ZERO: PROGRAMMING LONG-HORIZON MATHEMATICAL RESEARCH ACROSS AGENTS AND TOOLS

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce ZERO, a new framework for programming long-horizon mathematical research. ZERO represents research procedures as executable workflows. A meta-agent constructs and revises these workflows, while a runtime executes them across agents and mathematical tools. Researchers can start from a problem, provide procedural knowledge, or augment an existing research system. ZERO lifts an existing system's workflow into its executable representation, where the different stages can be modified, reassigned to different resources, or extended with new capabilities. Across four studies, ZERO (i) proves optimality of a previously open discrete-packing instance and finds novel constructions for more than 15 related variants; (ii) improves the state of the art for the five-color almost-coloring problem of the plane, lowering the removed fraction from the previous state-of-the-art value of 3.74% to below 3.5%, with the construction validated by exact-arithmetic geometric checks; (iii) resolves two open questions in Lagrangian relaxation; and (iv) extends existing research harnesses with formal verification and open-solver backends. Together, these results suggest a shift from programming individual mathematical agents to programming the mathematical research process itself.

1 INTRODUCTION

AI capabilities for mathematics are rapidly increasing, with AI systems now contributing to research-level mathematical results (OpenAI, 2026a;b). Some of these results arise through general-purpose chat interfaces or coding agents (Bubeck et al., 2025; Feng et al., 2026b; Zimmer et al., 2026), while others rely on increasingly specialized systems. Evolutionary agents such as AlphaEvolve search for mathematical constructions encoded as programs (Novikov et al., 2025). Mathematical reasoning systems such as Aletheia develop proofs and explanations (Feng et al., 2026c). Formal-proving systems produce machine-checked proofs (Achim et al., 2025; Wang et al., 2026a).

Together, these systems make increasingly powerful capabilities available to mathematicians, but organizing them into an effective research process remains largely manual. Existing systems typically instantiate a particular procedure, for example an evolutionary search loop (Novikov et al., 2025; Lange et al., 2025), a team of planner, prover, and critic agents (An et al., 2026; Zhao et al., 2026; Liu et al., 2026b; Schmitt et al., 2026; He et al., 2026; Cao et al., 2026), or a domain-specific research harness (Zhang et al., 2026a; Kim et al., 2026). By contrast, frontier mathematical research often requires systems to be combined or modified for the problem at hand, rather than used unchanged. Georgiev et al. (2025), for example, manually connected AlphaEvolve, Gemini Deep Think, and AlphaProof so that a discovered construction was proved correct and then formally verified. However, such procedures are difficult to optimize through repeated scored execution, as in workflow and harness search (Zhang et al., 2025a; Lee et al., 2026), since a single attempt can take hours or days and the decisive signal often arrives only once a construction is checked or a proof verified.

We introduce ZERO, a framework for constructing such procedures as explicit, executable programs for the problem at hand. A workflow specifies how agents, deterministic computation, researcher decisions, and complete external research systems interact. A meta-agent constructs and modifies workflows, while a separate runtime executes them and maintains their state, including long-running interactive agent sessions and points where the researcher decides. The meta-agent can draw on prior research cases, reusable workflows, and external systems, but the resulting workflow is explicit and can be inspected and edited. This separates the research procedure from the particular systems used

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.

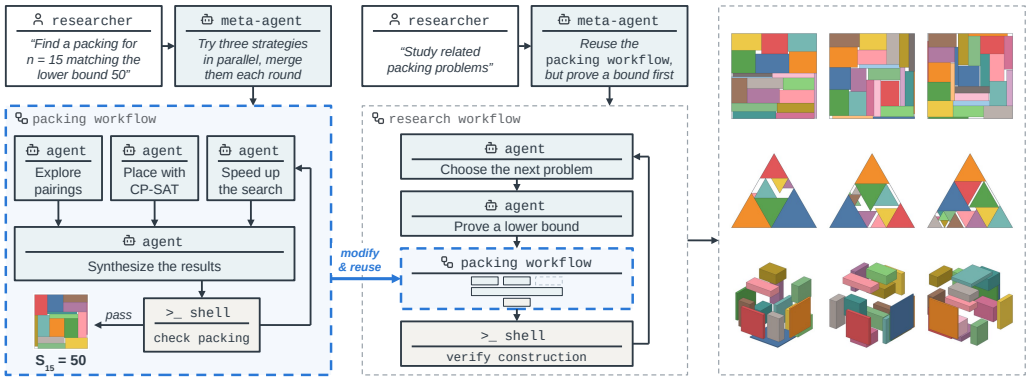


Figure 1: A workflow that solves one problem becomes a component for solving many. (i) The researcher states only the goal, a packing of 15 rectangles with side lengths $1, \ldots, 30$ into a 50×50 square (matching the area lower bound), and provides existing search code. The meta-agent designs the `packing workflow` itself, which finds such a packing and thereby solves a case open for over twenty years. (ii) The researcher then outlines a research loop (choose a problem, prove a lower bound, construct a packing) but does not specify which problems to study. The meta-agent implements this loop as a `research workflow`, using a modified `packing workflow` (blue) as its construction step. (iii) Packings found by the `research workflow` for rectangles, triangles, and cuboids, many of which match their lower bound and are therefore optimal.

to carry it out. The programmable object in ZERO is therefore the procedure itself, rather than a fixed mathematical agent, model, or harness. A workflow can be constructed from a problem, instantiated from researcher guidance, or derived by exposing and modifying the stages of an existing research system.

Once represented as a workflow, a procedure can be revised at test time, reused on related problems, or modified locally. The representation also makes explicit the evidence produced at each stage, including numerical constructions, informal arguments, and formal proofs.

We evaluate ZERO in four studies. In discrete rectangle packing, ZERO constructs a search procedure that finds a 50×50 packing for $n = 15$, matching the area lower bound and settling a case open for more than twenty years (OEIS Foundation Inc., 2026); the procedure is then reused in a research workflow over related rectangle, cuboid, triangle, and tetrahedron packings. For Hadwiger–Nelson variants, ZERO lowers the removed area of the best known planar almost-five-coloring from 3.74% to 3.49%, adapts the procedure to lower the removed volume of an almost-14-coloring of $\mathbb{R}^3$ from 3.4622% to 1.9851%, and extends the known continuum of six-colorings from $[0.354, 0.657]$ to $[0.3064832, 0.6818709]$ (Mundinger et al., 2024; 2025). From a researcher-specified procedure, ZERO constructs counterexamples answering two open questions of Dey et al. (2025) on extensions of Geoffrion's theorem, with the arguments formalized in Lean. Finally, ZERO augments two existing research harnesses, adding formal verification to VALG (Zhang et al., 2026a) and replacing AutoOPT's commercial solvers with an open stack (Kim et al., 2026).

Our contributions are as follows.

1. **We introduce ZERO**, a framework for programming long-horizon mathematical research as executable workflows. At test time, a meta-agent builds and revises workflows for each research problem, while a stateful runtime executes them across interactive agents, mathematical tools, researcher decisions, and entire external research systems.

2. **We validate ZERO in four research-level studies.** It settles a rectangle-packing case open for over twenty years, answers two open questions on extensions of Geoffrion's theorem, improves Hadwiger–Nelson constructions, and extends existing mathematical research tools.

2 THE ZERO SYSTEM

ZERO represents a mathematical research procedure as an executable workflow. A workflow can assign a task and a set of tools to a subagent, execute a command, or invoke a complete research system such as AlphaEvolve (Novikov et al., 2025). Its control flow determines when these operations run and how their results shape subsequent work. A harness-agnostic runtime executes the workflow and maintains its current state in a persistent workspace.

The researcher interacts with ZERO through a meta-agent, which is a general-purpose CLI coding agent configured with a ZERO-specific system prompt and a set of skills. The meta-agent can construct and modify workflows, submit them for execution, and monitor their progress. The starting point depends on how far the researcher has already developed the research procedure. In mode (i), *problem-led construction*, the project begins with a mathematical problem and relevant context but no prescribed procedure. The meta-agent decides how to organize the work and constructs the workflow. In mode (ii), *researcher-directed construction*, the researcher describes how some or all of the research should proceed. The meta-agent turns this guidance into a complete workflow and fills in the remaining execution details. In mode (iii), *system augmentation*, the researcher begins with an implemented mathematical research system. The meta-agent represents its procedure as a workflow so that individual parts can be modified, extended, or combined with other capabilities.

2.1 EXECUTABLE RESEARCH WORKFLOWS

ZERO represents a mathematical research workflow using three basic programming primitives. An `agent` node starts an interactive CLI-agent session and assigns it a task. Its configuration selects the CLI agent and provides native settings such as the model and available tools. The node further specifies the working directory. Agents primarily exchange work through the shared file system. A node may end with an explicit structured handoff, but this is optional: otherwise, the harness-agnostic runtime derives the node outcome from the agent's final state. While it runs, researchers and other agents can steer it by sending additional messages. A `shell` node executes a command, ranging from a small mathematical check to a complete external research system. A `pause` node waits for information or a decision from the researcher. Workflows arrange these nodes in sequences, parallel branches, and bounded loops, and pass results and files between them.

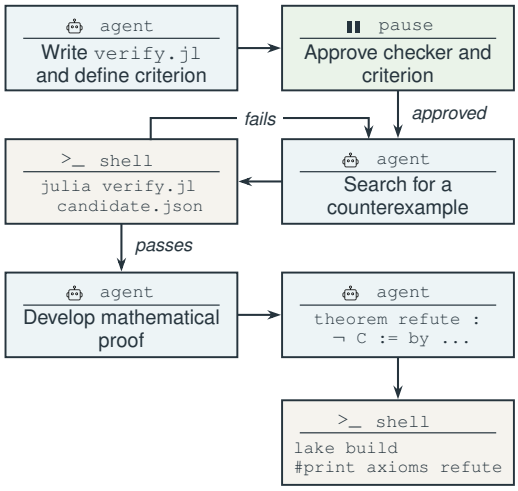


Figure 2: Example workflow from the Lagrangian-relaxation study. Candidate counterexamples are tested by a preapproved checker, then proved mathematically and verified in Lean.

A workflow records the stages of a research procedure and the checks connecting them. Figure 2 gives a simplified example based on the Lagrangian-relaxation study in Section 3.3. An agent first writes `verify.jl`, which numerically tests candidate counterexamples, and the researcher reviews the script and its acceptance criterion before search begins. Inside a loop, another agent proposes candidates and the same script checks each one. If the test passes, this provides computational evidence, but does not establish that the counterexample is correct. The workflow therefore sends passing candidates to further stages that develop a mathematical proof, formalize it in Lean, and finally build the development and inspect its axioms.

2.2 CONSTRUCTING AND RUNNING WORKFLOWS

The meta-agent is a CLI coding agent and the researcher's interface to ZERO. It is separate from the `agent` nodes introduced above. The meta-agent writes and edits workflows, submits them to the

runtime, and monitors their execution. Its system prompt specifies this role. When a skill is invoked, the meta-agent reads its instructions and gains access to any accompanying tools or templates.

ZERO provides one skill for each starting point. `/design-workflow` starts from a mathematical objective, its context, and the available artifacts, leaving the meta-agent to choose the research procedure. `/implement-workflow` starts from a partial or complete procedure and translates it into the primitives from Section 2.1. `/augment-system` starts from an existing research system and a requested change. The meta-agent maps the relevant stages to workflow nodes and modifies them there. The researcher can invoke a skill directly, or leave the choice to the meta-agent.

Before launch, the runtime checks the workflow and its inputs. This check covers node definitions, input schemas, launch profiles, conditions, and loop bounds without executing any nodes. Once it passes, the runtime records the execution plan and starts nodes as their dependencies complete. The researcher can answer `pause` nodes or message running agents. ZERO does not rewrite the workflow while it is executing. Adaptation occurs at two levels: agents may change how they pursue their assigned tasks within a run, while the meta-agent can use completed outputs and failures to revise the workflow for a subsequent run. Appendix F gives the complete JSON for a minimal bounded prover–verifier loop.

2.3 FINDING AND INCORPORATING RESEARCH SYSTEMS

ZERO uses `source-observatory` to find relevant case studies and research systems in papers and GitHub repositories. Our snapshot contains 735 mathematical case studies extracted from 230 papers, together with 207 candidate repositories from a separate GitHub search (Appendix A). For a new task, BM25 retrieves related case studies. Executable procedures are kept separately in the workflow library.

When a system appears useful, the meta-agent can add its code, wrap a pinned implementation, or connect to a hosted interface. It can treat the system as a single workflow step or expose its internal stages when they need to be modified or combined. Section 3.4.1 illustrates this representation process with OpenEvolve. Appendix F documents representations of the three FirstProof Batch 2 multi-agent submissions (Abouzaid et al., 2026b).

3 EXPERIMENTS AND CASE STUDIES

We evaluate ZERO across four studies that exercise different ways of programming mathematical research. Packing and coloring test workflow construction, reuse, and adaptation (Sections 3.1 and 3.2). Lagrangian relaxation starts from a researcher-specified procedure that ZERO turns into an executable workflow (Section 3.3). The final studies begin from existing research systems and test whether their procedures can be represented, modified, and extended as workflows (Section 3.4). Appendix B summarizes the starting points, system roles, human review, and validation for each study.

3.1 RECTANGLE PACKING

Packing problems are common testbeds for AI-assisted discovery (Novikov et al., 2025; Lange et al., 2025; Wang et al., 2025c). We study a discrete variant in which n rectangles with $2n$ distinct integer side lengths must be packed, axis-aligned and without overlap, into the smallest square of side S_n. A solution specifies both the *pairing* of side lengths and the positions. The following lower bound holds:

$$S_n \geq \left\lceil \sqrt{n(n+1)(2n+1)/3} \right\rceil,$$

For $n = 15$, this lower bound equals 50. The case $n = 15$ remained the only unresolved value among $n \leq 17$ for over 20 years (OEIS Foundation Inc., 2026). Since this lower bound is 50, any packing in a 50×50 square proves optimality.

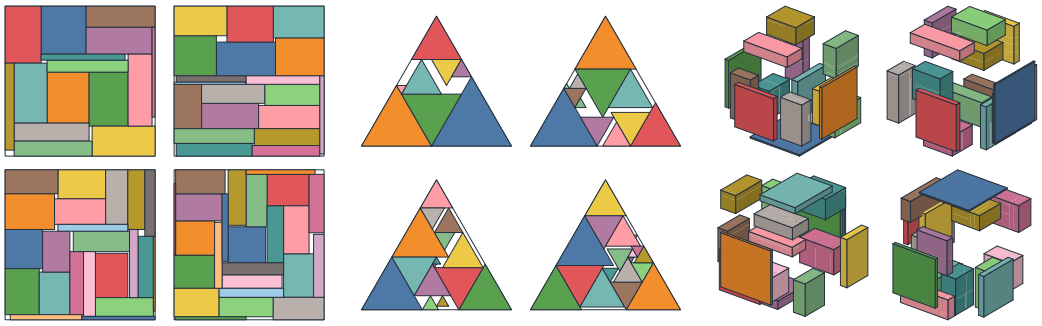


Figure 3: Selected packings produced in the discrete-packing study. From left to right, the panels show axis-aligned rectangles in a square, equilateral triangles in an equilateral triangle, and axis-aligned cuboids in a cube. The three-dimensional views are exploded to make the individual pieces visible.

3.1.1 RESOLVING $n = 15$

We used problem-led mode, invoking `/design-workflow` with the objective and an existing Java search code. We used GPT 5.5 for this experiment. ZERO built a multi-round workflow with a solver agent tuning direct formulations, an efficiency agent porting the search to Rust, a free exploration agent, and a synthesis agent combining their results after each round. The exploration agent enumerated promising pairings, and the solver agent passed them to CP-SAT as fixed input to the placement problem. This produced a packing with side 50, proving $S_{15} = 50$ (Appendix C). We chose baselines to test whether direct optimization, parallel sampling, iterative program improvement, or a larger autonomous-research budget was sufficient. CP-SAT and SCIP jointly optimized pairings and placements on 32 CPU cores, each with a 24-hour wall-clock budget. Best-of-16 ran 16 independent Pi coding-agent searches (Earendil Works, 2025), testing inference-time scaling without cross-run synthesis. ShinkaEvolve (Lange et al., 2025) tested evolutionary program improvement, while Agentic Researcher (Zimmer et al., 2026) tested long-horizon autonomous research at a substantially larger token budget. None found a packing with side 50 (Table 1).

Method	Tokens	Best S
CP-SAT	n/a	n/f
SCIP	n/a	n/f
Best-of-16 agents	22M	51
ShinkaEvolve	23M	51
Agentic Researcher	257M	51
ZERO	21M	**50**

Table 1: Search for $S_{15} = 50$. All methods ran for 24 hours. Token counts include cache reads and are rounded to the nearest million. The n/a entries denote solver-only runs. The n/f entries indicate that no packing was found.

3.1.2 FROM SOLVING A PROBLEM TO PROGRAMMING RESEARCH

In our second packing experiment, we tested whether the $n = 15$ methodology could seed a research program for discovering and solving related packing problems. We did not specify in advance which packing problems to study, that is, the number of objects, their dimension, or their shape, nor what lower bounds to seek. Using `/implement-workflow`, we asked ZERO to construct a workflow in which an outer agent selected the next problem based on the results so far, prioritizing experiments expected to be informative. For each problem, an inner agent proposed and attempted to prove a lower bound, which gave the construction agent a concrete target. The construction agent applied the $n = 15$ methodology or adapted it to the new geometry. Proved bounds and verified constructions then informed the outer agent's next choice.

For rectangles with $18 \leq n \leq 24$, the workflow found optimal packings in three cases and packings within 1 of the lower bound in the other four; Appendix C.4 gives every resulting construction. It then extended the program to three-dimensional cuboids, equilateral triangles, and tetrahedra. For the latter two, side lengths were fixed rather than paired, so the construction agent fixed the largest shape and solved the remaining placement with CP-SAT (Figure 3). To the best of our knowledge, these constructions have not previously appeared in the literature.

3.2 IMPROVING HADWIGER–NELSON CONSTRUCTIONS

The Hadwiger–Nelson problem asks for the minimum number of colors needed to color the Euclidean plane so that points at unit distance receive different colors. More than seventy years after it was posed, the answer remains known only to lie between five and seven. We study two families of related construction problems. See Appendix D for precise definitions and historical context. We used GPT 6 Astra for the Hadwiger–Nelson experiments.

3.2.1 ALMOST-COLORINGS

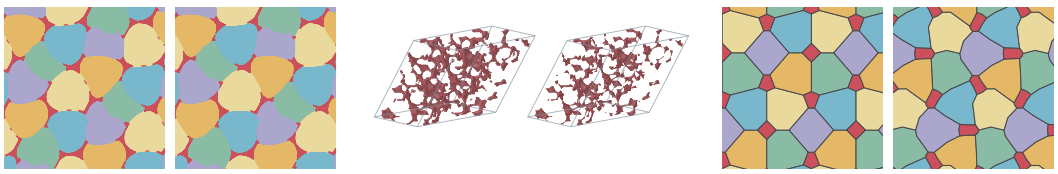
An *almost-k-coloring* leaves part of the space uncolored while requiring every colored class to avoid unit distance. The objective is to minimize the fraction of area or volume left uncolored. We consider an almost-five-coloring of the plane (Figure 4i) and an almost-14-coloring of $\mathbb{R}^3$ (Figure 4ii). Using a neural approach, Mundinger et al. (2025) reduced the removed fraction in the planar problem from 4.01% to 3.74% and constructed a three-dimensional almost-14-coloring with removed fraction 3.4622%.

We used `/design-workflow` with only the planar almost-five-coloring objective. ZERO designed a workflow in which an agent gathered the relevant context and files for GPT Pro, whose guidance informed three approaches pursued by independent agents. The successful approach combined refinement of a periodic coloring, local recoloring, and integer optimization to recover removed area, with separately implemented exact geometric checks validating the construction. It reduced the removed area from 3.74% to 3.49%. For comparison, both a coding agent working directly in the CLI and a domain expert using GPT 6 Astra inside Codex obtained 3.57%. We then supplied the successful procedure through `/implement-workflow` and adapted both the refinement method and the geometric checks to almost-14-coloring in three dimensions. The resulting construction reduced the removed volume from 3.4622% to 1.9851%, a relative reduction of 42.7%.

3.2.2 EXTENDING THE CONTINUUM OF SIX-COLORINGS

An *off-diagonal six-coloring* colors the whole plane with five colors that avoid unit distance and a sixth color that avoids a distance $d > 0$ (Figure 4iii). The objective is to extend the range of values of d for which such a coloring is known. Mundinger et al. (2024) established $0.354 \leq d \leq 0.657$, the first extension of the known range in thirty years, using two overlapping constructions. Their first construction varies with d and covers $[0.354, 0.553]$, while their second is a single fixed coloring valid throughout $[0.418, 0.657]$. Together, the two intervals certify the full continuum between their outer endpoints.

This experiment therefore required a family of constructions rather than a single improved coloring. We used `/implement-workflow` to construct an iterative workflow around the current set of certified intervals. An outer agent first decided whether the next attempt should lower the left endpoint, raise the right endpoint, or fill a gap between existing intervals. An inner construction agent then searched for an off-diagonal six-coloring addressing that target. A separate verifier agent attempted to prove that the proposed coloring avoids the required distances throughout its claimed interval,



(i) Planar almost-five (ii) 3D almost-14 (iii) Off-diagonal six

Figure 4: Three variants of the Hadwiger–Nelson problem. Each pair compares a published construction (Mundinger et al., 2024; 2025) (left panel) with our construction (right panel). (i) Almost-five-coloring of the plane, with a significant reduction in the removed area. (ii) Almost-14-coloring of $\mathbb{R}^3$, with a visibly smaller removed region. (iii) Off-diagonal six-coloring of the plane, where our search finds a more complex periodic pattern.

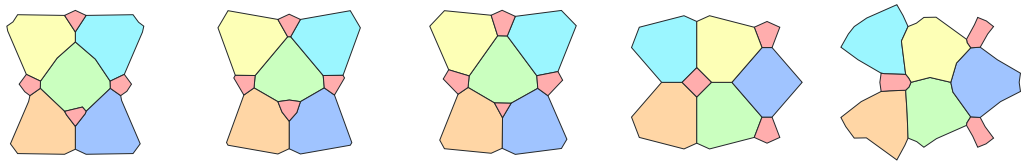


Figure 5: Five fixed periodic six-colorings whose certified intervals overlap to cover $[0.3064832, 0.6818709]$. Each panel is a complete repeat block for a different interval, and consecutive intervals overlap so that their union certifies the full range.

following the geometric verification strategy of Mundinger et al. (2024). Only a verified interval was added to the certified union, after which the outer agent selected the next target and the process repeated. The workflow produced five fixed periodic colorings whose overlapping intervals jointly certify $0.3064832 \leq d \leq 0.6818709$, extending both endpoints of the continuum established by Mundinger et al. (2024) (Figure 5).

3.3 Resolving open questions on Lagrangian relaxation

Lagrangian relaxation is a standard way to bound an integer program without solving it directly. Given an integer set $X \subseteq \mathbb{Z}^n$, objective $c^\top x$, and constraints $Ax \geq b$, two natural bounds are

$$v^L = \sup_{\lambda \geq 0} \inf_{x \in X} \left(c^\top x + \lambda^\top (b - Ax) \right), \qquad \bar{v}^\star = \inf \left\{ c^\top x \mid Ax \geq b,\ x \in \overline{\mathrm{conv}}(X) \right\}.$$

The first is the best bound obtained by moving the constraints into the objective, while the second replaces the discrete set X by its closed convex hull. Geoffrion's theorem identifies settings in which they coincide, that is, $v^L = \bar{v}^\star$.

Dey et al. (2025) extended this equality and asked whether it still holds when the closed convex hull is locally polyhedral but the finite optimum $\bar{v}^\star$ is not attained, and whether a result for one rational constraint extends to multiple rational constraints. A counterexample to either extension requires $v^L < \bar{v}^\star$.

We used these questions to evaluate researcher-directed workflow construction. We specified a procedure that searched for a counterexample using a numerical checker, proved and independently verified any accepted candidate, and then formalized the proof in Lean. If the search exhausted its budget, the procedure instead attempted to prove and formalize the proposed extension. ZERO converted these instructions into the executable workflow shown in Figure 2 and assigned its stages to agents and tools.

For both questions, the counterexample branch succeeded. For the nonattainment question, the workflow constructed $X \subseteq \mathbb{Z}^4$ with $v^L = -1 < 0 = \bar{v}^\star$, where the closed convex hull is locally polyhedral and the latter infimum is not attained. For the multiple-constraint question, it constructed an example with $v^L = 0 < 1 = \bar{v}^\star$ using the integer points of a closed, non-polyhedral convex set in $\mathbb{R}^4$. A subsequent run strengthened this result to $X = Q \cap \mathbb{Z}^5$, where Q is a polyhedron defined by seven linear inequalities. These counterexamples answer both open questions negatively.

The same workflow formalized the accepted arguments using GPT 5.6 Sol in Pi and Aristotle. It produced three Lean developments that compiled without admissions and passed the axiom and statement-preservation checks. The complete constructions and proofs appear in Appendix E.

Finally, the workflow generated a standalone paper modeled on Dey et al. (2025). The results were correct, but the exposition required human revision and did not consistently reach publication quality.

3.4 Representing and extending existing research systems

We first show how ZERO represents the internal procedures of an existing research system in editable form. We then test whether such representations can be extended with new capabilities.

3.4.1 Representing and modifying evolutionary search

We use OpenEvolve to illustrate how ZERO can represent an existing research system in a form that remains editable. Calling the complete implementation through a single opaque shell node does not meet this criterion. Our representation uses shell nodes for selection, evaluation, and archive updates, while an agent node performs LLM-based mutation (Figure 6). This decomposition makes modifications local. The lower workflow replaces the selection shell node with an agent that chooses which archived program to modify, while retaining the same mutation, evaluation, and archive components. The exposed components can also be extended with task-specific validation or composed with other workflows. Appendix F describes additional representations of prover–verifier loops and the ProofCouncil, Moonshot, and Momus systems from FirstProof.

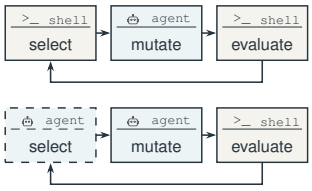


Figure 6: OpenEvolve (top) and an extension that replaces selection with an agent choosing the program to modify (bottom).

We use `/augment-system` to test whether two skill-based systems, VALG and AutoOPT, can be translated into workflows and extended with a missing capability. The resulting workflows separate agentic reasoning from deterministic computation and validation.

3.4.2 VALG: ADDING FORMAL VERIFICATION

VALG guides machine-learning theory research from problem formulation to candidate proofs (Zhang et al., 2026a). Its 22 reported theorem candidates received LLM review and a limited human audit, leaving formal verification as an explicit gap. We added a ZERO workflow in which agents planned and implemented the proof, while Lean compilation and statement checks provided deterministic validation. The theorem estimates an unknown mean from one bit per independent observation, with every query fixed before the observations are seen (Lau & Scarlett, 2026). For its measure-theoretic and concentration arguments, ZERO retrieved AI4SLT, a Lean library for formal statistical learning theory, and pinned a compatible Lean 4.32 environment (Zhang et al., 2026d).

The first complete development compiled, but source review identified a circular construction of an auxiliary probability law and four mismatches with the source statement. Repair and integration stages corrected these defects and reran the checks. The final 36-module, 7,853-line development formalizes the protocol, sample bound, and probability guarantee in all three moment regimes, compiles without admissions, and matches the source under human and GPT Pro review. Appendix G gives the formal target, repairs, verification records, and cost breakdown.

3.4.3 AUTOOPT: REPLACING COMMERCIAL SOLVERS

AutoOPT combines numerical search for first-order methods with LLM-assisted derivation and Lean formalization (Kim et al., 2026). Its numerical stages depended on MOSEK and Gurobi, which prevented the harness from running with an entirely open solver stack. We translated the fixed-method, local-design, and global-search stages into deterministic workflow nodes and replaced their backends with Clarabel, Ipopt, and SCIP.

Clarabel reproduced the fixed-method programs, and Ipopt recovered the local designs. A direct SCIP translation found candidate methods but produced weak lower bounds because it did not enforce the positive-semidefinite slack effectively. We therefore ported the two-pass BnB-PEP procedure and added a SCIP constraint handler that separates violated positive-semidefinite constraints. The resulting workflow recovered the published ITEM-f and lemniscate results. Its global runs reached the reported 1% tolerance for $N = 1, \ldots, 5$, with the source's support restrictions retained for $N = 3, 4, 5$. Independent primal and dual PEPs checked the recovered methods. Appendix H gives the numerical comparisons and implementation details.

4 RELATED WORK

Systems for mathematical research. Recent systems support mathematical research through long-horizon reasoning, persistent workspaces, and coordinated agents. Aletheia develops and revises natural-language proofs while using literature and computational tools (Feng et al., 2026c;a). AI Co-Mathematician provides an interactive workspace in which researchers and agents retain conjectures, failed attempts, and intermediate artifacts (Zheng et al., 2026). Other systems organize proof search

among planners, proposers, critics, and verifiers (An et al., 2026; Zhao et al., 2026; Liu et al., 2026b; He et al., 2026; Cao et al., 2026), while Moonshine and MECA place greater emphasis on constructing conjectures (Chen & Jiang, 2026; Long et al., 2026). FirstProof and related studies examine lightweight or researcher-directed procedures for attacking unpublished problems (Abouzaid et al., 2026a; Meng et al., 2026; Corneli, 2026). More specialized harnesses encode procedures for computational mathematics, machine-learning theory, and optimization (Chen et al., 2026b; Zhang et al., 2026a; Kim et al., 2026; Kim & Pilanci, 2026; Gong et al., 2026; Snopov & Magai, 2026). These systems offer varied architectures for mathematical work; ZERO can use them as workflow resources or expose their stages for modification and composition.

Specialized resources and mathematical evidence. Program-search systems use executable objectives to discover mathematical constructions and algorithms (Romera-Paredes et al., 2024; Novikov et al., 2025; Lange et al., 2025; Assumpção et al., 2026; Khrulkov et al., 2025). Recent variants also adapt the hypotheses, strategies, or search procedure used during evolution (Chen et al., 2026d; Luo et al., 2026; Liu et al., 2026c; Wang et al., 2025c; Yuksekgonul et al., 2026). Formal-proving systems provide a different resource by producing proof objects checked by a trusted kernel (Achim et al., 2025; Chen et al., 2025; Wang et al., 2026a; Ju et al., 2026; Tsoukalas et al., 2026). Several studies connect informal reasoning, coding agents, specialized provers, and proof assistants within a single mathematical project (Feng et al., 2026b; Ilin, 2026; Lau, 2026; Sothanaphan, 2026). Other work develops supporting capabilities such as predicting proof progress and retrieving relevant library results (George et al., 2026; Gao et al., 2025). Georgiev et al. (2025) manually combine AlphaEvolve, Gemini Deep Think, and AlphaProof so that a programmatically discovered construction is explained and then formally verified. ZERO makes such compositions explicit, including their computational checks, formal-verification stages, and researcher decisions.

Programmable and adaptive agent workflows. Agent workflows are increasingly treated as programmable and optimizable objects. DSPy expresses model pipelines through modules and control flow, while AFlow searches over code-represented workflows (Khattab et al., 2023; Zhang et al., 2025a). XFlow, AgentSPEX, Artic, and FlowAgent make execution structure, state, dependencies, or procedural constraints explicit (Li et al., 2026; Wang et al., 2026b; Xu et al., 2026c; Shi et al., 2025). For mathematical research specifically, ProofCouncil provides an agent-building library for expressing systems as editable conditional DAGs with model calls, computation, deterministic gates, control flow, and human-in-the-loop nodes (Schmitt et al., 2026). Related work exposes execution traces to meta-agents or compiles textual skills into executable procedures (Yu et al., 2026; Zhang et al., 2026c; Liu et al., 2026a; Guo et al., 2026). Workflows can also be generated from task descriptions (Zhang et al., 2025b; Xu & Tai, 2026) or optimized using execution feedback (Wang et al., 2025a; Hu et al., 2026; Wang et al., 2025b; Xu et al., 2026b; Chen et al., 2026c; Lee et al., 2026; Zhang et al., 2026b). Other systems adapt prompts, skills, memory, or agent organization during deployment (Karten et al., 2026; Wong et al., 2026; Chen et al., 2026a; Xue & Yang, 2026). AFlow and Meta-Harness use repeated scored evaluations to search over workflows or harness code (Zhang et al., 2025a; Lee et al., 2026); Continual Harness instead adapts within an ongoing episode without resetting the environment (Karten et al., 2026). Reported gains from workflow optimization can depend strongly on the chosen representation and baseline (Nie et al., 2026; Cemri et al., 2025; Xu et al., 2026a; Gideoni et al., 2026). ZERO uses a meta-agent to construct and modify research workflows from a mathematical objective, a researcher-specified procedure, or an existing research system. Successful workflows can be reused and adapted across problems without repeated scored evaluations of complete research runs.

5 CONCLUSION AND LIMITATIONS

ZERO makes mathematical research workflows programmable across models, agents, provers, solvers, and existing systems. It helped settle a twenty-year-old packing instance, improve Hadwiger–Nelson constructions, resolve questions in Lagrangian relaxation, and augment research systems. However, limitations remain. Agentic systems for mathematical research are challenging to evaluate. Outcomes reflect workflows, models, tools, and researcher judgment, while consequential runs are costly to repeat. Claims remain bounded by their validation; explicit checks, provenance, and researcher oversight remain essential.

AI USE STATEMENT

Generative AI systems were central to the research workflows studied and were also used for literature analysis, code and workflow development, mathematical reasoning and formalization, and manuscript drafting and editing. The authors reviewed all AI-assisted work, validated the reported results using the procedures described in the paper, and take responsibility for the final content.

REPRODUCIBILITY STATEMENT

The main text describes the framework, research workflows, experimental protocols, and validation procedures; the appendices state the assumptions and provide proofs, certificates, and verifier details. The supplementary material contains the reported packing and coloring constructions together with compact checkers and evaluation instructions. We will release the complete source code upon acceptance.

REFERENCES

Mohammed Abouzaid, Andrew J. Blumberg, Martin Hairer, Joe Kileel, Tamara G. Kolda, Paul D. Nelson, Daniel Spielman, Nikhil Srivastava, Rachel Ward, Shmuel Weinberger, and Lauren Williams. First Proof, March 2026a. URL https://arxiv.org/abs/2602.05192.

Mohammed Abouzaid, Nikhil Srivastava, Rachel Ward, and Lauren Williams. First Proof Second Batch, June 2026b. URL https://arxiv.org/abs/2606.18119.

Tudor Achim, Alex Best, Alberto Bietti, Kevin Der, Mathïs Fédérico, Sergei Gukov, Daniel Halpern-Leistner, Kirsten Henningsgard, Yury Kudryashov, Alexander Meiburg, Martin Michelsen, Riley Patterson, Eric Rodriguez, Laura Scharff, Vikram Shanker, Vladmir Sicca, Hari Sowrirajan, Aidan Swope, Matyas Tamas, Vlad Tenev, Jonathan Thomm, Harold Williams, and Lawrence Wu. Aristotle: IMO-level Automated Theorem Proving, October 2025. URL https://arxiv.org/abs/2510.01346.

Chenyang An, Qihao Ye, Minghao Pan, and Jiayaun Zhang. QED: An Open-Source Multi-Agent System for Generating Mathematical Proofs on Open Problems, June 2026. URL https://arxiv.org/abs/2604.24021.

Henrique Assumpção, Diego Ferreira, Leandro Campos, and Fabricio Murai. CodeEvolve: An open source evolutionary coding agent for algorithmic discovery and optimization, March 2026. URL https://arxiv.org/abs/2510.14150.

Sébastien Bubeck, Christian Coester, Ronen Eldan, Timothy Gowers, Yin Tat Lee, Alexandru Lupsasca, Mehtaab Sawhney, Robert Scherrer, Mark Sellke, Brian K. Spears, Derya Unutmaz, Kevin Weil, Steven Yin, and Nikita Zhivotovskiy. Early science acceleration experiments with GPT-5, November 2025. URL https://arxiv.org/abs/2511.16072.

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, July 2026. URL https://arxiv.org/abs/2607.04394.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why Do Multi-Agent LLM Systems Fail?, October 2025. URL https://arxiv.org/abs/2503.13657.

Jiangjie Chen, Wenxiang Chen, Jiacheng Du, Jinyi Hu, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Wenlei Shi, Zhihong Wang, Mingxuan Wang, Chenrui Wei, Shufa Wei, Huajian Xin, Fan Yang, Weihao Gao, Zheng Yuan, Tianyang Zhan, Zeyu Zheng, Tianxi Zhou, and Thomas Hanwen Zhu. Seed-Prover 1.5: Mastering Undergraduate-Level Theorem Proving via Learning from Experience, December 2025. URL https://arxiv.org/abs/2512.17260.

Jianming Chen, Xuanbin Ye, Yawen Wang, Junjie Wang, Qing Wang, and Fanjiang XU. VCE-Skill: Enhancing Skill Self-Evolution with Version-Change Experience, August 2026a. URL https://arxiv.org/abs/2608.16544.

Leheng Chen, Zihao Liu, Wanyi He, and Bin Dong. Iteris: Agentic Research Loops for Computational Mathematics, June 2026b. URL https://arxiv.org/abs/2606.02484.

Tingyang Chen, Shuo Lu, Kang Zhao, Weicheng Meng, Hanlin Teng, Tianhao Li, Chao Li, Xule Liu, Jian Liang, Zhizhong Zhang, Yuan Xie, Heng Qu, Kun Shao, and Jian Luan. HarnessX: A Composable, Adaptive, and Evolvable Agent Harness Foundry, July 2026c. URL https://arxiv.org/abs/2606.14249.

Xiaoyang Chen and Xiang Jiang. Moonshine: An Autonomous Mathematical Research Agent Centered on Conjecture Generation, June 2026. URL https://arxiv.org/abs/2606.10806.

Yongqiang Chen, Chenxi Liu, Zhenhao Chen, Tongliang Liu, Bo Han, and Kun Zhang. CausalEvolve: Towards Open-Ended Discovery with Causal Scratchpad, March 2026d. URL https://arxiv.org/abs/2603.14575.

Joseph Corneli. A First Proof Sprint, February 2026. URL https://arxiv.org/abs/2602.13587.

Santanu S. Dey, Frédéric Meunier, and Diego Moran Ramirez. Geoffrion's theorem beyond finiteness and rationality, October 2025. URL https://arxiv.org/abs/2510.10966.

Earendil Works. Pi: AI agent toolkit, 2025.

Tony Feng, Junehyuk Jung, Sang-hyun Kim, Carlo Pagano, Sergei Gukov, Chiang-Chiang Tsai, David 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, March 2026a. URL https://arxiv.org/abs/2602.21201.

Tony Feng, Trieu Trinh, Garrett Bingham, Jiwon Kang, Shengtong Zhang, Sang-hyun Kim, Kevin Barreto, Carl Schildkraut, Junehyuk Jung, Jaehyeon Seo, Carlo Pagano, Yuri Chervonyi, Dawsen Hwang, Kaiying Hou, Sergei Gukov, Cheng-Chiang Tsai, Hyunwoo Choi, Youngbeom Jin, Wei-Yuan Li, Hao-An Wu, Ruey-An Shiu, Yu-Sheng Shih, Quoc V. Le, and Thang Luong. Semi-Autonomous Mathematics Discovery with Gemini: A Case Study on the Erdős Problems, February 2026b. URL https://arxiv.org/abs/2601.22401.

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, Kaiying Hou, Golnaz Ghiasi, Yi Tay, YaGuang Li, Chenkai Kuang, Yuan Liu, Hanzhao Lin, Evan Zheran Liu, Nigamaa Nayakanti, Xiaomeng Yang, Heng-Tze Cheng, Demis Hassabis, Koray Kavukcuoglu, Quoc V. Le, and Thang Luong. Towards Autonomous Mathematics Research, March 2026c. URL https://arxiv.org/abs/2602.10177.

Guoxiong Gao, Haocheng Ju, Jiedong Jiang, Zihan Qin, and Bin Dong. A Semantic Search Engine for Mathlib4, February 2025. URL https://arxiv.org/abs/2403.13310.

Robert Joseph George, Suozhi Huang, Peiyang Song, and Anima Anandkumar. LeanProgress: Guiding Search for Neural Theorem Proving via Proof Progress Prediction, January 2026. URL https://arxiv.org/abs/2502.17925.

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

Yonatan Gideoni, Sebastian Risi, and Yarin Gal. Simple Baselines are Competitive with Code Evolution, February 2026. URL https://arxiv.org/abs/2602.16805.

Ting Gong, Michael Ruofan Zeng, and Yong Yang. Albilich: Steerable Proof-State Orchestration for LLM-Based Mathematical Research with CAS Integration, July 2026. URL https://arxiv.org/abs/2607.27705.

Zhongxin Guo, Danrui Qi, Hanwen Gu, Peng Cheng, and Yongqiang Xiong. SKILL-DISCO: Distilling and Compiling Agent Traces into Reusable Procedural Skills, June 2026. URL https://arxiv.org/abs/2606.26669.

Yutong He, Daibo Li, Guohong Li, Jiahe Geng, Zhengyang Huang, Can Ren, Zekun Zhang, Yifan Liu, Shuchen Zhu, Hengrui Zhang, Boao Kong, Ming Sun, Shu Li, Chenyi Li, Jiang Hu, Kun Yuan, Zaiwen Wen, and Pingwen Zhang. ReasFlow: Assisting Reasoning-Centric Scientific Discovery in Applied Mathematics via a Knowledge-Based Multi-Agent System, August 2026. URL https://arxiv.org/abs/2607.14178.

Herdr. Herdr: The runtime your coding agents live on, 2026.

Yuntong Hu, Yuting Zhang, Matthew Trager, Yi Zhang, Shuo Yang, Wei Xia, and Stefano Soatto. EvoMAS: Evolutionary Generation of Multi-Agent Systems, May 2026. URL https://arxiv.org/abs/2602.06511.

Vasily Ilin. Semi-Autonomous Formalization of the Vlasov-Maxwell-Landau Equilibrium, March 2026. URL https://arxiv.org/abs/2603.15929.

Haocheng Ju, Guoxiong Gao, Jiedong Jiang, Bin Wu, Zeming Sun, Shurui Liu, Leheng Chen, Yutong Wang, Yuefeng Wang, Zichen Wang, Wanyi He, Peihao Wu, Liang Xiao, Ruochuan Liu, Bryan Dai, and Bin Dong. Automated Conjecture Resolution with Formal Verification, May 2026. URL https://arxiv.org/abs/2604.03789.

Seth Karten, Joel Zhang, Tersoo Upaa, Ruirong Feng, Wenzhe Li, Chengshuai Shi, Chi Jin, and Kiran Vodrahalli. Continual Harness: Online Adaptation for Self-Improving Foundation Agents, May 2026. URL https://arxiv.org/abs/2605.09998.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines, October 2023. URL https://arxiv.org/abs/2310.03714.

Valentin Khrulkov, Andrey Galichin, Denis Bashkirov, Dmitry Vinichenko, Oleg Travkin, Roman Alferov, Andrey Kuznetsov, and Ivan Oseledets. GigaEvo: An Open Source Optimization Framework Powered By LLMs And Evolution Algorithms, November 2025. URL https://arxiv.org/abs/2511.17592.

Heechang Kim, Ernest K. Ryu, and Shuvomoy Das Gupta. A Domain-Specific Harness for End-to-End Automation of Optimization Research, August 2026. URL https://arxiv.org/abs/2608.07407.

Sungyoon Kim and Mert Pilanci. AI-Assisted Discovery of Convex Relaxations via Dual Agents, June 2026. URL https://arxiv.org/abs/2606.31182.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. ShinkaEvolve: Towards Open-Ended And Sample-Efficient Program Evolution, September 2025. URL https://arxiv.org/abs/2509.19349.

Gabriel Rongyang Lau. Using Aristotle API for AI-Assisted Theorem Proving in Lean 4: A Formalisation Case Study of the Grasshopper Problem, May 2026. URL https://arxiv.org/abs/2605.20120.

Ivan Lau and Jonathan Scarlett. Open problem: Is interaction necessary for order-optimal 1-bit mean estimation? In *Proceedings of the 39th Annual Conference on Learning Theory*, 2026. URL https://proceedings.mlr.press/v336/lau26a.html.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-End Optimization of Model Harnesses, March 2026. URL https://arxiv.org/abs/2603.28052.

Hanqi Li, Jing Peng, Zijian Wang, Lu Chen, and Kai Yu. XFlow: An Executable Protocol Programming System for Reliable Multi-Agent Workflows, June 2026. URL https://arxiv.org/abs/2606.14790.

Hongjun Liu, Yifei Ming, Shafiq Joty, and Chen Zhao. Harnessing LLM Agents with Skill Programs, May 2026a. URL https://arxiv.org/abs/2605.17734.

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, July 2026b. URL https://arxiv.org/abs/2607.06447.

Shu Liu, Shubham Agarwal, Monishwaran Maheswaran, Mert Cemri, Zhifei Li, Qiuyang Mang, Ashwin Naren, Ethan Boneh, Audrey Cheng, Melissa Z. Pan, Alexander Du, Kurt Keutzer, Alvin Cheung, Alexandros G. Dimakis, Koushik Sen, Matei Zaharia, and Ion Stoica. EvoX: Meta-Evolution for Automated Discovery, March 2026c. URL https://arxiv.org/abs/2602.23413.

Wentao Long, Yunfei Zhang, Chenyi Li, and Zaiwen Wen. MECA: A Mechanism-Centered Agent for Constructing Well-Specified and Valuable Mathematical Conjectures, July 2026. URL https://arxiv.org/abs/2607.27709.

Sichun Luo, Yi Huang, Haochen Luo, Fengyuan Liu, Guanzhi Deng, Lei Li, Qinghua Yao, Zefa Hu, Junlan Feng, and Qi Liu. SeaEvo: Advancing Algorithm Discovery with Strategy Space Evolution, April 2026. URL https://arxiv.org/abs/2604.24372.

Lve Meng, Weilong Zhao, Yanzhi Zhang, Haoxiang Guan, and Jiyan He. Can a Lightweight Automated AI Pipeline Solve Research-Level Mathematical Problems?, March 2026. URL https://arxiv.org/abs/2602.13695.

Konrad Mundinger, Sebastian Pokutta, Christoph Spiegel, and Max Zimmer. Extending the Continuum of Six-Colorings, April 2024. URL https://arxiv.org/abs/2404.05509.

Konrad Mundinger, Max Zimmer, Aldo Kiem, Christoph Spiegel, and Sebastian Pokutta. Neural Discovery in Mathematics: Do Machines Dream of Colored Planes?, June 2025. URL https://arxiv.org/abs/2501.18527.

John Forbes Nash, Jr. and Michael Th. Rassias (eds.). *Open Problems in Mathematics*. Springer International Publishing, 2018. ISBN 978-3-319-81210-6.

Allen Nie, Xavier Daull, Zhiyi Kuang, Abhinav Akkiraju, Anish Chaudhuri, Max Piasevoli, Ryan Rong, YuCheng Yuan, Prerit Choudhary, Shannon Xiao, Rasool Fakoor, Adith Swaminathan, and Ching-An Cheng. Understanding the Challenges in Iterative Generative Optimization with LLMs, March 2026. URL https://arxiv.org/abs/2603.23994.

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, June 2025. URL https://arxiv.org/abs/2506.13131.

OEIS Foundation Inc. A060655: Pack n integer-sided rectangles into the smallest possible square so that no sides of the rectangles are the same. The On-Line Encyclopedia of Integer Sequences, 2026. URL https://oeis.org/A060655.

OpenAI. On the navier–stokes millennium prize problem, September 2026a. URL https://openai.com/index/navier-stokes-solution/.

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

Jaan Parts. What percent of the plane can be properly 5- and 6-colored? *Geombinatorics*, 30(1):25–39, 2020. doi: 10.48550/arXiv.2010.12668. URL https://arxiv.org/abs/2010.12668.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, January 2024. ISSN 0028-0836, 1476-4687. doi: 10.1038/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, July 2026. URL https://arxiv.org/abs/2607.09474.

Yuchen Shi, Siqi Cai, Zihan Xu, Yuei Qin, Gang Li, Hang Shao, Jiawei Chen, Deqing Yang, Ke Li, and Xing Sun. FlowAgent: Achieving Compliance and Flexibility for Workflow Agents, February 2025. URL https://arxiv.org/abs/2502.14345.

Pavel Snopov and German Magai. Evaluating SageMath-Augmented LLM Agents for Computational and Experimental Mathematics, July 2026. URL https://arxiv.org/abs/2607.06820.

A Soifer. Six-realizable set X6. *Geombinatorics*, 3(4):140–145, 1994.

Nat Sothanaphan. Resolution of Erdős Problem #728: A writeup of Aristotle's Lean proof, January 2026. URL https://arxiv.org/abs/2601.07421.

George Tsoukalas, Anton Kovsharov, Sergey Shirobokov, Anja Surina, Moritz Firsching, Gergely Bérczi, Francisco J. R. Ruiz, Arun Suggala, Adam Zsolt Wagner, Eric Wieser, Lei Yu, Aja Huang, Miklós Z. Horváth, Andrew Ferraiuolo, Henryk Michalewski, Edward Lockhart, Codrut Grosu, Thomas Hubert, Matej Balog, Pushmeet Kohli, and Swarat Chaudhuri. Advancing Mathematics Research with AI-Driven Formal Proof Search, June 2026. URL https://arxiv.org/abs/2605.22763.

Hanyu Wang, Ruohan Xie, Yutong Wang, Guoxiong Gao, Xintao Yu, and Bin Dong. Aria: An Agent For Retrieval and Iterative Auto-Formalization via Dependency Graph, July 2026a. URL https://arxiv.org/abs/2510.04520.

Pengcheng Wang, Jerry Huang, Jiarui Yao, Rui Pan, Peizhi Niu, Yaowenqi Liu, Ruida Wang, Renhao Lu, Yuwei Guo, and Tong Zhang. AgentSPEX: An Agent SPecification and EXecution Language, April 2026b. URL https://arxiv.org/abs/2604.13346.

Yingxu Wang, Siwei Liu, Jinyuan Fang, and Zaiqiao Meng. EvoAgentX: An automated framework for evolving agentic workflows. In Ivan Habernal, Peter Schulam, and Jörg Tiedemann (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 643–655, Suzhou, China, November 2025a. Association for Computational Linguistics. ISBN 979-8-89176-334-0. doi: 10.18653/v1/2025.emnlp-demos.47.

Yinjie Wang, Ling Yang, Guohao Li, Mengdi Wang, and Bryon Aragam. ScoreFlow: Mastering LLM Agent Workflows via Score-based Preference Optimization, February 2025b. URL https://arxiv.org/abs/2502.04306.

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

Alizer Wong, Heng Cui, Yi Tan, Xiongchao Zhan, Liang Lin, Yuxiang Guo, Zhaorong Dai, Zixin Zeng, and Wenyuan Li. Eureka: Task-Conditioned Meta-Agent Orchestration for Scientific Discovery, August 2026. URL https://arxiv.org/abs/2608.19047.

Andy Xu and Yu-Wing Tai. Meta-Agent: From Task Descriptions to Verified Multi-Agent Systems, May 2026. URL https://arxiv.org/abs/2605.25233.

Jiawei Xu, Arief Koesdwiady, Sisong Bei, Yan Han, Baixiang Huang, Dakuo Wang, Yutong Chen, Zheshen Wang, Peihao Wang, Pan Li, and Ying Ding. Rethinking the Value of Multi-Agent Workflow: A Strong Single Agent Baseline, January 2026a. URL https://arxiv.org/abs/2601.12307.

Shengxiang Xu, Jiayi Zhang, Shimin Di, Yuyu Luo, Liang Yao, Hanmo Liu, Jia Zhu, and Min-Ling Zhang. RobustFlow: Towards Robust Agentic Workflow Generation, August 2026b. URL https://arxiv.org/abs/2509.21834.

Xiangzhe Xu, Hanxi Guo, Guangyu Shen, Siyuan Cheng, and Xiangyu Zhang. Natural-Language Workflows Are Not Software Yet: Artifact-Driven Compilation for Reliable Agent Execution, August 2026c. URL https://arxiv.org/abs/2608.21341.

Hui Xue and Fan Yang. Rethinking Self-Evolving Agents: Do We Still Need Prescribed Optimization Pipelines?, August 2026. URL https://arxiv.org/abs/2608.09629.

Simon Yu, Derek Chong, Ananjan Nandi, Dilara Soylu, Jiuding Sun, Christopher D. Manning, and Weiyan Shi. Shepherd: Enabling Programmable Meta-Agents via Reversible Agentic Execution Traces, June 2026. URL https://arxiv.org/abs/2605.10913.

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

Dechen Zhang, Xuan Tang, Xinxiang Yin, Xingwu Chen, Jian Qian, and Difan Zou. VALG: An Agentic System for ML Theory Research, August 2026a. URL https://arxiv.org/abs/2608.13060.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Godel Machine: Open-Ended Evolution of Self-Improving Agents, March 2026b. URL https://arxiv.org/abs/2505.22954.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating Agentic Workflow Generation, April 2025a. URL https://arxiv.org/abs/2410.10762.

Xi Zhang, Meijun Gao, Yuntian Zhao, Xinyu Tan, Yilun Yao, Feiyu Wang, Yanshu Wang, Dingsiyi, and Tong Yang. Formal Skill: Programmable Runtime Skills for Efficient and Accurate LLM Agents, May 2026c. URL https://arxiv.org/abs/2605.19604.

Yaolun Zhang, Xiaogeng Liu, and Chaowei Xiao. MetaAgent: Automatically Constructing Multi-Agent Systems Based on Finite State Machines, July 2025b. URL https://arxiv.org/abs/2507.22606.

Yuanhe Zhang, Jason D. Lee, and Fanghui Liu. AI4SLT: Empirical processes in Lean 4 for formal statistical learning theory. In *Forty-third International Conference on Machine Learning*, 2026d. URL https://openreview.net/forum?id=dfqmQ9WhCP.

Zelin Zhao, Bo Yuan, Jaemoo Choi, and Yongxin Chen. RMA: An Agentic System for Research-Level Mathematical Problems, May 2026. 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, Fernanda Viegas, Dimitri Kanevsky, Vineet Kahlon, Hartmut Maennel, Sophia Alj, George Holland, Alex Davies, and Pushmeet Kohli. AI co-mathematician: Accelerating mathematicians with agentic AI, May 2026. URL https://arxiv.org/abs/2605.06651.

Max Zimmer, Nico Pelleriti, Christophe Roux, and Sebastian Pokutta. The Agentic Researcher: A Practical Guide to AI-Assisted Research in Mathematics and Machine Learning, March 2026. URL https://arxiv.org/abs/2603.15914.

A CORPUS CONSTRUCTION

The corpus contains 230 included papers and 735 extracted mathematical case studies. Screening retained work in which AI contributed to a research-level mathematical result or formalization. Each selected source was processed separately to produce linked problem descriptions, solution approaches, and resource configurations.

The extraction snapshot contains 199 TeX-source packets, five PDF fallbacks, and 26 metadata-only packets. The latter reflect unavailable sources, extraction limits, or rate limits and may omit details of the reported methods. The records describe published claims and their reported validation, not independent reproductions. Repository inspection and local evaluation are separate steps.

A separate `source-observatory` snapshot queried GitHub for seven categories of mathematical research systems between July 10 and September 10, 2026. It returned 210 results and retained 207 unique candidate repositories after deduplication. Forks and archived repositories were excluded.

B EXPERIMENTAL SETUP, ROLES, AND VALIDATION

Table 2 separates the starting point of each study from the work performed by ZERO and its agents and from subsequent human review and validation.

Table 2: Starting points, system roles, and validation across the studies.

Study	Starting point	Workflow, agents, and tools	Human role and validation
Packing	Objective and existing Java search code, followed by a researcher-specified loop for studying related problems.	ZERO designed the multi-agent search workflow. Agents explored pairings, optimized placement, and reused or adapted the procedure across related packing problems.	Researchers supplied the objective, initial code, and later loop. Exact certificates checked the final constructions. For $n = 15$, the run used GPT 5.5 for 24 hours and recorded 21M tokens.
Hadwiger–Nelson	Planar objective only, followed by the successful procedure for later adaptations.	ZERO designed the initial workflow and its later adaptations. Agents pursued multiple approaches, developed the improved construction, adapted it to three dimensions, and searched for verified six-color intervals.	Researchers supplied the objective and later the successful procedure for adaptation. Exact geometric checks and certificates validate the constructions. The studies used GPT 6 Astra, with GPT Pro guidance in the planar study.
Lagrangian relaxation	Researcher-specified search → proof → verification → Lean procedure and a researcher-approved numerical checker.	ZERO turned the procedure into an executable workflow. Agents found counterexamples, developed arguments, and produced Lean formalizations.	Researchers specified the procedure, approved the numerical checker, and revised the generated exposition. Three developments compiled without admissions and passed axiom and statement checks.
VALG	Existing VALG system and a formal target fixed before formalization.	ZERO coordinated the formalization, repair, and integration workflows. Agents planned and implemented the Lean development.	Human source review found one construction defect and four statement mismatches. The final development was rebuilt and reviewed. Pi recorded 540.4M tokens. Aristotle telemetry is excluded.
AutoOPT	Existing AutoOPT system and the goal of replacing commercial solvers.	ZERO represented the stages as workflows. Agents implemented open-solver replacements, including the revised BnB-PEP and PSD-separation procedure.	Recovered results were checked with primal and dual PEPs, analytic comparisons, and reported optimization gaps.

C Discrete rectangle packing and optimality certificates

This appendix defines the rectangle-packing problem studied in Section 3.1 and proves optimality of our construction for $n = 15$. The problem is recorded as OEIS A060655.

C.1 Problem definition

Given $n \geq 1$, choose n axis-aligned rectangles whose $2n$ dimensions use each integer in $\{1, \ldots, 2n\}$ exactly once. Both the pairing of lengths into rectangles and their placement are optimization variables. Each rectangle may be rotated by 90°. The objective is to minimize the side length S_n of an enclosing square. Boundary contact is permitted, but rectangle interiors must be disjoint.

Write (w_i, h_i) for the oriented dimensions of rectangle i, and (x_i, y_i) for its lower-left corner. The precise optimization problem is

$$\begin{aligned} S_n = \min \quad & S \\ \text{subject to} \quad & (w_1, h_1, \ldots, w_n, h_n) \text{ is a permutation of } (1, \ldots, 2n), \\ & S > 0, \qquad x_i, y_i \geq 0 && (1 \leq i \leq n), \\ & x_i + w_i \leq S, \qquad y_i + h_i \leq S && (1 \leq i \leq n), \\ & (x_i + w_i \leq x_j) \ \vee \ (x_j + w_j \leq x_i) \\ & \qquad \vee \ (y_i + h_i \leq y_j) \ \vee \ (y_j + h_j \leq y_i) && (1 \leq i < j \leq n). \end{aligned} \tag{1}$$

Initially S, x_i, y_i may be real. The final disjunction requires a horizontal or vertical separation for every pair of rectangles. No particular pairing of the side lengths is prescribed.

One may equivalently allow any $2n$ distinct positive integer dimensions. If their ordered values are $\ell_1 < \cdots < \ell_{2n}$, then $\ell_k \geq k$. Replacing every occurrence of ℓ_k by k, while preserving the lower-left corners and orientations, only shrinks the rectangles. Thus restricting the dimensions to $1, \ldots, 2n$ does not increase the optimum.

C.2 Integer coordinates suffice

Every feasible real-coordinate packing can be replaced by an integer-coordinate packing in a square of no greater side length. To see this, select one satisfied separation inequality for each pair of rectangles. For every selected horizontal inequality $x_i + w_i \leq x_j$, place an edge $i \to j$ of weight w_i in a horizontal precedence graph. Add a source with an edge of weight zero to every rectangle. The graph is acyclic, since a directed cycle would contradict feasibility by summing inequalities with strictly positive widths.

Let x'_i be the longest-path distance from the source to i. These distances are nonnegative integers, satisfy every selected horizontal inequality, and obey $x'_i \leq x_i$. Apply the same construction to the selected vertical inequalities to obtain integers $y'_i \leq y_i$. Every rectangle pair retains its selected separation, so the new placement is feasible. Its required square side

$$S' = \max_{1 \leq i \leq n} \{x'_i + w_i, \ y'_i + h_i\}$$

is an integer and satisfies $S' \leq S$. Consequently the minimum exists and is unchanged if S, x_i, y_i are restricted to integers.

C.3 Area lower bound and the case $n = 15$

The minimum possible total rectangle area pairs the smallest available length with the largest, recursively. Indeed, if the smallest length a is paired with b and the largest d with c, replacing these pairs by (a, d) and (b, c) changes the area by

$$ad + bc - ab - cd = (d - b)(a - c) \leq 0.$$

Induction therefore gives

$$\sum_{i=1}^{n} w_i h_i \geq \sum_{i=1}^{n} i(2n + 1 - i) = \frac{n(n+1)(2n+1)}{3}. \tag{2}$$

Containment, disjoint interiors, and the integer-coordinate argument imply

$$S_n \geq \left\lceil \sqrt{\frac{n(n+1)(2n+1)}{3}} \right\rceil . \tag{3}$$

This area-minimizing pairing establishes a lower bound; other pairings remain admissible in the packing search.

For $n = 15$, the minimum area is 2480, and $49^2 < 2480 < 50^2$ gives $S_{15} \geq 50$. Table 3 supplies a packing in a square of side 50, proving

$$S_{15} = 50. \tag{4}$$

Its actual rectangle area is 2482. Optimality of the enclosing square does not require the construction itself to minimize total rectangle area.

The certificate can be checked using only integer comparisons. Verify the dimension permutation, the containment inequalities with $S = 50$, and the separation disjunction for each of the $\binom{15}{2} = 105$ pairs in Equation (1). For other values of n, a feasible packing similarly proves an upper bound on S_n; it proves optimality only when accompanied by a matching lower bound.

Table 3: Exact certificate for $S_{15} \leq 50$. Rows give (w_i, h_i, x_i, y_i); dimensions $1, \ldots, 30$ occur exactly once.

w_i	h_i	x_i	y_i
15	16	12	0
14	17	14	22
13	18	28	22
12	19	0	0
11	20	3	19
21	10	29	40
22	9	27	7
8	24	41	16
23	7	27	0
25	6	3	39
26	5	3	45
27	4	14	18
3	29	0	19
28	2	12	16
1	30	49	7

C.4 Complete construction inventory

Table 4 accounts for every construction produced in the packing study. Here L is the elementary area or volume lower bound. A dash means that we claim feasibility only, without a matching lower bound. The two entries for each of $n = 15$ and $n = 16$ in the cuboid family are distinct constructions. All coordinates below are integers, so containment and pairwise interior disjointness can be checked by exact comparisons.

Table 4: Complete accounting of the packing constructions. "Optimal" means that the displayed construction attains the stated elementary lower bound.

Family	Instance	S	L	Conclusion
Rectangles	$n = 15$	50	50	optimal
	$n = 18$	66	65	gap 1
	$n = 19$	71	71	optimal
	$n = 20$	77	76	gap 1
	$n = 21$	82	82	optimal
	$n = 22$	88	88	optimal
	$n = 23$	94	93	gap 1
	$n = 24$	100	99	gap 1
Cuboids	$n = 14$	43	–	feasible
	$n = 15$a	46	–	feasible
	$n = 15$b	46	–	feasible
	$n = 16$a	50	–	feasible
	$n = 16$b	51	–	feasible
Equilateral triangles	$n = 8$	15	15	optimal
	$n = 10$	21	20	gap 1
	$n = 12$	27	26	gap 1
	$n = 15$	37	36	gap 1
	$n = 16$	40	39	gap 1
	$n = 17$	43	43	optimal
	$n = 18$	47	46	gap 1
Tetrahedra	$n = 2$	3	3	optimal
	$n = 3$	5	4	feasible
	$n = 4$	7	5	feasible
	$n = 5$	9	7	feasible
	$n = 6$	11	8	feasible
	$n = 7$	13	10	feasible
	$n = 8$	15	11	feasible
	$n = 9$	17	13	feasible
	$n = 10$	19	15	feasible
	$n = 11$	21	17	feasible
	$n = 12$	24	19	feasible

Rectangle certificates. An entry $(w, h; x, y)$ denotes the closed rectangle $[x, x + w] \times [y, y + h]$; interiors must be disjoint. The $n = 15$ certificate is already displayed in Table 3. The remaining seven certificates are as follows.

Cuboid certificates. The cuboid variant uses the dimensions $1, \ldots, 3n$ exactly once. An entry $(w, h, d; x, y, z)$ denotes an axis-aligned cuboid with the given dimensions and lower corner, contained in $[0, S]^3$.

Equilateral-triangle certificates. Use lattice coordinates in $\Delta_S = \{(u, v) : u, v \geq 0,\ u + v \leq S\}$, embedded along two unit vectors at angle 60°. A tuple $(k, U; a, b)$ has vertices $(a, b), (a+k, b), (a, b+k)$; a tuple $(k, D; a, b)$ has vertices $(a + k, b), (a, b + k), (a + k, b + k)$. Thus each tuple specifies a complete equilateral triangle of side k.

Tetrahedron certificates. Use barycentric coordinates $\Sigma_S = \{q \in \mathbb{R}^4_{\geq 0} : \sum_i q_i = S\}$. For anchor a, the tuple $(k, U; a_1, a_2, a_3, a_4)$ is the convex hull of $a + ke_i$ for $i = 1, \ldots, 4$. The tuple $(k, D; a_1, a_2, a_3, a_4)$ is the convex hull of $a + k(\mathbf{1} - e_i)$ for $i = 1, \ldots, 4$. These are regular tetrahedra of side proportional to k in a containing regular tetrahedron of side proportional to S.

For the equilateral and tetrahedral families, the bounds in Table 4 follow respectively from $S^2 \geq \sum_{k=1}^{n} k^2$ and $S^3 \geq \sum_{k=1}^{n} k^3$. All displayed certificates use each required side length exactly once.

Table 5: Complete certificates for the rectangle constructions with $18 \leq n \leq 24$. Tuples are $(w, h; x, y)$.

n	S	Certificate
18	66	(20,18;0,0) (19,17;20,0) (21,16;39,0) (22,15;39,16) (25,14;0,18) (13,26;25,17) (24,12;38,31) (23,11;0,32) (32,10;0,43) (28,9;32,43) (29,8;32,52) (31,7;0,53) (30,6;0,60) (5,27;61,0) (4,33;62,27) (34,3;30,60) (35,2;30,63) (36,1;30,65)
19	71	(28,16;20,17) (23,18;48,15) (20,19;0,14) (22,17;25,0) (24,15;47,0) (25,14;0,0) (27,12;13,46) (29,11;40,47) (13,21;0,37) (30,9;13,37) (38,7;0,58) (26,10;43,37) (31,8;38,58) (36,6;1,65) (32,5;37,66) (35,4;34,33) (33,3;1,33) (2,37;69,33) (34,1;0,36)
20	77	(31,18;0,0) (23,19;31,0) (21,20;54,0) (24,17;0,18) (25,16;24,19) (26,15;49,20) (28,13;0,35) (29,12;28,35) (30,11;28,47) (40,8;0,58) (14,22;58,35) (32,9;40,58) (27,10;0,48) (34,7;0,66) (35,6;34,67) (5,36;72,35) (37,4;0,73) (38,3;37,73) (2,33;75,0) (39,1;37,76)
21	82	(26,19;0,12) (20,24;62,1) (31,15;27,64) (21,22;6,60) (27,17;31,47) (23,18;59,25) (29,14;30,29) (25,16;6,44) (30,13;0,31) (34,11;28,1) (28,12;0,0) (36,9;26,12) (10,32;58,47) (33,8;26,21) (7,37;73,44) (6,38;0,44) (5,35;68,47) (40,4;31,43) (41,3;27,79) (2,39;80,43) (42,1;32,0)
22	88	(22,23;0,35) (29,21;0,14) (20,27;0,58) (19,26;53,49) (18,25;20,60) (28,17;31,0) (16,24;22,36) (30,15;29,17) (31,14;0,0) (13,32;59,0) (33,12;40,36) (34,11;53,75) (10,35;72,0) (9,36;78,39) (8,37;38,48) (7,38;46,48) (6,39;82,0) (5,40;73,35) (41,4;31,32) (42,3;3,85) (43,2;45,86) (1,44;87,39)
23	94	(23,24;34,0) (25,22;59,22) (26,21;58,44) (27,20;32,24) (28,19;30,44) (29,18;0,44) (17,30;0,62) (31,16;0,28) (32,15;0,13) (33,14;61,65) (34,13;0,0) (35,12;58,10) (36,11;17,81) (37,10;57,0) (38,9;17,72) (39,8;55,79) (40,7;53,87) (41,6;17,63) (5,42;84,22) (4,43;89,22) (44,3;17,69) (45,2;1,92) (1,46;93,10)
24	100	(25,24;36,38) (26,23;1,34) (27,22;0,57) (28,21;62,3) (20,29;73,56) (30,19;0,79) (31,18;1,16) (17,32;73,24) (33,16;1,0) (34,15;66,85) (14,35;48,3) (36,13;30,85) (12,37;36,0) (11,38;62,24) (10,39;90,3) (40,9;32,70) (41,8;32,62) (7,42;93,43) (43,6;30,79) (5,44;27,35) (4,45;32,16) (46,3;48,0) (47,2;1,98) (1,48;0,9)

Table 6: Complete cuboid certificates. Tuples are $(w, h, d; x, y, z)$.

n	ID	S	Certificate
14	–	43	(42,41,1;0,0,0) (2,38,39;40,0,1) (3,35,36;0,0,1) (33,4,32;3,0,1) (5,30,29;17,13,1) (6,27,26;22,16,1) (15,7,40;0,36,1) (37,16,8;3,0,35) (17,9,34;3,4,1) (18,10,31;22,4,1) (11,24,23;28,16,20) (12,28,19;28,14,1) (25,20,13;3,16,30) (14,22,21;3,13,1)
15	a	46	(1,44,45;45,0,0) (2,41,42;43,4,0) (3,39,38;40,4,0) (35,4,36;10,0,0) (5,32,33;35,4,0) (30,6,29;7,40,17) (7,26,27;0,20,19) (43,16,8;0,4,38) (40,9,17;0,37,0) (10,18,37;0,0,0) (34,11,19;0,26,0) (31,20,12;7,20,33) (28,21,13;7,18,19) (24,14,23;10,4,15) (25,22,15;10,4,0)
15	b	46	(1,45,44;0,0,0) (43,40,2;0,0,44) (39,3,41;1,43,0) (4,42,38;1,0,0) (36,33,5;1,10,39) (34,6,23;12,37,14) (7,20,28;5,22,0) (8,35,18;38,0,0) (9,37,25;37,0,18) (30,10,16;5,0,28) (29,27,11;5,10,28) (32,17,12;5,1,15) (24,19,13;12,18,15) (26,21,14;12,22,0) (31,22,15;5,0,0)
16	a	50	(43,1,48;0,49,0) (47,2,45;0,0,0) (42,3,46;0,46,0) (38,4,34;0,2,0) (33,44,5;14,0,45) (6,25,40;44,25,0) (37,35,7;0,6,0) (41,16,8;9,6,27) (9,26,28;0,6,7) (36,19,10;14,2,35) (11,23,27;38,2,0) (12,13,39;0,33,7) (14,31,15;0,2,35) (30,24,17;14,22,28) (32,18,21;12,28,7) (29,22,20;9,6,7)
16	b	51	(48,47,1;3,1,50) (44,45,2;4,3,48) (41,3,42;1,0,0) (4,38,39;0,11,11) (5,36,35;46,0,0) (6,33,32;14,11,11) (29,7,30;14,3,17) (46,8,17;0,3,0) (9,43,18;4,3,30) (10,40,19;4,11,11) (20,37,11;0,11,0) (12,21,34;34,11,1) (27,26,13;20,10,35) (14,22,31;20,11,1) (28,15,23;20,36,24) (25,16,24;20,33,0)

Table 7: Complete equilateral-triangle certificates. Tuples are $(k, \text{orientation}; a, b)$.

n	S	Certificate
8	15	(1,D;0,6) (2,U;5,8) (3,D;2,7) (4,U;1,6) (5,U;0,10) (6,D;1,0) (7,U;0,0) (8,U;7,0)
10	21	(1,U;0,10) (2,U;0,19) (3,D;8,6) (4,U;11,6) (5,D;10,1) (6,U;15,0) (7,U;8,1) (8,D;0,1) (9,U;0,0) (10,U;1,9)
12	27	(1,U;0,12) (2,U;4,8) (3,U;1,12) (4,D;0,8) (5,U;13,0) (6,D;6,0) (7,D;11,0) (8,U;10,8) (9,U;18,0) (10,D;0,6) (11,U;0,16) (12,U;0,0)
15	37	(1,D;11,12) (2,U;11,13) (3,U;18,1) (4,U;15,0) (5,D;7,17) (6,D;0,23) (7,U;7,22) (8,U;0,29) (9,U;11,4) (10,U;14,12) (11,D;0,4) (12,D;12,0) (13,U;24,0) (14,U;0,15) (15,U;0,0)
16	40	(1,U;0,8) (2,D;1,4) (3,D;0,6) (4,D;4,0) (5,U;3,4) (6,D;3,3) (7,U;0,0) (8,U;9,0) (9,D;9,0) (10,U;18,0) (11,D;17,0) (12,U;28,0) (13,D;0,12) (14,U;14,11) (15,U;0,25) (16,U;0,9)
17	43	(1,U;6,15) (2,D;11,21) (3,U;18,12) (4,U;13,20) (5,U;17,21) (6,D;11,20) (7,U;11,15) (8,D;13,12) (9,U;21,12) (10,U;0,16) (11,D;0,15) (12,D;18,0) (13,U;30,0) (14,U;16,0) (15,D;1,0) (16,U;0,0) (17,U;0,26)
18	47	(1,D;19,26) (2,D;21,17) (3,U;21,19) (4,D;20,19) (5,D;15,10) (6,U;24,17) (7,D;23,10) (8,D;13,14) (9,U;21,10) (10,D;20,0) (11,U;12,23) (12,D;0,22) (13,U;0,34) (14,U;15,1) (15,D;0,1) (16,U;0,0) (17,U;30,0) (18,U;0,16)

Table 8: Complete tetrahedron certificates. Tuples are $(k, \text{orientation}; a_1, a_2, a_3, a_4)$.

n	S	Certificate
2	3	(1,D;0,0,0,0) (2,U;0,0,0,1)
3	5	(1,U;1,1,1,1) (2,U;0,0,3,0) (3,U;0,0,0,2)
4	7	(1,U;1,1,2,2) (2,U;0,2,2,1) (3,U;0,0,4,0) (4,U;0,0,0,3)
5	9	(1,U;1,1,3,3) (2,U;0,2,3,2) (3,U;4,2,0,0) (4,U;0,0,5,0) (5,U;0,0,0,4)
6	11	(1,D;1,1,1,5) (2,D;3,1,0,1) (3,U;2,1,2,3) (4,U;0,0,1,6) (5,U;3,3,0,0) (6,U;0,0,5,0)
7	13	(1,D;2,2,3,3) (2,D;7,0,0,0) (3,D;4,0,0,0) (4,U;9,0,0,0) (5,U;4,4,0,0) (6,U;2,0,5,0) (7,U;0,1,0,5)
8	15	(1,U;0,2,6,6) (2,U;0,8,0,5) (3,U;0,3,5,4) (4,U;6,4,0,1) (5,U;4,1,3,2) (6,U;0,6,2,1) (7,U;0,0,8,0) (8,U;0,0,0,7)
9	17	(1,D;0,0,10,4) (2,U;3,7,0,5) (3,D;0,4,4,0) (4,D;4,0,0,1) (5,U;2,0,10,0) (6,U;0,0,0,11) (7,U;1,1,4,4) (8,U;0,9,0,0) (9,U;8,0,0,0)
10	19	(1,U;8,0,8,2) (2,U;2,9,3,3) (3,D;0,6,2,2) (4,U;2,2,7,4) (5,D;1,0,1,2) (6,U;0,6,5,2) (7,U;1,11,0,0) (8,U;0,0,11,0) (9,U;6,2,2,0) (10,U;0,0,0,9)
11	21	(1,U;0,0,3,17) (2,U;0,3,0,16) (3,U;0,0,0,18) (4,U;0,5,9,3) (5,U;6,7,2,1) (6,U;5,2,7,1) (7,D;0,0,0,0) (8,U;0,0,13,0) (9,U;0,12,0,0) (10,U;11,0,0,0) (11,U;0,1,2,7)
12	24	(1,D;0,0,10,11) (2,U;0,2,10,10) (3,U;0,7,5,9) (4,U;0,4,8,8) (5,U;4,7,8,0) (6,U;10,8,0,0) (7,U;17,0,0,0) (8,D;0,0,0,0) (9,U;8,1,3,3) (10,U;0,10,2,2) (11,U;0,0,12,1) (12,U;0,0,0,12)

D HADWIGER–NELSON VARIANTS: DEFINITIONS AND OBJECTIVES

This appendix specifies the three construction problems studied in Section 3.2: almost-five-coloring of the plane, almost-14-coloring of three-dimensional space, and the continuum of six-colorings of the plane. Throughout, distance means Euclidean distance. A proper unit-distance k-coloring of $\mathbb{R}^n$ is a map $c : \mathbb{R}^n \to \{1, \dots, k\}$ such that

$$\|x - y\|_2 = 1 \quad \Longrightarrow \quad c(x) \neq c(y) \qquad \text{for every } x, y \in \mathbb{R}^n. \tag{5}$$

The least such k is the chromatic number $\chi(\mathbb{R}^n)$; the classical Hadwiger–Nelson problem asks for $\chi(\mathbb{R}^2)$. The variants below either permit an explicitly measured uncolored region or change the distance forbidden in one color class.

D.1 ALMOST-COLORINGS AND THE REMOVED FRACTION

An *almost-k-coloring* consists of a measurable removed set $R \subseteq \mathbb{R}^n$ and a measurable map $c : \mathbb{R}^n \setminus R \to \{1, \dots, k\}$ satisfying Equation (5) for every pair of points in its domain. Equivalently, extend c to all of $\mathbb{R}^n$ by assigning the label 0 to R. The exact constraint is then

$$c : \mathbb{R}^n \to \{0, 1, \dots, k\}, \qquad \|x - y\|_2 = 1 \quad \Longrightarrow \quad c(x) \neq c(y) \text{ or } c(x) = c(y) = 0. \tag{6}$$

Thus there are k constrained colors and one unconstrained removal label. This is the discrete coloring underlying the additional-color formulation of Mundinger et al. (2025, Section 3.3).

For the measurable periodic constructions considered here, let $A \in \mathbb{R}^{n \times n}$ be invertible and let $\Lambda = A\mathbb{Z}^n$ be a period lattice, so that $c(x + Az) = c(x)$ for every $x \in \mathbb{R}^n$ and $z \in \mathbb{Z}^n$. Choose the half-open fundamental cell $F = A[0, 1)^n$. Writing μ_n for n-dimensional Lebesgue measure, the *removed fraction* is

$$\rho(c, A) = \frac{\mu_n(R \cap F)}{\mu_n(F)} = \frac{1}{|\det A|} \int_F \mathbf{1}_{\{c(x)=0\}} \, \mathrm{d}x. \tag{7}$$

This fraction is independent of the choice of fundamental cell for the period lattice. It is also the limiting fraction removed from expanding Euclidean boxes. The periodic optimization objective is

$$\rho^{\mathrm{per}}_{n,k} = \inf_{(c,A)} \rho(c, A), \tag{8}$$

where the infimum ranges over all invertible A and measurable, $A\mathbb{Z}^n$-periodic c satisfying Equation (6). An explicit admissible construction supplies an upper bound on this infimum; optimizing within a chosen representation need not attain it. Periodicity specifies the class of constructions evaluated here, rather than a requirement of the underlying unit-distance problem.

Planar almost-five-coloring. Here $(n, k) = (2, 5)$: five color classes must avoid unit distance, while $R = c^{-1}(0)$ is the uncolored part of the plane. The objective is to minimize its area fraction $\rho(c, A)$, or equivalently maximize the colored fraction $1 - \rho(c, A)$. Parts (2020) studied this question for five and six colors. The five-color construction of Mundinger et al. (2025, Sections 2 and 4.1) reduced the reported removed fraction from approximately 4.0060% to 3.7356%. These are construction bounds, not claimed optima.

Three-dimensional almost-14-coloring. Here $(n, k) = (3, 14)$: the map takes values in $\{0, 1, \dots, 14\}$, and each of the fourteen nonzero color classes must avoid unit distance in $\mathbb{R}^3$. The objective is the removed *volume* fraction in Equation (7). This combines the higher-dimensional unit-distance problem with the almost-coloring relaxation. Mundinger et al. (2025, Section 4.3 and Appendix D) report an almost-14-coloring with removed fraction 3.4622%. A construction with a nonempty removed region does not by itself give a proper 14-coloring of all of $\mathbb{R}^3$.

D.2 THE CONTINUUM OF SIX-COLORINGS

For positive numbers $d_1, \dots, d_k$, a coloring $c : \mathbb{R}^2 \to \{1, \dots, k\}$ has *type* $(d_1, \dots, d_k)$ if

$$c(x) = c(y) = i \quad \Longrightarrow \quad \|x - y\|_2 \neq d_i \qquad \text{for every } x, y \in \mathbb{R}^2 \text{ and } i \in \{1, \dots, k\}. \tag{9}$$

Each color has its own forbidden distance. The six-color variant fixes $(d_1, \ldots, d_6) = (1, 1, 1, 1, 1, d)$ and asks which $d > 0$ are realizable. The *continuum of six-colorings*, or *six-realizable set*, is

$$X_6 = \big\{d > 0 : \text{there exists a coloring of } \mathbb{R}^2 \text{ of type } (1, 1, 1, 1, 1, d)\big\}. \tag{10}$$

All points receive a color, including boundary points; there is no removed region. The sixth color must avoid distance d, although it may contain unit-distance pairs when $d \neq 1$. In particular, $1 \in X_6$ would imply $\chi(\mathbb{R}^2) \leq 6$. The definition of X_6 does not impose measurability or periodicity; a valid measurable periodic construction is one way to establish membership.

Origin and difficulty. Soifer (1994) posed the problem of determining X_6. Soifer's discussion in *Open Problems in Mathematics* (Nash & Rassias, 2018) describes this problem as "still open and extremely difficult." The previously established interval was $[\sqrt{2} - 1,\ 1/\sqrt{5}] \subseteq X_6$. Mundinger et al. (2024, Theorem 1.1) extended it to

$$[0.354,\ 0.657] \subseteq X_6, \tag{11}$$

the first reported extension in thirty years. The term "continuum" names the realizability problem; it does not assume that the entire set X_6 is a single interval.

What an interval construction establishes. To certify $I \subseteq X_6$ for an interval $I \subset (0, \infty)$, it suffices to construct a family satisfying

$$\text{for every } d \in I \text{ there exists } c_d \text{ of type } (1, 1, 1, 1, 1, d). \tag{12}$$

The coloring may depend on d. A single coloring valid for every $d \in I$ is a stronger witness: its sixth color avoids every distance in I simultaneously. The two constructions in Mundinger et al. (2024) illustrate this distinction: one varies with d over $[0.354, 0.553]$, whereas the other is fixed over $[0.418, 0.657]$. The construction objective is to certify additional values or intervals in X_6, for example by lowering a known interval's lower endpoint or raising its upper endpoint. Establishing new members does not characterize all of X_6 or exclude values outside the certified range.

D.3 Pointwise validity of periodic constructions

All three variants require constraints for *every* relevant pair of points. For a periodic construction, set $P_i = F \cap c^{-1}(i)$. The complete color class is $\bigcup_{z \in \mathbb{Z}^n} (P_i + Az)$, and its forbidden-distance condition is exactly

$$\|x - y + Az\|_2 \neq d_i \qquad \text{for every } x, y \in P_i \text{ and } z \in \mathbb{Z}^n. \tag{13}$$

For the almost-colorings, $d_i = 1$ for $i = 1, \ldots, k$ and the removal label is unrestricted. For six-colorings, the six distances are as in Equation (10). This formulation includes pairs crossing fundamental-cell boundaries and all lattice translates. Any reduction to finitely many geometric checks must justify why the remaining translates cannot realize a forbidden distance.

Boundary assignments are part of the coloring: sets of measure zero do not change the removed fraction, but can still contain a forbidden pair. Similarly, a low sampled conflict rate or a small neural training loss is not a substitute for Equation (13). The reported almost-coloring objective is the measure of the removed set in a valid construction; an interval claim for six-colorings must cover every parameter value in that interval, including any claimed endpoints.

E Geoffrion counterexamples and formal certificates

We give the three constructions used in Section 3.3. As in (Dey et al., 2025), $v^\star$ denotes the infimum over $\mathrm{conv}(X)$ subject to $Ax \geq b$, and $\bar{v}^\star$ uses its closure. Neither notation denotes the original integer optimum. The Lagrangian convention is $c^\top x + \lambda^\top (b - Ax)$ with $\lambda \geq 0$.

E.1 Two rational constraints

Let $\ell(u, v) = \sqrt{2}u + v$ and

$$K = \{(u, v, r, z) \mid 0 \leq z \leq 1,\ r \geq 0,\ \ell(u, v) \geq 0, r\ell(u, v) \geq (1 - z)^2\}, \qquad X = K \cap \mathbb{Z}^4.$$

The last three conditions express positive semidefiniteness of the matrix with diagonal entries $r, \ell(u,v)$ and off-diagonal entry $1-z$. Thus K is closed and convex. We minimize z subject to $u \le 0$ and $v \le 0$. On this constraint set, $\ell = 0$ forces $u = v = 0$ and then $z = 1$. Since $\overline{\mathrm{conv}}(X) \subseteq K$ and $(0,0,0,1) \in X$, the integer optimum and both convexified values are one. The unconstrained infimum over X is zero, attained, for example, at $(1,-1,3,0)$.

The projection of the $z=0$ integer points onto (u,v) is exactly

$$L = \{(u,v) \in \mathbb{Z}^2 \mid \sqrt{2}u + v > 0\}.$$

Indeed, any such pair can be extended by an integer $r \ge 1/\ell(u,v)$. Write $n = (\sqrt{2}, 1)$. For nonnegative multipliers, the dual function is

$$g(\lambda) = \inf_{x \in X}(z + \lambda_1 u + \lambda_2 v) = \begin{cases} 0, & \lambda \in \mathbb{R}_+ n, \\ -\infty, & \lambda \notin \mathbb{R}_+ n. \end{cases}$$

To prove the first case, write $\lambda = tn$ with $t \ge 0$. The objective is nonnegative on X. The integers defined by $p_k + q_k\sqrt{2} = (1+\sqrt{2})^{2k+1}$ satisfy $p_k^2 - 2q_k^2 = -1$, so

$$0 < \sqrt{2}q_k - p_k = \frac{1}{\sqrt{2}q_k + p_k} \longrightarrow 0.$$

Taking $(u,v) = (q_k, -p_k)$ and a sufficiently large integer r gives $z=0$ points whose Lagrangian values tend to zero. For the second case, the open cone $\{d \mid n \cdot d > 0,\ \lambda \cdot d < 0\}$ is nonempty. Otherwise λ would vanish on $\ker(n)$ and be a nonnegative multiple of n. The cone contains an integer vector d, and the points $(kd_1, kd_2, 1, 0)$ belong to X for sufficiently large integers k, with Lagrangian values tending to $-\infty$. Consequently $v^L = 0 < 1 = \bar{v}^\star = v^\star$.

E.2 Removing attainment

For integers $k \ge 1$, set $\ell_k(r) = (2k+1-r)/(k(k+1))$ and define

$$C_0 = \{(r,s,t) \in \mathbb{R}^3 \mid r \ge 1,\ s \le 0,\ t \ge -1,\ t \ge \ell_k(r) + ks \text{ for all } k \ge 1\}.$$

This set is closed and convex. On $s=0$, its lower boundary is $f(r) = \sup_k \ell_k(r)$, with $f(r) = \ell_k(r)$ for $r \in [k, k+1]$. This follows by comparing consecutive lines,

$$\ell_k(r) - \ell_{k+1}(r) = \frac{2(k+1-r)}{k(k+1)(k+2)}.$$

Thus $f(k) = 1/k$, and $f(r) > 0$ for every finite $r \ge 1$. Minimizing t over C_0 with the additional constraint $s \ge 0$ has infimum zero and no minimizer. The set is locally polyhedral. In a bounded box with $r \le R$, choose an integer $N > R+1$. For each r, an index $j \le N$ attains the upper envelope. For $k > N$, $\ell_k(r) \le \ell_j(r)$ and $ks \le js$, so only finitely many defining inequalities are needed within the box.

For each $\lambda \ge 0$, the infimum of $t - \lambda s$ over C_0 is -1. The lower bound follows from $t \ge -1$ and $s \le 0$. For the reverse bound, fix $\varepsilon > 0$. For all sufficiently large r, $(r, -\varepsilon, -1) \in C_0$. To see this, $\ell_k(r) \le 2/(k+1)$ handles all sufficiently large k in the inequalities $\ell_k(r) - k\varepsilon \le -1$, and increasing r handles the finitely many remaining indices. Hence the infimum is at most $-1 + \lambda\varepsilon$, and letting ε tend to zero proves the claim.

To obtain an integer instance, define

$$T(x_1,x_2,x_3,x_4) = (x_1 + \sqrt{2}x_4,\ x_2 + \sqrt{3}x_4,\ x_3 + \sqrt{6}x_4), \qquad X = T^{-1}(C_0) \cap \mathbb{Z}^4.$$

We have $\overline{\mathrm{conv}}(X) = T^{-1}(C_0)$. The inclusion from left to right follows from closedness and convexity. For the reverse inclusion, put $\theta = (\sqrt{2}, \sqrt{3}, \sqrt{6})$. The rational independence of $1, \sqrt{2}, \sqrt{3}, \sqrt{6}$ implies by Kronecker density that integer translates $p + q\theta$ approximate any point of $\mathbb{R}^3$ with q arbitrarily large positive or negative. For an interior point $y \in C_0$, choose integer points $a_j^\pm = (p_j^\pm, q_j^\pm)$ with $Ta_j^\pm \to y$ and $q_j^\pm \to \pm\infty$. They eventually lie in X. For any fixed fourth coordinate τ, take the convex combination with weight

$$\alpha_j = \frac{\tau - q_j^-}{q_j^+ - q_j^-}.$$

Its fourth coordinate is τ and its image under T tends to y, so it converges to the unique point with that image and fourth coordinate. This proves inclusion of every fiber over the interior. The point $(2, -1, 0)$ has an ℓ_∞ neighborhood of radius $1/10$ in C_0. Interpolating toward it and taking limits extends the inclusion to the boundary. Local polyhedrality of $T^{-1}(C_0)$ follows because the image of a bounded box under T is bounded.

Set $A = (0, 1, 0, \sqrt{3})$, $b = 0$, and $c = (0, 0, 1, \sqrt{6})$. The constraint $Ax \geq 0$ and objective $c^\top x$ become $s \geq 0$ and t under T. Therefore $\bar{v}^\star = 0$ is not attained. The infimum of a continuous linear function is unchanged by taking the closed convex hull, so every Lagrangian subproblem has value -1 and $v^L = -1$. The integer problem is feasible at $(1, 0, 1, 0)$. Integer feasibility forces $x_2 = x_4 = 0$, and then its positive integer objective has minimum one. This gives the required nonattainment counterexample with real constraint and objective coefficients.

E.3 A FINITELY DESCRIBED POLYHEDRAL EXAMPLE

In coordinates (u, v, r, y, z), define the polyhedron

$$Q = \left\{ (u, v, r, y, z) \in \mathbb{R}^5 \;\middle|\; \begin{array}{l} \sqrt{2}u + v \geq 0, \quad r \geq u, \quad r \geq -u, \quad 0 \leq z \leq 1, \\ \frac{1}{4} - \frac{z}{4} \leq y - \sqrt{3}u \leq \frac{3}{4} + \frac{z}{4} \end{array} \right\}.$$

Set $X = Q \cap \mathbb{Z}^5$ and minimize z subject to $u \leq 0$ and $v \leq 0$. The unconstrained infimum is zero, attained at $(1, -1, 1, 2, 0)$. The additional constraints force $u = v = 0$. At $z = 0$, the strip would require an integer $y \in [1/4, 3/4]$, which is impossible. The point $(0, 0, 0, 0, 1)$ is feasible, so the integer optimum is one.

The closed-convexified value is also one. Suppose convex combinations $x_j \in \mathrm{conv}(X)$ converge to $(0, 0, r, y, z)$ with $z < 1$. Since integer points have $z \in \{0, 1\}$, the weight on their $z = 0$ terms tends to $\alpha = 1 - z > 0$. The bound $r_i \geq |u_i|$ and boundedness of the averaged r-coordinates imply that, for a sufficiently large fixed M, at least $\alpha/4$ of this weight has $|u_i| \leq M$. On these $z = 0$ points, $\sqrt{2}u_i + v_i$ has a positive lower bound η_M. Equality to zero would force $u_i = v_i = 0$, contradicting the strip condition, and only finitely many integer u_i occur. All other terms have nonnegative $\sqrt{2}u_i + v_i$. Thus $\sqrt{2}u_j + v_j \geq \alpha\eta_M/4$, contradicting convergence to zero. It follows that every closed-convexified feasible point has $z \geq 1$. The feasible integer point above then gives $v^\star = \bar{v}^\star = 1$.

For the dual value, write $d = \lambda_1 - \sqrt{2}\lambda_2$. Choose nonzero integers u with $\{\sqrt{3}u\} \in [1/4, 3/4]$ and set

$$v = \lceil -\sqrt{2}u \rceil, \quad r = |u|, \quad y = \lfloor \sqrt{3}u \rfloor + 1, \quad z = 0.$$

These points lie in X and have Lagrangian value $du + \lambda_2\{\sqrt{2}u\}$. Irrational rotation provides admissible u arbitrarily far in both directions. If $d \neq 0$, choosing the appropriate direction makes this value tend to $-\infty$. If $d = 0$, the Lagrangian is nonnegative everywhere on X. Joint density of $(\sqrt{2}u, \sqrt{3}u)$ modulo one supplies admissible points with $\{\sqrt{2}u\} \to 0$, proving that its infimum is zero. Hence $v^L = 0 < 1 = \bar{v}^\star = v^\star$.

The polyhedron Q is genuinely non-rational. Its slice $z = 1$, projected onto (u, y), is the strip $0 \leq y - \sqrt{3}u \leq 1$, whose recession line has irrational slope. Such a strip cannot be a projection of a rational polyhedron. The stronger assertion that X cannot equal the integer points of any rational polyhedron follows from semilinearity. The corresponding integer strip is infinite but contains no infinite arithmetic progression with a nonzero integer direction, since boundedness of $y - \sqrt{3}u$ would require an integer direction (a, b) satisfying $b = \sqrt{3}a$. It is therefore not semilinear, whereas projections of the integer points of rational polyhedra are semilinear. This argument is separate from the Lean value certificate.

E.4 FORMALIZATION AND VERIFICATION

Formalization used a workflow separate from the mathematical prover–verifier procedure. Workers completed Lean proof obligations and used compiler feedback to repair them. The recorded formalization gates checked compilation, absence of admissions and custom axioms, and preservation of frozen target statements. The nonattainment proof initially lacked the required two-sided density

```
{
  "version": 1,
  "name": "prover-verifier",
  "inputs": {
    "problem": { "required": true, "schema": { "type": "string" } }
  },
  "profiles": {
    "prover": { "kind": "pi" }, "verifier": { "kind": "codex" }
  },
  "steps": [{
    "id": "proof_loop",
    "kind": "loop",
    "maxIterations": 3,
    "until": { "ref": "steps.verify.json.passed", "op": "eq", "value": true },
    "steps": [
      {
        "id": "prove", "kind": "agent", "profile": "prover",
        "prompt": "Prove {input.problem}. Address this prior review: {steps.verify.json.
     feedback?}"
      },
      {
        "id": "verify", "kind": "agent", "profile": "verifier",
        "prompt": "Check this proof: {steps.prove.output}",
        "resultSchema": { "type": "object", "required": ["passed", "feedback"],
          "properties": { "passed": { "type": "boolean" }, "feedback": { "type": "string" } }
      }
      }
    ]
  }]
}
```

Figure 7: A minimal executable prover–verifier loop in ZERO's Herdr workflow format. Rejected proofs and verifier feedback are available to the next prover iteration; the loop stops when the verifier accepts or fails after three attempts.

lemma. Aristotle supplied a torus-density development, and this result was reused in the final five-variable project. The collected projects report successful checks with axiom dependencies restricted to `propext`, `Classical.choice`, and `Quot.sound`.

The five-variable certificate proves the unconstrained, integer, closed-convexified, and dual values. The ordinary convexified value follows by inclusion between the integer set and its closed convex hull.

F ADDITIONAL WORKFLOW REPRESENTATIONS

In addition to evolutionary program improvement, we represented prover–verifier loops and IM-ProofBench ProofCouncil, UCLA Moonshot, and Princeton Momus, the three FirstProof Batch 2 multi-agent submissions, as workflows (Abouzaid et al., 2026b). The representations expose agent roles, resource assignments, information exchange, persistent artifacts, review stages, and continuation conditions as workflow components. The Princeton representation includes parallel attempts, synthesis, separate reviews, a shared notebook, and a conjecture stage triggered by stagnation. Mathematicians reviewed each workflow for fidelity to the published system description. We also ran all three FirstProof workflows on a selected Batch 2 problem and had their outputs independently assessed for correctness.

These executions evaluate the represented procedures on a common task, rather than reproduce the original submissions under their reported models and budgets. Our Princeton representation uses a smaller default search budget, omits a child-lineage conjecture mode, and requires explicit model assignments for cross-provider review. The workflows allow individual workers and review stages to be replaced, extended, or composed with other procedures.

G VALG FORMALIZATION AND VALIDATION

G.1 FORMAL TARGET AND IMPLEMENTATION

The target in Section 3.4 includes a concrete non-adaptive one-bit protocol, a deterministic sample size, and a uniform probability guarantee in all three moment regimes. The development also includes two baseline reductions. It establishes an upper bound, not a minimax lower bound. The source statement was fixed before formalization.

Let $\mathcal{D}(k, \lambda, \sigma)$ be the laws on $\mathbb{R}$ with mean $\mu(D) \in [-\lambda, \lambda]$ and $\mathbb{E}_D|X - \mu(D)|^k \le \sigma^k$. We formalized the following upper bound.

Candidate theorem 1 (Fully non-adaptive one-bit mean estimation)**.** *For every fixed real $k > 1$, there are constants $c_k, C_k > 0$ such that, for known $\lambda \ge \sigma > 0$, $0 < \epsilon \le c_k\sigma$, and $0 < \delta < 1/2$, a fully non-adaptive protocol using one precommitted Borel query bit per independent observation returns $\widehat{\mu}$ with*

$$\sup_{D \in \mathcal{D}(k,\lambda,\sigma)} \Pr_D(|\widehat{\mu} - \mu(D)| > \epsilon) \le \delta.$$

Its deterministic sample size satisfies

$$n \le C_k \left[\log \frac{\lambda}{\sigma} + \begin{cases} \dfrac{\sigma^2}{\epsilon^2} \log \dfrac{1}{\delta}, & k > 2, \\ \dfrac{\sigma^2}{\epsilon^2} \log \dfrac{\sigma}{\epsilon} \log \dfrac{1}{\delta}, & k = 2, \\ \left(\dfrac{\sigma}{\epsilon}\right)^{k/(k-1)} \log \dfrac{1}{\delta}, & 1 < k < 2. \end{cases} \right]$$

Target selection and proof planning compared the one-bit theorem with a Pfaffian alternative. The one-bit theorem supplied more of the required probability and algorithmic structure, and the SLT library provided relevant measure-theoretic results. The planning workflow divided the proof into modules and reviewed the mathematical arguments and proposed Lean interfaces. A persistent Pi `/goal` run implemented the proof using compiler feedback.

G.2 SOURCE REVIEW AND REPAIRS

The first complete development compiled, but source review found a defect in an auxiliary probability law. The law had been defined by the pushforward that a later theorem was intended to establish. Although the outer probability bound used the primitive law directly, this did not supply the intended construction of the auxiliary law. A repair workflow specified that law separately and proved the required identity.

A further review found four discrepancies with the source. The variance constants differed, the public theorem omitted design conditions, the probability guarantee lacked the required real-valued formulation, and the baseline interfaces omitted several identities. Separate workers repaired these issues. An integration agent combined the changes and rebuilt the development.

The final development contains 36 modules and 7,853 nonblank, noncomment lines and compiles without admissions in the pinned Lean 4.32.0 environment. Local checks examined the declarations and axiom dependencies. Human review and a separate GPT Pro review compared the formal statement and construction with the source. GPT Pro did not have Lean available and did not independently reproduce the build. Its assessment was source review, with compilation and axiom logs supplied as supporting evidence.

G.3 RECORDED MODEL USE

Table 9 reports Pi session tokens, including cache-read context. These counts are neither unique generated tokens nor monetary costs. Aristotle jobs lack comparable token telemetry and are excluded.

Table 9: Recorded Pi model use for VALG formalization and review, including cached context.

Stage	Pi session tokens
Planning and reviews	122.4M
Persistent formalization	394.6M
Auxiliary-law repair and review	6.35M
Further repairs, integration, and reviews	17.0M
Total (rounded)	540.4M

H AutoOPT with open solvers

H.1 Solver replacement

AutoOPT's numerical procedure evaluates fixed methods through performance estimation problems (PEPs), searches locally over algorithm coefficients, and uses branch and bound for global method design. We used Julia 1.11.1 and JuMP 1.31.1 with Clarabel 0.11.1, Ipopt 1.15.0, and SCIP 10.0.3 through SCIP.jl 0.12.8. Table 10 lists their roles.

The global stage required more than a solver substitution. A direct formulation with semidefinite slack $Z = PP^\top$ found candidate solutions but gave weak lower bounds. We implemented the two-pass BnB-PEP procedure. The first pass keeps Z explicit. When a candidate has an eigenpair (λ, u) with $\lambda < -10^{-7}$, a SCIP constraint handler adds

$$u^\top Z u \geq 0.$$

This inequality is valid for positive-semidefinite matrices and excludes the violating candidate. A second pass uses the factorization $Z = PP^\top$ to check the candidate method against the lower bound.

The implementation reconstructs the computation from the published equations and available BnB-PEP implementation. It is not a replay of the original AutoOPT runs. For horizons $N = 3, 4, 5$, it retains heuristic support restrictions on interpolation multipliers and the Cholesky factor. The resulting gaps apply to those restricted search problems.

Table 10: Open replacements and their validation in the AutoOPT reproduction.

Stage	Open backend	Output check
Fixed primal and dual PEP	Clarabel	primal–dual agreement
Convex ITEM-f design	Clarabel	fixed PEP at recovered coefficients
Local lemniscate design	Ipopt	analytic target and fixed PEP
Global lower-bound pass	SCIP with PSD separation	incumbent, bound, and gap
Final factorized QCQP	SCIP	fixed primal and dual PEP

H.2 Numerical comparisons and checks

For ITEM-f with $L = 1$, $\mu = 0.1$, and $N = 1, \ldots, 5$, the unrestricted convex design SDP returned optimal status at each horizon. The largest differences from the analytic objective and coefficients were 1.962×10^{-11} and 2.490×10^{-10}, respectively.

For lemniscate acceleration, the published coefficient table for $N = 1, \ldots, 7$ agrees to its four-decimal precision. Ipopt recovered local designs through $N = 25$, with maximum target discrepancy 3.817×10^{-9}. Evaluating the analytic dual identities through $N = 25$ gave a maximum matrix residual of 1.055×10^{-14}.

Table 11 reports the global-search results. Horizons $N = 1, 2$ use full multiplier and Cholesky support; $N = 3, 4, 5$ use the restricted support described above. All five runs meet the requested 1% relative-gap tolerance.

We checked fixed methods by solving primal and dual PEPs. Recovered algorithm coefficients were also evaluated in separately assembled fixed-method PEPs, and the analytic lemniscate certificate was

Table 11: Open SCIP reproduction of the lemniscate global-search stage.

N	Support	Incumbent	Lower bound	Relative gap
1	full	0.1715728696	0.1715569603	9.27×10^{-5}
2	full	0.05770784979	0.05770784979	0
3	source-style	0.02561688140	0.02561687189	3.71×10^{-7}
4	source-style	0.01332026836	0.01331897935	9.68×10^{-5}
5	source-style	0.007684506692	0.007608556359	9.88×10^{-3}

checked entry by entry. For global runs, checks included the feasible objective, lower bound, relative gap, and retained support. These are numerical comparisons at the stated horizons and tolerances, not formal verification or a certificate of unrestricted global optimality.

I IMPLEMENTATION DETAILS

ZERO's harness-agnostic runtime is implemented in Herdr, which provides a common interface for launching and accessing interactive agent sessions (Herdr, 2026). Agent nodes may return an explicit structured handoff; when they do not, Herdr derives the node outcome from the agent's final state. The runtime also maintains the persistent workspace and workflow state, schedules ready nodes, and routes files and results between them.

We used ZERO's system-augmentation mechanism during its own development. An early Pi-specific implementation was migrated with ZERO's assistance to the current Herdr-based runtime, separating the workflow layer from the underlying agent harness.