continued)

```
  {original problem}
  {assigned subproblem}
  {knowledge directory, when available}
  {current master document}
  {proposed section solution}
  {previous falser report, when available}

Do not repair or rewrite the section. Decide whether it actually
    proves
the assigned subproblem. Focus first on a fatal bridge: a false
equivalence, wrong target, wrong-sided bound, dropped factor, missing
constraint, index mismatch, unsupported theorem use, or code-probe
misuse. Audit every explicit open obligation and preserve earlier
objections unless the new section resolves them.

Classify the section as ready, conditional, or rejected. Ready means
later assembly may rely on the claimed result. Conditional means
    useful
progress remains but an explicit bridge or obligation is unresolved.
Reject a section that proves the wrong target or relies on a false or
constraint-dropping bridge. Any explicit gap precludes a ready verdict.


Return a concise summary, the fatal objections, and the claims that
    later
stages must not treat as established.
```

Subproblem Aggregator

```
Input:
  {problem and assigned subproblem}
  {knowledge directory, when available}
  {(section solution, falser report) pairs for one subproblem}
  {current master document}
  {previous proof and global verifier feedback, when available}

Synthesize the strongest single section replacement. Consume both each
candidate solution and its attached falser report. Do not silently
inherit an attacked claim: reject it, weaken it, repair it, or retain
    it
as an open obligation.

The resulting section must remain compatible with the notation,
assumptions, definitions, and established claims in the already-filled
master document. It must not contain a section heading or document
preamble. Preserve incomplete but useful progress only with an
    explicit
gap and a status of partial or blocked.

Return the synthesized section together with its claimed result,
    essential
bridge claims, unresolved obligations, completion status, and any
    concrete
reason that another attempt is needed.
```

C.4 Global Verification and Revision

The global verifier reviews the assembled proof together with compact solver–falser audit cards (summaries of the local subproblem reviews). If the verifier rejects the proof, its feedback guides

targeted outline or section revision, after which the proof is verified again. If the verifier's findings undermine the central strategy, the system returns to exploration instead.

Global Verifier

```
Input:
  {original problem}
  {knowledge directory, when available}
  {complete assembled proof}
  {subproblem solver-falser audit cards}

Evaluate the proof against the original problem, not merely for
    internal
consistency. Read the entire proof from beginning to end even if a
    fatal
gap appears early. Check the exact hypotheses, scope, and conclusion,
    and
audit theorem uses, citations, computations, boundary cases, and every
claim attacked or left conditional in a subproblem audit card. A ready
local verdict is not itself a proof.

If the proof is complete, accept it and preserve the complete document.

Otherwise reject it and write a standalone, actionable feedback
    document
that localizes every material defect.

Verifier code probes are adversarial checks only. Use them to test a
claimed identity, bound, computation, or boundary case, never to
    repair
the proof.
```

Revision-Specific Instructions

```
Input:
  {original problem}
  {knowledge directory, when available}
  {current proof, decomposition, and ordered subproblems}
  {global verifier feedback}
  {original section and task, for section revision}

First identify every concrete mathematical issue in the global review.
For the outline cleanup, the allowed changes are to rewrite abstract
    or
roadmap material, slightly revise section names and tasks, and delete
redundant, empty, harmful, or superseded sections. Do not add sections,

invent placeholder tokens, reorder the surviving plan, or introduce a
new proof strategy.

For a section revision, fix every review issue that affects that
    section.
Keep material that is already correct and do not rewrite merely for
style. Preserve the paragraph order, displayed equations, named
environments, and LaTeX labels unless the review gives a concrete
mathematical reason to change them. Mark any remaining gap explicitly.

Return either a conservatively revised outline or a revised section
    body,
as appropriate. Candidate revisions are aggregated before the proof is
reassembled and verified again.
```

D Research Results and Extended Case Studies

We summarize five research results to which runs of the workflow contributed. Each subsection states the motivating question and principal advance; the cited papers provide complete definitions, attribution, and proofs.

D.1 Strong Coresets for ℓ_p Subspace Approximation When $p > 2$

Given a matrix $A \in \mathbb{R}^{n\times d}$, the ℓ_p subspace approximation problem asks for a low-dimensional subspace that minimizes the aggregate distance of the rows of A to that subspace. A strong coreset samples and rescales a small number of rows to form SA while preserving, simultaneously for every subspace F of dimension at most k, the objective

$$\|SA(I - P_F)\|_{p,2}^p = (1 \pm \varepsilon)\|A(I - P_F)\|_{p,2}^p.$$

For $p > 2$, Woodruff and Yasuda obtained coreset size $\widetilde{O}_p(k^{p/2}\varepsilon^{-p})$ using a recursive ridge-leverage-score sampling framework (Woodruff & Yasuda, 2025). It was unclear whether the ε^{-p} dependence was intrinsic to the sampling rule or arose from the analysis of the recursive row-count recurrence.

The proof initially obtained by Colosseum shows that the same sampling rule supports a coreset of size $\widetilde{O}_p(k^{p/2}\varepsilon^{-2})$, while retaining $\widetilde{O}_p(\mathrm{nnz}(A)+d^{\omega})$ running time. The key step keeps the truncation in the sampling probabilities when bounding the surviving rows, which changes the fixed point of the recurrence from an ε^{-p} to an ε^{-2} dependence. The complete construction and analysis are given in (Lin et al., 2026b).

D.2 The Condition-Number Barrier in Sparse Least Squares

Sparse convex optimization seeks a vector with few nonzero coordinates whose objective value is close to that of the best k-sparse solution. For objectives with restricted condition number κ, known algorithms require output sparsity with essentially linear dependence on κ. Axiotis and Sviridenko gave an algorithm achieving this dependence and conjectured that it could not be improved by a polynomial-time algorithm (Axiotis & Sviridenko, 2021).

For least-squares objectives, the new result establishes this barrier conditional on the randomized exact-volume Small-Set Expansion Hypothesis. Under this hypothesis, for every fixed $\gamma \in (0, 1]$, no randomized polynomial-time algorithm can, with probability at least $2/3$, return a vector x such that, writing $s = \|x\|_0$,

$$\|Ax - b\|_2^2 \le \min_{\|z\|_0 \le k} \|Az - b\|_2^2 + \varepsilon \quad \text{and} \quad s = O\left(k\kappa_{s+k}^{1-\gamma}\right),$$

where κ_r is the restricted condition number at sparsity level r. Thus, no fixed sublinear power of the condition number can replace the linear dependence. The complete reduction and parameter regime are given in (Lin et al., 2026a).

D.3 Dimension Lower Bounds for Maximum Inner Product Embeddings

Multi-vector embeddings represent an item by a point cloud and compare point clouds using Chamfer similarity, whereas single-vector embeddings compare one vector per item by an inner product. For singleton queries, Chamfer similarity reduces to maximum inner product similarity. An upper bound of $m^{O(1/\varepsilon^2)}$ was known for representing point clouds of size at most m by single vectors, while the earlier lower bound $(\varepsilon^2 m)^{\Omega(1/\varepsilon)}$ left a gap between $1/\varepsilon$ and $1/\varepsilon^2$ in the exponent of m (Jayaram, 2026).

The new lower bound nearly closes this gap. For every fixed $\delta \in (0, 1)$, there are constants $A_\delta, c_\delta > 0$ such that, for sufficiently small $\varepsilon > 0$ and $m \ge (1/\varepsilon)^{A_\delta}$, there are unit query vectors and document point clouds for which any single-vector representation approximating all maximum inner products to additive error ε must have dimension

$$D \ge m^{c_\delta/\varepsilon^{2-2\delta}}.$$

The lower bound holds even for fully data-dependent representations and extends to Chamfer similarity. The construction and approximate-rank argument are given in (Jayaram et al., 2026a).

D.4 Single-Stage Hadamard Quantization

Randomized Hadamard transforms provide fast preprocessing for quantizing high-dimensional vectors in similarity search, distributed learning, and model compression. Earlier work gave an unbiased dithered quantizer with sharp mean-squared error guarantees (Feng et al., 2026c). Its finer $1/d$-scale inner-product estimator, however, used a second randomized transform and a residual quantization stage, adding both communication and a larger leading constant.

The new estimator shows that the second stage is unnecessary for attaining the same $1/(d4^b)$ mean-squared-error scaling. Pairwise-independent dithers across Hadamard coordinates yield an unbiased, single-stage estimator using $b \geq 2$ bits per coordinate and satisfying

$$\mathbb{E}\Big[|\langle y, \widehat{x} - x\rangle|^2\Big] \leq \left(\frac{3\pi\sqrt{3}}{2} + o(1)\right) \frac{\|y\|_2^2}{d\,4^b}.$$

Here x is a fixed unit vector, y is a fixed query, d admits a normalized Hadamard matrix, and the $o(1)$ term tends to zero as $b \to \infty$. It removes the residual-stage $O(d)$-bit payload and reduces the leading constant in the proved upper bound by a factor of approximately 5.93. The estimator and proof are given in (Lin et al., 2026c).

D.5 Lower Bounds for Prefix-Matrix Factorizations

Let Q be the $n \times n$ lower-triangular all-ones matrix, which maps a vector to its sequence of prefix sums. For a factorization $Q = AB$, the quantity

$$\gamma_{2,1}(Q) = \inf_{Q=AB} \|A\|_{2\to\infty} \|B\|_{1\to 1}$$

governs space bounds for factorization-based rank and quantile estimation in turnstile streams, as well as error bounds for matrix mechanisms in continual counting. The matrix-factorization framework in (Bulánek et al., 2026) made a sharp characterization of this cost a central question.

Lin et al. (2026d) prove the lower bound

$$\gamma_{2,1}(Q) = \Omega\left(\frac{\log^{3/2} n}{(\log\log n)^{3/2}}\right)$$

over real factorizations of arbitrary finite inner dimension. A dyadic factorization gives the upper bound $O(\log^{3/2} n)$, so the two bounds differ by only a factor of $(\log\log n)^{3/2}$. The proof combines right-sided Haar projections with a scale-dependent numerical-sparsity decomposition, then aggregates the resulting estimates across dyadic scales. Concurrent work improves the dependence on $\log\log n$; see Lin et al. (2026d, Section 1.2) for the comparison.

The following two case studies complement the results above by illustrating the workflow on unusually long proof artifacts and under information isolation.

D.6 Case Study: Long-Form Proof Construction for Knuth's Cycles

For an integer $m > 2$, Knuth's cycles problem asks whether the directed edges of the Cayley graph

$$\Gamma_m = \mathrm{Cay}\big(\mathbb{Z}_m^3, \{e_1, e_2, e_3\}\big)$$

can be partitioned into three directed Hamiltonian cycles. The odd case admits a simple construction with a complete proof, while the even case remained more difficult (Knuth, 2026). The updated notes also record an earlier, more intricate even-case construction generated by GPT-5.3-Codex, for which a complete proof was subsequently obtained with GPT-5.4 Pro (GPT-5.4 Pro, 2026). A later multi-agent search produced a much simpler construction for even m and verified it computationally through $m \leq 2000$ (Aquino-Michaels, 2026), but a rigorous symbolic proof for arbitrary even m remained open. A subsequent note introduced a second simple construction and reports full-length AI-generated proof drafts for both (Brenner et al., 2026).

Although each construction is specified by a compact set of local routing rules, proving that it yields three Hamiltonian cycles requires a substantially more involved global argument. The rules depend

on parity and include several exceptional boundary cases, whose interactions must be controlled uniformly for arbitrary even m. In particular, the proof must show that every directed edge is assigned exactly once and that each color class traverses all m^3 vertices in a single cycle rather than decomposing into shorter cycles. The workflow then produced a 46-page proof draft for the earlier construction and a 75-page proof draft for the new one (Brenner et al., 2026). This case study shows how a proof far beyond the length of a typical single model response can instead be developed as a persistent, revisable document organized around an explicit dependency structure.

D.7 Case Study: Independent Rediscovery of the Erdős Unit-Distance Breakthrough

Let $u(n)$ be the maximum number of unit-distance pairs determined by n points in the plane. Erdős's classical grid construction gives $n^{1+\Omega(1/\log\log n)}$ unit-distance pairs, and he conjectured that $u(n) = n^{1+o(1)}$. OpenAI reports that an internal model generated a counterexample to this long-standing conjecture in 2026. A group of mathematicians then distilled, simplified, and human-verified the AI-generated argument, establishing that infinitely many point sets determine at least $n^{1+\delta}$ unit-distance pairs for some fixed $\delta > 0$ (Alon et al., 2026).

We then ran Colosseum on the same problem using Gemini 3.1 Pro as the base model, with internet access disabled and only the original problem statement as the initial prompt. The model's documented knowledge cutoff is January 2025 (Google, 2026), before the reported counterexample. The resulting 22-page draft (Colosseum Research Harness, 2026) developed a number-theoretic approach based on unramified towers and relative unit groups, independently arriving at the central architecture of the OpenAI solution (Alon et al., 2026) and pursuing a bound of the form $u(n) \geq n^{1+\varepsilon_0}$ for infinitely many n, with $\varepsilon_0 > 0$ given explicitly. Notably, the run proceeded through 15 exploration rounds. Across these rounds, the accumulated-knowledge layer carried partial results, objections, and failed attempts forward across successive rounds, allowing the strategy-selection loop to synthesize the accumulated evidence into subsequent research directions. This provides a concrete example of the architecture sustaining a coherent long-horizon research process.

E Competitive-Programming Evaluation Details

To test whether the same architecture transfers from theorem proving to executable algorithmic tasks, we follow the Codeforces evaluation described for Gemini 3 Deep Think: all 222 problems with clist.by difficulty estimates above 1500 from contests held between April and October 2025 (Google DeepMind, 2026). In our evaluation snapshot, these problems come from 52 Codeforces contests numbered 2084–2162. For the rating calculation, we use the numerical difficulty estimates stored in the evaluation corpus. These values are on a Codeforces-like scale and are distinct from the problem ratings displayed by Codeforces. They have a median of 2381 and range from 1530 to 4599; Table 5 gives the full distribution. Each submitted C++ solution is compiled and run against the complete hidden test set using the problem's original checker, and is accepted only if every test passes. Strict as-submitted grading yields the same score, so no accepted solution depends on an output repair.

Table 5: Distribution of the evaluation-corpus difficulty estimates for the 222 Codeforces problems.

Estimated rating band	< 2000	$[2000, 2400)$	$[2400, 2800)$	$[2800, 3200)$	$[3200, 3400)$	≥ 3400
Problems	59	53	44	32	8	26

Previous evaluations map model performance onto the Codeforces scale using either contest-calibrated rating systems or probabilistic models over rated problem sets (Quan et al., 2025; Zheng et al., 2025; Zhou et al., 2026). Because the resulting number depends on both the corpus and the calibration rule, we define ours explicitly and do not treat it as an official contestant rating. A problem with estimated difficulty r is treated as an opponent that a contestant of strength x solves with probability

$$P(\text{solve} \mid r, x) = \frac{1}{1 + 10^{(r-x)/400}}.$$

For difficulty estimates $r_1, \ldots, r_{222}$, the corpus-level performance rating $\hat{x}$ is the unique solution to

$$\sum_{i=1}^{222} \frac{1}{1 + 10^{(r_i - \hat{x})/400}} = n_{\text{solved}}.$$

Competitive-programming systems illustrate two ways of scaling inference. AlphaCode and AlphaCode 2 generate large candidate populations and reduce them through compilation, sample filtering, and behavior-based selection (Li et al., 2022; AlphaCode Team, 2023). Later systems make execution iterative: tool-assisted reasoning, differential or generated tests, and repair turn runtime evidence into feedback for the next attempt (El-Kishky et al., 2025; Li et al., 2026b; Wang et al., 2026; Li et al., 2026a). The case study below asks a different systems question: whether a proof-oriented exploration–decomposition workflow transfers to executable programs without being replaced by a code-specific controller.

Architecture Transfer. When applied to competitive programming, COLOSSEUM retains the proof-oriented architecture used for mathematical research. Exploration searches over solution strategies and passes the selected route to the decomposer. Rather than partitioning a program into software components, the decomposer represents that route as a directed acyclic graph of mathematical and algorithmic subproblems. Each subproblem is solved and aggregated as described in Section 3. The decomposition differs only at its endpoint: a terminal implementation subproblem realizes the algorithm established by the preceding nodes as a single C++ program. Table 6 gives an example decomposition for Codeforces 2084F.

Table 6: Dependency graph produced for Codeforces 2084F.

Node	Subproblem	Depends on
1	Equivalence of Reachability to Inversion Subset	—
2	DAG Bounds Propagation using Fenwick Trees	1
3	EDF Scheduling of Missing Elements	2
4	Global Verification via Fenwick Trees	3
5	**C++ Implementation**	1, 2, 3, 4

Execution Feedback. The one addition to this reasoning pipeline is a C++ execution probe. Once a candidate implementation is available, the probe compiles and runs it on public samples and model-generated stress inputs, returning checker outcomes together with time and memory measurements to the existing verification–revision loop. Hidden tests remain inaccessible to the workflow and are used only for final grading. The probe makes implementation-stage failures available to the existing verification–revision loop.

Results. With Gemini 3.1 Pro as the base model, the execution-enabled configuration records 218 accepted solutions on the 222-problem suite and obtains a corpus-level performance rating of 4263. The comparison configuration without the execution probe records 213 accepted solutions and obtains a rating of 3918. These results show that the same proof-oriented architecture used for mathematical research can also achieve a high acceptance rate on competitive-programming problems.

Table 7: Comparison of the baseline and execution-enabled configurations on the 222-problem Codeforces evaluation.

Configuration	Accepted	Performance rating
Without execution probe	213	3918
With execution probe	218	4263

F EXTENDED RELATED WORK

We review work on inference-time search and critique, discovery guided by verifiers and evaluators, and systems that support sustained research.

F.1 Search, Critique, and Aggregation

Sampling diverse reasoning paths and aggregating their answers (Wang et al., 2023), searching over intermediate states (Yao et al., 2023), revising against model-generated feedback (Madaan et al., 2023), and organizing multi-agent debate (Du et al., 2024) provide complementary ways to use additional inference. Wu et al. (2025) compare inference strategies under fixed compute budgets and introduce REBASE, which allocates tree expansions using intermediate rewards. Other work focuses on falsification. REFUTE evaluates whether language models can construct counterexamples to incorrect competitive-programming submissions (Sinha et al., 2025). Momus extracts key conjectures from stalled proof attempts and asks fresh solver instances to try to prove each conjecture and its negation. This context-detachment mechanism aims to escape solver–grader "cognitive wells" (Dang et al., 2026). Colosseum uses search, critique, and aggregation throughout the research workflow. Within each stage, proposals remain paired with their falsification reports during aggregation. The resulting artifact, unresolved objections, and repair targets remain available during strategy selection, proof construction, and verification.

F.2 Discovery Guided by Verifiers and Evaluators

Machine-checkable feedback provides a concrete test for proposals during search. Learned proposals can guide symbolic deduction (Trinh et al., 2024), while formal proof checking can guide theorem proving (Hubert et al., 2026); executable objectives instead guide the search for mathematical constructions (Romera-Paredes et al., 2024; Novikov et al., 2025; Georgiev et al., 2025). Recent systems extend this principle to research-scale formalization through agentic Lean search (Tsoukalas et al., 2026) and retrieval-assisted proof development (Ju et al., 2026). LeanMarathon uses an evolving blueprint and a proof dependency graph to coordinate parallel development and local repair, with changes checked through continuous integration (Zhang et al., 2026).

These systems ultimately require a proof that passes formal checking against a precise Lean statement (Tsoukalas et al., 2026; Ju et al., 2026; Zhang et al., 2026). Colosseum reviews provisional strategies, intermediate claims, and natural-language proof drafts. Formal checks and executable tests, when available, contribute evidence alongside model-generated critiques. The workflow uses this combined evidence to revise its research state and assess the assembled argument.

F.3 Autonomous and Collective Research Workflows

Research workflows must decide how to reuse intermediate work and when to ask for further verification. Aletheia maintains a generate–verify–revise loop over long natural-language solutions (Feng et al., 2026b). Other systems assign these tasks to different roles: QED separates decomposition, proof generation, and verification (An et al., 2026); ProofCouncil iterates between an author and a critic (Schmitt et al., 2026); and RMA coordinates initializer, proposer, and verifier agents through shared structured memory (Zhao et al., 2026). These systems use verification repeatedly as an argument develops.

These systems also differ in how they store and share intermediate work. Danus admits verifier-approved claims into a fact graph together with their proofs and logical dependencies (Liu et al., 2026). Other systems combine persistent proof state, literature retrieval, computational or formal tools, and human steering (Gong et al., 2026; Cao et al., 2026; Zheng et al., 2026). Their shared state may be organized around verified claims, proof obligations, working files, or an interactive workspace.

Some frameworks support ongoing research communities. Within Station, heterogeneous agents choose their own directions and publish an internal literature that later agent generations can extend (Chung et al., 2026); in EinsteinArena, user-supplied agents interact asynchronously through shared submissions, verifiers, and discussions (Bianchi et al., 2026).

Related work also studies collaboration and review across the scientific research process. Gemini-assisted case studies identify decomposition and iterative refinement as common techniques and also document models serving as adversarial reviewers (Woodruff et al., 2026). Co-Scientist applies generate–critique–evolve loops to research hypotheses (Gottweis et al., 2026), while the AI Scientist systems extend such orchestration through experimentation, analysis, and manuscript production

(Lu et al., 2024; Yamada et al., 2025). PAT applies inference-scaled agents downstream to scientific review and verification (Jayaram et al., 2026b).

Aletheia and OpenAI have reported results on FirstProof (Feng et al., 2026a; OpenAI, 2026a). OpenAI also reports ten advances in mathematics and theoretical computer science, each accompanied by a Lean certificate (OpenAI, 2026b). Other reports describe Claude-assisted results concerning the Riemann zeta function (Anthropic, 2026b), cryptanalysis (Anthropic, 2026a), and the Jacobian conjecture (Alpöge, 2026; Gao, 2026). Differences in human involvement, disclosure, and evaluation make it difficult to isolate the contribution of any one workflow component to these outcomes.

COLOSSEUM combines these ideas to connect strategy exploration with the development and repair of a long proof. Its readiness gate controls when a route is decomposed into subproblems, and its aggregation procedure carries critiques alongside the proposals they address. Shared drafts and research knowledge keep that evidence available during both local revision and renewed exploration.

G LIMITATIONS AND FUTURE WORK

G.1 LIMITATIONS

The evaluations assess COLOSSEUM as a complete workflow, but the methods are not compared under the same inference budget. Total compute depends on the candidate populations, the number of subproblems, and the local retries and revision rounds used in each run. Measuring this cost and comparing methods at a fixed budget would help determine how efficiently the workflow uses additional inference. Ablation studies under the same budget are also needed to separate the effects of the readiness gate, overlapping aggregation, decomposition, and research memory. These comparisons would clarify which parts of the workflow contribute most to performance and where additional compute is most useful.

G.2 FUTURE WORK

We describe two classes of possible extensions to COLOSSEUM that operate at different timescales. Within a run, the workflow could adapt the organization of exploration, the local dependency structure, and the allocation of inference compute to the evolving state of the investigation. Across runs, validated research trajectories could be used as post-training data to improve the underlying model.

G.2.1 ADAPTING THE INFERENCE-TIME WORKFLOW

Clustered Exploration of Distinct Research Directions. Exploration currently aggregates a mixed population of candidate strategies. When many candidates develop variants of the same idea, a less common but genuinely different direction may disappear before it has been explored in sufficient depth. An extension is to cluster strategies by their central mechanism, reduction, or representation, and to run a separate exploration–falsification–aggregation process within each cluster. The resulting cluster-level strategies can then be compared and integrated at a later stage.

Preliminary, non-systematic runs suggest that this organization can preserve productive minority directions and produce viable inputs to downstream proof construction. More systematic work is needed to determine how clusters should be formed and updated, how inference should be allocated among them, and when distinct directions should be merged.

Local Restructuring after Subproblem Failure. The current local retry keeps the assigned subproblem and its dependencies fixed while regenerating the corresponding section. Repeated failure may instead indicate that the subproblem boundary is poorly chosen: the task may combine several distinct claims, depend on an intermediate result that was not isolated, or require a different interface with nearby sections.

A local restructuring step could revise a small neighborhood of the dependency graph before retrying. It might split the failed subproblem, change the tasks of adjacent sections, introduce an intermediate section, or redirect local dependency edges. The revised neighborhood would retain its interface with the rest of the proof and remain acyclic, so completed sections outside that neighborhood would not

need to be regenerated. This would provide an intermediate response between retrying one fixed task and revising the proof outline as a whole.

Adaptive Inference Allocation. The current configurations fix population widths, aggregation fan-in, and sample counts before a run begins. The value of an additional sample or round, however, can vary across stages and subproblems. A future controller could use signals already produced by the workflow, such as strategy diversity, unresolved objections, repeated local failures, and agreement among independently generated candidates, to expand uncertain branches, grant additional retries to unstable sections, and stop stages whose outputs have stabilized. Compute-matched evaluation would be needed to distinguish improved allocation from simply using more inference.

G.2.2 Learning from Research Trajectories

The harness records structured research trajectories rather than only final answers: candidate strategies, falsifier critiques, aggregation decisions, dependency graphs, intermediate drafts, and revision histories. A natural direction is to use trajectories from runs with externally validated outcomes as post-training data for the base model. Intermediate states could provide supervision for strategy selection, decomposition, objection handling, and revision, while rejected routes and verifier feedback could supply negative and corrective signals. This could distill some of the benefits of inference-time orchestration into the underlying model and improve the starting point for later runs. The main challenge is credit assignment: a successful final result does not by itself reveal which intermediate strategies, critiques, or revisions were responsible for progress. The branching structure of the harness may help identify useful training signals by comparing candidates that share the same context but lead to different downstream outcomes.