TCSEXAM: EVALUATING FRONTIER LLMS ON THE FULL RANGE OF TCS RESEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Can large language models (LLMs) advance theoretical computer science and resolve its long-standing open conjectures? To study progress toward this goal, we introduce TCSExam, a unified framework and benchmark covering the full range of TCS research. It evaluates *theorem proving*, *solver engineering*, and *constructions* alongside direct attempts to prove or refute *open conjectures*. TCSExam comprises 100 expert-reviewed tasks, with submissions checked by the Lean kernel or an executable task checker. Our evaluation of six frontier LLMs reveals substantial differences in theorem proving. When no proof outline is supplied, the strongest model produces complete, verified proofs in 70% of attempts, compared with at most 11.7% for any other model. Submitted solvers improve on selected public competition solutions, and constructions surpass selected reference bounds. None of the six models fully proves or refutes any of the 30 open conjectures under a budget of up to 12 hours per attempt. Individual attempts nevertheless yield independently verified mathematical results, including special cases and explicit bounds. With shared task interfaces and reproducible verification, TCSExam provides an extensible foundation for developing and evaluating LLMs for autonomous TCS research.

1 INTRODUCTION

Can large language models (LLMs) resolve long-standing *open conjectures* and advance theoretical computer science? Studying progress toward this ambition poses a challenge for evaluation: when a conjecture remains unresolved, the final outcome alone reveals little about which research tasks an LLM can perform or where its limitations lie. An informative evaluation should therefore examine concrete research accomplishments alongside direct attempts to prove or refute open conjectures.

Three forms of research output provide concrete ways to examine these questions. *Theorem proving* requires complete arguments for research-level mathematical statements. *Solver engineering* produces reusable programs whose effectiveness is tested across problem instances. *Constructions* provide explicit mathematical objects with verified properties that establish or improve bounds. Each output has independent research value, and the methods used to obtain them can interact: solvers search for candidate objects, proofs establish their properties, and constructions can supply witnesses or counterexamples. Evaluating these outputs helps TCS researchers judge which research work LLMs can undertake and helps model developers identify priorities for improvement.

Existing benchmarks address selected aspects of TCS research (Table 1). We introduce **TCSExam**, a unified framework and benchmark covering the full range of TCS research (Figure 1). Its four tracks evaluate theorem proving, solver engineering, constructions, and open-conjecture settlement, with success criteria defined by the requested outputs. Research areas can span multiple tracks, and shared task interfaces and verification protocols support the incorporation of additional problems.

TCSExam comprises 100 tasks: 20 research theorems to prove in Lean 4 (de Moura & Ullrich, 2021), 30 solver-engineering problems evaluated over instance families, 20 construction problems scored against documented records, and 30 open conjectures in prove-or-refute form. Authors with expertise in the relevant TCS areas reviewed all 100 tasks for research significance. Every submission is checked by the Lean kernel or an executable task checker. This combines expert judgment about which problems matter with reproducible verification of the submitted results.

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.

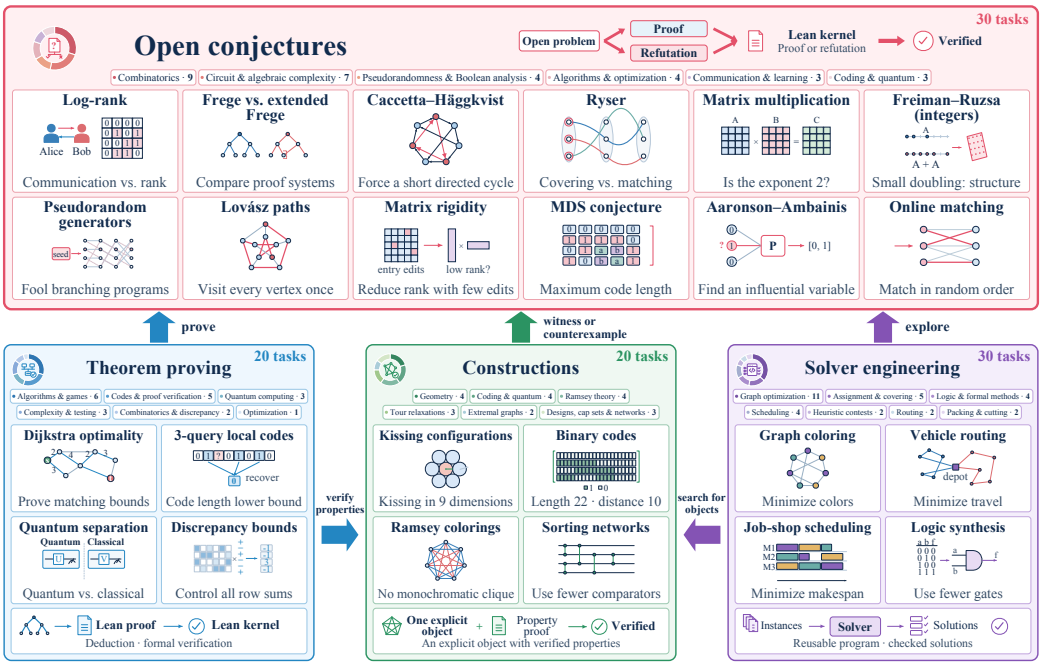


Figure 1: TCSExam covers the full range of TCS research through four evaluation tracks. Arrows show how proving, searching, and constructing can contribute to one another and to resolving open conjectures. The examples illustrate tasks within each track; research areas can span multiple tracks.

We evaluate six frontier LLMs on TCSExam. On 20 research theorems without supplied intermediate lemmas, GPT-6-Astra achieves a per-attempt complete-proof success rate of 70%, compared with at most 11.7% for each of the other five models. Solver performance varies across instance families, with submitted programs improving on public competition circuits. DeepSeek-Flash produces a verified construction that exceeds a task reference despite completing none of its 60 theorem-proving attempts. None of the six models fully proves or refutes any of the 30 open conjectures under a budget of up to 12 hours per attempt. Individual attempts also yield independently verified special cases and bounds (Section 5.5).

Our contributions are as follows.

- **A framework spanning the full range of TCS research.** Four tracks evaluate research outputs and open-conjecture settlement through shared task interfaces, distinct success criteria, and a common protocol for incorporating additional problems.
- **An expert-reviewed benchmark with verifiable outcomes.** We curate 100 research tasks with track-specific scoring and verification by the Lean kernel or executable task checkers.
- **A systematic evaluation of six frontier LLMs.** We characterize performance across the four tracks and examine submitted proofs, programs, and constructions to identify research accomplishments and remaining limitations.

2 Related Work

Theorem proving in mathematics and TCS. TCS-BENCH (Cohen-Addad et al., 2026) evaluates research-level natural-language proofs using an LLM verifier. Despite reported accuracy above 90% against expert judgments, acceptance remains subject to verifier errors and provides no formal correctness guarantee. FormalTCS (Wang et al., 2026) evaluates end-to-end formalization and proving of TCS results. Their benchmark scope does not include reusable solver performance or construction quality. Formal evaluation also spans competition mathematics, undergraduate mathematics, and repository-context proofs (Zheng et al., 2021; Tsoukalas et al., 2024; Azerbayev et al., 2023;

Table 1: Coverage of selected benchmarks. Task counts exclude internal instances.

Benchmark / track	# Tasks	Theorem proving	Solver engineering	Constructions	Open conjectures
TCS-BENCH	300	✓	✗	✗	✗
FormalTCS	143	✓	✗	✗	✗
FrontierCS (1.0)	240	✗	✓	✓	✗
EdgeBench-Formal	13	✓	✗	✗	○
EdgeBench-Optimization	19	✗	✓	○	✗
TCSExam	**100**	✓	✓	✓	✓

✓ evaluated; ✗ outside scope; ○ related tasks with a different scope.
○: machine-specific halting proofs (Formal); graph-coloring optimization (Optimization).

Yang et al., 2023; Hu et al., 2024). CombiBench provides formal combinatorial problems in Lean 4 (Liu et al., 2025), while LeanGeo-Bench evaluates competition geometry in the LeanGeo framework (Song et al., 2025). Kimina-Prover uses reinforcement learning to train structured reasoning for Lean 4 proof generation (Wang et al., 2025). Proof guidance and decomposition are studied by Draft, Sketch, and Prove, DeepSeek-Prover-V2, and Seed-Prover (Jiang et al., 2022; Ren et al., 2025; Chen et al., 2025).

Solver programs and explicit constructions. AlphaTensor and AlphaDev demonstrate algorithm discovery through reinforcement learning (Fawzi et al., 2022; Mankowitz et al., 2023); FunSearch and AlphaEvolve use program search for mathematical and computational improvements (Romera-Paredes et al., 2024; Novikov et al., 2025). FrontierCS evaluates executable programs on open-ended algorithmic and research tasks (Mang et al., 2026; Frontier-CS Team, 2026), with mathematical constructions also present in its 2.0 collection (FrontierCS Contributors, 2026). EdgeBench includes formal proving and combinatorial optimization in long-interaction evaluation (Zhu et al., 2026). TCSExam distinguishes reusable solver performance across instance families from the parameters attained by explicit constructions, with separate submission interfaces and success criteria.

Open conjectures and research-level evaluation. FrontierMath evaluates advanced mathematical reasoning through automatically checkable answers (Glazer et al., 2024). Formal Conjectures supplies formalized research problems and frozen evaluations of verified discovery (Firsching et al., 2026); the proving-ground repository reports complete solutions and verified reductions on its open tier (phyter1, 2026). TCSExam evaluates complete proof or refutation of open TCS conjectures, retaining intermediate outputs for separate analysis.

Positioning of TCSExam. Research-agent benchmarks also evaluate ML engineering and research replication (Chan et al., 2024; Wijk et al., 2024; Starace et al., 2025). Table 1 compares the closest TCS benchmarks. TCSExam brings four forms of TCS research output into one extensible framework, with distinct success criteria and common procedures for task development, verification, and version control.

3 TCSExam

3.1 Research scope and task interfaces

TCSExam organizes tasks across TCS areas by their required outputs and success criteria into the four tracks in Figure 1. Search can produce candidate objects, proofs establish their properties, and both can contribute to resolving open conjectures. Each track also has independent value: a reusable solver or an improved construction can advance TCS without settling a conjecture. Appendix A lists the 100 tasks and their sources.

Theorem proving (20 tasks). Given a formal statement derived from a research theorem, the model submits a Lean 4 proof. We vary the supplied context through three settings with a common target: *bare* provides the main statement, fixed definitions, and mathematical assumptions; *ladder* additionally provides intermediate lemma statements; *guided* further supplies proof ideas. This design is related to TCS-BENCH's use of intermediate-result masking to vary proof-task difficulty (Cohen-

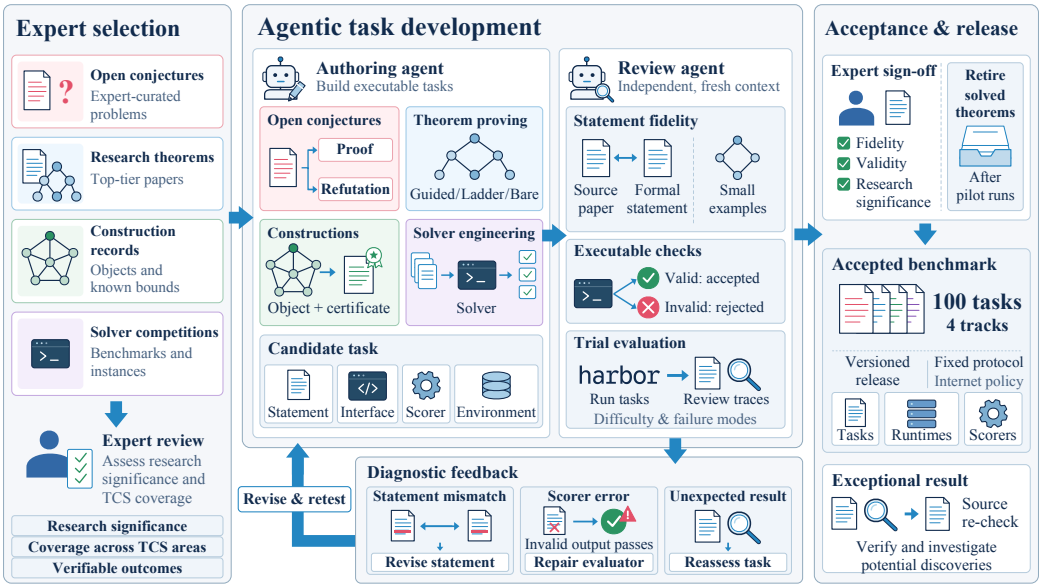


Figure 2: Task curation and validation. Expert selection is followed by agent-assisted authoring and review, executable checks, and trial evaluation. Statement mismatches, scorer errors, and unexpected results prompt revision and re-testing. Tasks are included after expert approval; exceptional results trigger verification and source re-checks.

Addad et al., 2026). Our main evaluation uses bare to measure complete-proof success. Ladder and guided support a separate analysis of proof guidance, with partial checkpoint credit and independent complete-proof checks (Section 3.3).

Solver engineering (30 tasks). The model receives a problem specification, a naive baseline, and public instances, then submits a reusable solver program. Final judging evaluates the program across the task's designated instance set under task-specific resource limits. Checkers validate solutions or certificates and recompute objective values.

Constructions (20 tasks). The model submits an explicit mathematical object and a certificate of its required properties. Examples include codes, geometric configurations, and graph colorings. Tasks use Lean 4 formalizations and executable checks to verify the object and its achieved parameter. Documented construction records provide reference values against which progress is measured; the model may use mathematical reasoning, search code, or both to produce its submission.

Open conjectures (30 tasks). The model receives a formalized conjecture and two Lean proof slots, `conj` and `refuted`, for the statement and its negation. A submission must prove either side in full. A counterexample must come with a proof that it refutes the stated conjecture. This interface accepts either direction of a resolution.

3.2 Task curation and development

We select tasks for their **research significance**, **coverage across TCS areas**, and **verifiable outcomes**. Theorem tasks come from recent TCS papers (2023–2026) and span 13 subfields. Solver tasks draw on established competitions and instance libraries. Task-specific safeguards, including withholding hidden instances from the agent during development and applying structure-preserving transformations, mitigate direct reuse of stored answers. Construction tasks target open problems with documented records, while the open-conjecture track includes problems such as log-rank and Ryser. Expert review assesses the significance and representativeness of the selected problems.

Figure 2 summarizes the task development process. An authoring agent prepares the statement, submission interface, scorer, and environment. A separate review agent checks correspondence to the research source and the behavior of the verification procedure. Tests and pilot evaluations

identify defects and guide revision. Across development rounds, over 40 theorem problems were retired after being solved in pilot evaluations. Tasks are included after expert approval.

3.3 TRACK-SPECIFIC SCORING

Scores reflect the submitted output and the task's notion of success. Complete proof obligations use binary scores; optimization and construction tasks provide graded credit. The score definitions and integrity checks are specified per track (Appendix B).

Theorem proving. A valid proof of the complete bare-level statement scores 1; an incomplete proof scores 0. Guided and ladder tasks receive partial credit for the fraction of checkpoints passed; a verified complete proof receives full credit. Each checkpoint is checked in isolation with the other checkpoint statements supplied as assumptions; this also applies to the main-theorem checkpoint. Coverage therefore measures conditional proof obligations. Complete proofs are checked separately without unproved checkpoint assumptions (Appendix B.1).

Solver engineering. Optimization tasks measure the fraction of the difference between a naive baseline B and a task reference K recovered by the submitted solver. These scores are not clipped: values below 0 indicate performance worse than the baseline, and values above 1 indicate improvement over the reference. Decision tasks use solve rates. A separate Kimi-K3-relative score standardizes raw instance outcomes against repeated Kimi-K3 runs and applies a Gaussian cumulative-distribution transform (Appendix B.2). Scores are averaged over instances, then attempts within tasks, then tasks. Improvements over K undergo artifact and source verification before being described as new records.

Constructions. Scores measure attainment relative to a trivial construction and a documented record: matching the record scores 1, and surpassing it can score above 1. Task-specific shaping is cubic for 14 tasks and linear for the remainder. We report the achieved mathematical parameters alongside these scores, since the score transformation changes the numerical size of an improvement. A record provides a comparison point even when the optimum remains unknown.

Open conjectures. A verified proof of either the statement or its negation scores 1. All other submissions score 0 on the resolution criterion. Intermediate bounds, special cases, and counterexamples to related statements can still be useful research outputs; they are retained for separate analysis. Consequently, a zero on this track means that the stated conjecture was not fully resolved.

We report a performance profile across the four tracks without a composite score. This preserves the distinction between completing a proof, improving performance over an instance family, strengthening a construction bound, and settling an open conjecture.

3.4 TASK AND RESULT VALIDATION

Before a task enters the benchmark, task validation checks whether the encoded problem matches its research source. After submission, result validation checks outputs against the task rules, with additional review for candidate advances. Appendix C documents these checks and supporting evidence.

Formal statement fidelity. Agent-assisted reviews compare definitions, assumptions, quantifiers, and parameter ranges in the Lean statement with the source. Small examples and boundary cases test for degenerate or unintended interpretations, and re-translation provides an additional cross-check. Shared definitions and statements are checked across disclosure levels. Successful proofs and unexpected outcomes prompt renewed scrutiny of the statement and its assumptions. These checks also flagged potential inconsistencies in source statements or proofs; candidate tasks with unresolved concerns were excluded.

Result validation. Every submission is checked by the Lean kernel or an executable task checker. Candidate advances additionally undergo agent-assisted audits of the rollouts and submitted artifacts. These audits check compliance with task rules, proof dependencies or object validity, and the conditions of the claimed improvement. For solver results, this includes comparing instance sets and execution budgets. Source checks distinguish improvements over task references from new literature records; claims of new records require evidence for the prior best value and a matching problem setting.

Task versions and extensions. Each reported result is associated with the task statement, environment, verifier, and reference values used in evaluation. Corrections and additional problems enter new task versions, with changes and affected results recorded. Earlier versions and evaluation records are retained so that results remain interpretable as the benchmark expands.

4 Experimental Setup

Models and execution. We evaluate GPT-6-Astra (OpenAI, 2026), Kimi-K3 (Kimi Team, 2026), GLM-5.3 (Z.ai, 2026), Qwen3.8-max (Alibaba Cloud, 2026), Grok-4.6 (xAI, 2026), and DeepSeek-Flash (DeepSeek-AI, 2026). Qwen3.8-max denotes snapshot 0902, and DeepSeek-Flash denotes v4.1 Flash. Comparisons characterize model–harness systems. Appendix D documents harness versions, reasoning settings, attempt counts, and recorded token usage.

Tasks and resources. Each model is evaluated on 100 tasks: 20 bare-level theorems, 30 solver problems, 20 constructions, and 30 open conjectures. An attempt is one model run on one task. GPT-6-Astra, GLM-5.3, Qwen3.8-max, and Grok-4.6 each receive one attempt per task; Kimi-K3 and DeepSeek-Flash contribute three. Each attempt has a 12-hour wall-clock budget. Theorem and solver tasks are offline; construction and conjecture tasks allow literature access to support reuse and extension of published results. Tasks use the Harbor format (Harbor Framework Team, 2026) to package environments, submission interfaces, and verifiers. Public instances support solver development; final judging uses a fixed evaluation set under a separate per-instance or shared-pool budget. Appendix D.2 explains source adaptation, instance splits, public overlap in two tasks, and execution limits. Construction prompts disclose published reference bounds but withhold the numerical scoring formula for all six models (Appendix D.3).

Score aggregation. Each attempt receives one task score after combining its instance or parameter-setting results according to the task's scoring rule. For Kimi-K3's solver evaluation, for example, we average its three scores on each problem, then average the 30 problem means. Every problem has equal weight, regardless of its number of instances. The same procedure applies to the other tracks; models evaluated once contribute one score per task. For bare theorem proving, this equals complete-proof successes divided by all 20 or 60 scheduled attempts. These means describe single-attempt performance; success in any of several attempts is reported separately. Solver results include both the task score and the Kimi-K3-relative score (Appendix B.2).

Proof-guidance comparison. We compare Kimi-K3 and DeepSeek-Flash on the same 20 theorems, with three attempts per theorem in each setting. Bare supplies the target statement and definitions; ladder adds intermediate lemma statements; guided also supplies proof ideas. Complete-proof success requires a verified assembled or direct proof without unproved checkpoint assumptions. We report success per attempt and the number of theorems solved at least once; ladder and guided also receive partial checkpoint credit (Appendix D.3).

5 Results

5.1 Overall performance

Table 2 summarizes performance across the four tracks. Figure 3 shows task-level results for theorem proving, solver engineering, and constructions.

Research proofs distinguish the evaluated model–harness systems. GPT-6-Astra completes 14 of 20 theorems (70%) without supplied intermediate lemmas. Kimi-K3 succeeds in 7 of 60 attempts (11.67%), covering three distinct theorems. GLM-5.3, Qwen3.8-max, and Grok-4.6 each succeed in 1 of 20 attempts (5%); DeepSeek-Flash succeeds in none of 60. All successes by other models occur among Astra's 14 solved theorems; six theorems remain unproved across all recorded attempts. All scheduled attempts remain in the denominator, with success determined by the proof checker.

Construction means reflect reference attainment across tasks. Astra has the highest mean at 1.0495, matching the reference on 17 tasks and exceeding it on three. The other systems range from 0.8620 to 0.9769. GLM-5.3 ranks second by mean score without exceeding any reference, illustrating that broad reference attainment and individual improvements are distinct aspects of performance. Section 5.4 examines the verified improvements on specific tasks.

Table 2: **Experimental results.** Higher is better (↑) for all four metrics. Kimi-K3 and DeepSeek-Flash have three attempts per task; others have one. Tasks receive equal weight. Solver scores use the Kimi-K3-relative metric (Appendix B.2). Theorem outcomes use recorded checker scores.

Model	**Theorem proving** Success / attempts (%) ↑	**Solver** Relative (%) ↑	**Constructions** Task score ↑	**Open conjectures** Resolved / 30 ↑
GPT-6-Astra	14/20 (70.0)	64.64	1.0495	0/30
Kimi-K3	7/60 (11.7)	48.73	0.9555	0/30
GLM-5.3	1/20 (5.0)	48.97	0.9769	0/30
Qwen3.8-max	1/20 (5.0)	49.19	0.9508	0/30
Grok-4.6	1/20 (5.0)	43.72	0.8620	0/30
DeepSeek-Flash	0/60 (0.0)	48.20	0.8658	0/30

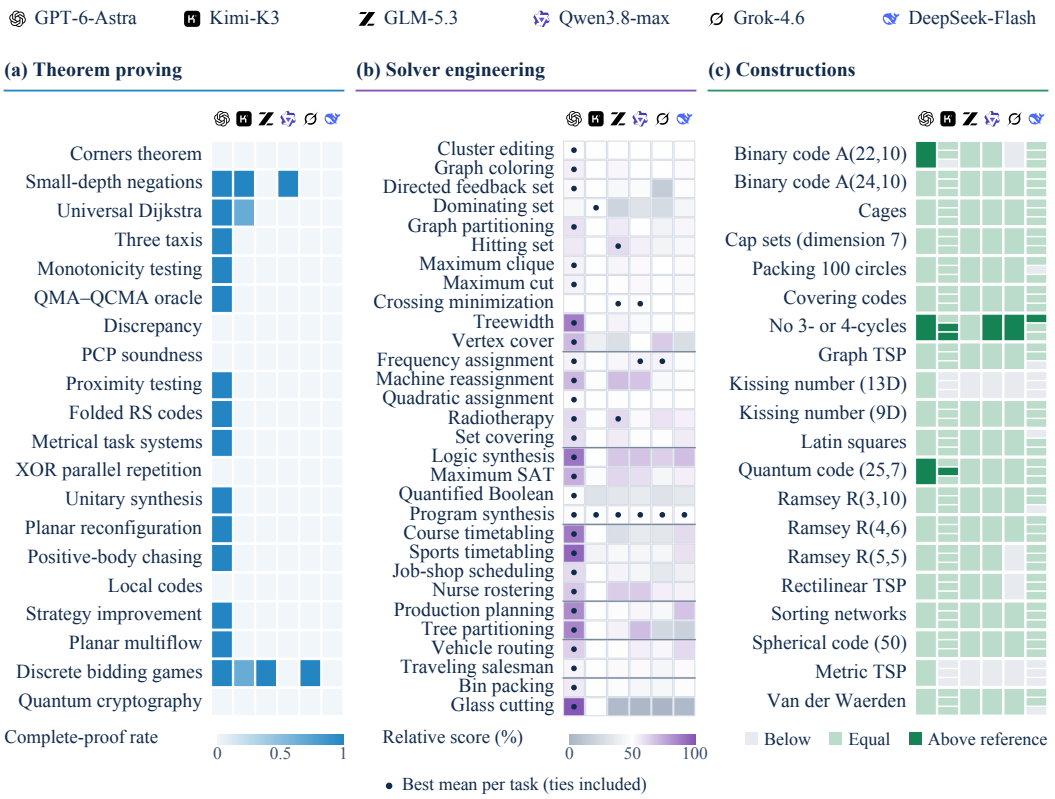


Figure 3: **Task-level results.** Rows are tasks and columns are models. (a) Complete-proof success rates. (b) Solver task means on the Kimi-K3-relative scale; dots mark the highest mean, including ties. (c) Each cell shows one or three vertically stacked construction attempts, colored below, at, or above the task reference score of 1 (tolerance 10^{-6}). Full scores appear in Appendix E.

5.2 Solver performance across instance families

Performance differs across problem families. On each instance, a relative score of 50 denotes a valid outcome equal to the Kimi-K3 reference mean. Astra averages 64.64, compared with 43.72 for Grok-4.6 and 48.20–49.19 for the other four models (Table 2). Astra has the highest unrounded task mean, including ties, on 27 of 30 problems (Figure 3b). It trails GLM-5.3 on hitting set, Kimi-K3 on dominating set, and GLM-5.3/Qwen3.8-max marginally on crossing minimization. Appendix E.2 gives detailed results.

Relative scoring limits baseline–reference scaling effects. The original score $(B - A)/(B - K)$ compares objective A with baseline B and reference K. A large $|B - K|$ compresses score

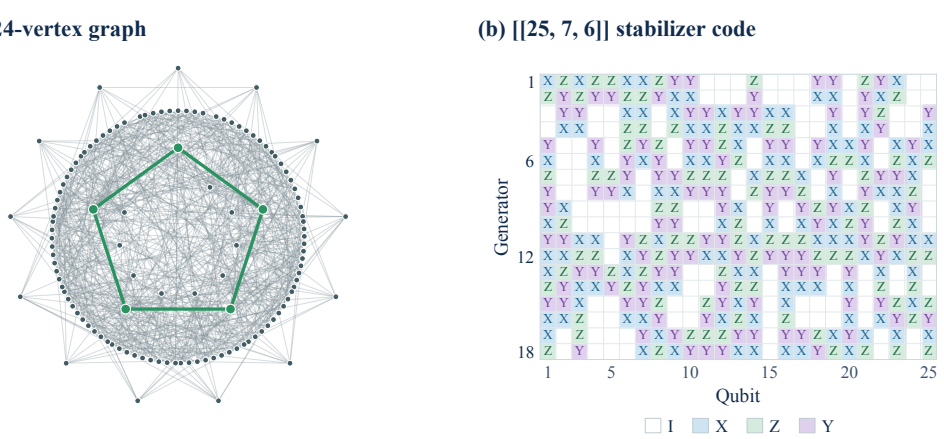


Figure 4: **Two constructions submitted by Kimi-K3.** (a) All 124 vertices and 630 edges of a graph with no 3- or 4-cycles, exceeding the 629-edge task reference. Green highlights a shortest cycle. (b) The 18 stabilizer generators of a $[[25, 7, 6]]$ quantum code, improving the reference distance from 5 to 6. Each row specifies an operator on 25 qubits; white cells denote the identity I. Both panels are drawn directly from submitted certificates.

differences near K; a small denominator can amplify individual contributions. We introduce a Kimi-K3-relative metric that compares raw outcomes with repeated Kimi-K3 results, retains sensitivity near their mean, and bounds contributions to $[0, 100]$ on the reported scale. For example, Qwen's valid IWLS 2023 circuit for `ex37` uses 273,020 AND gates, versus baseline 698 and reference 106. The ratio $(698 - 273{,}020)/(698 - 106) \approx -460$ allows this single oversized circuit to depress the original mean. Qwen averages 0.6473 on that scale and 49.19 on the bounded relative scale. Appendix B.2 defines the metric; Appendix E.2 gives raw outcomes.

Solver improvements across a benchmark family. On the IWLS 2023 logic-synthesis task, Astra's submitted program produces valid circuits on all 100 evaluation instances: it improves the task reference on 22, matches it on 31, and uses more AND gates than the reference on 47. Each of the other models' submitted programs also passes all 100 validity checks, but only one Kimi-K3 attempt and Grok-4.6 improve a reference, each on one instance. These counts describe the same program applied across a family of inputs. Appendix F.1 gives the circuit-level improvements and reference sources.

5.3 Understanding proof performance

More theorems are solved with additional statements and proof ideas. With three attempts on each of 20 theorems, Kimi-K3 solves 3, 4, and 7 theorems at least once under bare, ladder, and guided, respectively; DeepSeek-Flash solves 0, 2, and 6. Guided also has the highest per-attempt complete-proof success: Kimi-K3 succeeds in 18/60 attempts, compared with 7/60 under both ladder and bare. DeepSeek-Flash succeeds in 0/60 bare, 2/60 ladder, and 11/60 guided attempts. Appendix E.3 reports complete-proof outcomes, partial-credit scores, and individual proof attempts.

5.4 Verified construction improvements

Graph and code results. In the Erdős girth-5 extremal-graph task, Kimi-K3 constructs a 124-vertex, 630-edge graph without 3- or 4-cycles, exceeding the 629-edge reference (Goedgebeur et al., 2025). In the quantum stabilizer-code task, it constructs a binary $[[25, 7, 6]]$ code, increasing the minimum distance from 5 to 6 for a code on 25 qubits encoding seven logical qubits (Grassl, n.d.) (Figure 4). Astra also attains both parameters and proves that a binary code of length 22 and minimum Hamming distance 10 has at most 82 codewords ($A(22, 10) \leq 82$), improving the published upper bound of 84 (Gijswijt et al., 2012) (Appendix D.3). Qwen3.8-max, Grok-4.6, and DeepSeek-Flash each obtain a 630-edge graph. Appendix F records provenance and shows Astra's constructions.

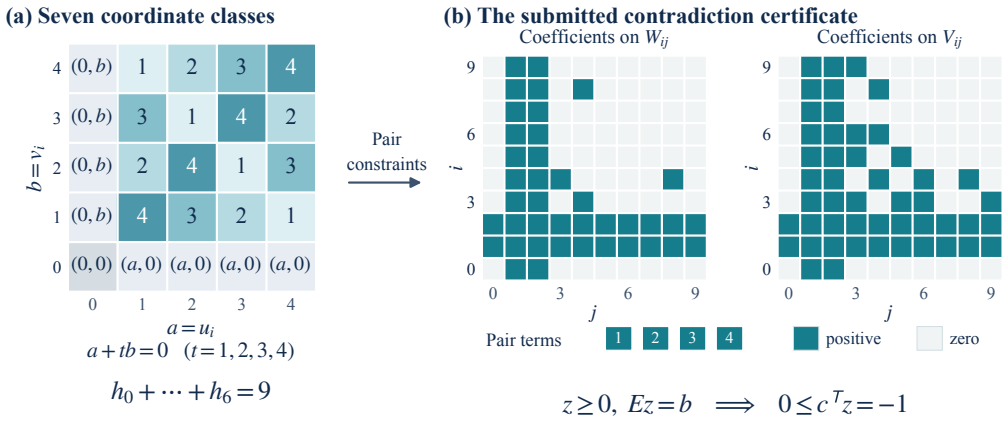


Figure 5: **Astra's proof certificate for the nine-dimensional Alon–Jaeger–Tarsi case over $\mathbb{F}_5$.** (a) Coordinate pairs fall into seven classes; numbered cells give the unique nonzero t with $a + tb = 0$. Class counts sum to nine and yield pair constraints. (b) The actual certificate has nonnegative coefficients on two arrays of weight counts and four pair terms. Combined with the counting identities, it forces a nonnegative sum to equal -1 after positive rescaling, excluding a minimal counterexample. Colored cells mark positive coefficients; pale cells mark zeros.

5.5 Open conjectures within the evaluation budget

No complete resolution within the budget. Under a budget of up to 12 hours per attempt, none of the six evaluated systems fully proves or refutes any of the 30 open conjectures. Recorded resource use is reported in Appendix D.5.

Verified intermediate results and potential advances. While attempting the full conjecture, GPT-6-Astra proves the Alon–Jaeger–Tarsi conjecture over $\mathbb{F}_5$ for every $n \leq 9$: every invertible $n \times n$ matrix A admits a vector x such that x and Ax have no zero coordinates. The $n = 9$ case extends the dimensional ranges $n \leq 7$ of Yu (1999) and $n \leq 8$ reported in a recent preprint (Zeng et al., 2026). The proof combines counting identities with an exact integer certificate to exclude a minimal counterexample (Figure 5). Independent Lean verification confirms the result using only standard axioms. Appendix F.3 gives the proof structure, formal verification, and its relation to prior work. The unrestricted conjecture remains unresolved. Four further verified cases concern a six-variable Fourier entropy–influence bound, an exact high-rank Hilbert-matrix rigidity formula, an influence bound for quadratic polynomials, and finite-instance online-matching bounds (Appendix F.4). Together, these five cases suggest that pursuing open conjectures can yield useful intermediate TCS results and potential research advances even without a complete resolution.

6 Conclusion

TCSExam provides a unified framework for evaluating LLMs across the full range of TCS research, connecting theorem proving, solver engineering, and explicit constructions with open-conjecture settlement. The six-model evaluation reveals differences in proof completion and solver performance, alongside verified solver and construction improvements over selected references. For both models tested across proof settings, supplying intermediate lemmas and proof ideas increases the number of theorems solved. All 30 open conjectures remain unresolved within the evaluation budget. Verified special cases and bounds suggest that pursuing these conjectures can also generate potential advances in TCS and mathematics. Extensible tasks, versioned verification, and research-output registration support continued evaluation (Appendix G).

AI USE STATEMENT

AI coding assistants were used in task authoring, formalization, checker implementation, and iterative review (Section 3.4). AI assistants also supported data processing, result analysis, manuscript drafting and editing, figure code, and literature search. Evaluated model outputs are identified as experimental submissions and checked under the relevant task protocol. Result tables and plots are generated programmatically from experiment records. The authors take responsibility for the final content of this work.

ETHICS STATEMENT

The benchmark uses mathematical problems and computational instance collections. Authors with relevant TCS expertise contributed to task development and selection. The experiments evaluate model-generated submissions. Mathematical sources and instance collections are identified in Appendix A. Artifact verification and source comparison are required before reporting a research advance (Appendix F.6). The evaluation records task versions and scoring rules for every system, with version differences disclosed alongside results. Evaluation entails substantial computation; Appendix D records the development budgets and separates them from task-specific judging resources.

REPRODUCIBILITY STATEMENT

The task format packages the environment, submission interface, and verifier in version-controlled definitions. Appendix B specifies the scoring quantities, Appendix D describes model and harness settings and score aggregation, and Appendix E organizes the complete results. If the paper is accepted, we will release the task manifest and shareable artifacts. Proofs require the pinned Lean toolchain and libraries; solver comparisons additionally require the documented judging resources. Hosted model APIs and external literature access can change over time.

REFERENCES

Alibaba Cloud. Qwen3.8-max: Model documentation and snapshot versions, 2026. URL https://www.alibabacloud.com/help/en/model-studio/qwen3-8-max. Snapshot qwen3.8-max-0902 (2026-09-02); accessed September 23, 2026.

Guy Avni and Suman Sadhukhan. Mean-Payoff and Energy Discrete-Bidding Games. In *34th EACSL Annual Conference on Computer Science Logic (CSL 2026)*, pp. 32:1–32:18, 2026. doi: 10.4230/LIPIcs.CSL.2026.32. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CSL.2026.32.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. ProofNet: Autoformalizing and Formally Proving Undergraduate-Level Mathematics, 2023. URL https://arxiv.org/abs/2302.12433. arXiv:2302.12433.

Nikhil Bansal and Haotian Jiang. Decoupling via Affine Spectral-Independence: Beck-Fiala and Komlós Bounds Beyond Banaszczyk, 2025. URL https://arxiv.org/abs/2508.03961. arXiv:2508.03961.

J. E. Beasley. OR-Library: Distributing Test Problems by Electronic Mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990. doi: 10.1057/jors.1990.166. URL https://doi.org/10.1057/jors.1990.166.

Amey Bhangale, Mark Braverman, Subhash Khot, Yang P. Liu, and Dor Minzer. Parallel Repetition for 3-Player XOR Games, 2024. URL https://arxiv.org/abs/2408.09352. arXiv:2408.09352.

Sayan Bhattacharya, Niv Buchbinder, Roie Levin, and Thatchaphol Saranurak. Chasing Positive Bodies. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 1694–1714, 2023. doi: 10.1109/focs57990.2023.00103. URL https://doi.org/10.1109/focs57990.2023.00103.

John Bostanci, Jonas Haferkamp, Chinmay Nirkhe, and Mark Zhandry. Separating QMA from QCMA with a Classical Oracle. In *Proceedings of the 58th Annual ACM Symposium on Theory of Computing*, pp. 595–606, 2026. doi: 10.1145/3798129.3800776. URL https://doi.org/10.1145/3798129.3800776.

Nicolas Bousquet and Daniel W. Cranston. A Linear Kernel for Independent Set Reconfiguration in Planar Graphs. In *43rd International Symposium on Theoretical Aspects of Computer Science (STACS 2026)*, pp. 19:1–19:18, 2026. doi: 10.4230/LIPIcs.STACS.2026.19. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.STACS.2026.19.

Andries E. Brouwer. Table of general binary codes. Online table, n.d. URL https://aeb.win.tue.nl/codes/binary-1.html. Accessed September 26, 2026.

Sébastien Bubeck, Christian Coester, and Yuval Rabani. The Randomized k-Server Conjecture Is False! In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pp. 581–594, 2023. doi: 10.1145/3564246.3585132. URL https://doi.org/10.1145/3564246.3585132.

Rainer E. Burkard, Stefan E. Karisch, and Franz Rendl. QAPLIB – A Quadratic Assignment Problem Library. *Journal of Global Optimization*, 10(4):391–403, 1997. doi: 10.1023/a:1008293323270. URL https://doi.org/10.1023/a:1008293323270.

Bruno Cavalar, Théo Borém Fabris, Partha Mukhopadhyay, Srikanth Srinivasan, and Amir Yehudayoff. Negations Are Powerful Even in Small Depth. In *Proceedings of the 58th Annual ACM Symposium on Theory of Computing*, pp. 2071–2082, 2026. doi: 10.1145/3798129.3800911. URL https://doi.org/10.1145/3798129.3800911.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. MLE-bench: Evaluating Machine Learning Agents on Machine Learning Engineering, 2024. URL https://arxiv.org/abs/2410.07095. arXiv:2410.07095.

Luoxin Chen, Jinming Gu, Liankai Huang, Wenhao Huang, Zhicheng Jiang, Allan Jie, et al. Seed-Prover: Deep and Broad Reasoning for Automated Theorem Proving, 2025. URL https://arxiv.org/abs/2507.23726. arXiv:2507.23726.

Mark Chen, Xi Chen, Hao Cui, William Pires, and Jonah Stockwell. Boolean function monotonicity testing requires (almost) $n^{1/2}$ queries. In *Proceedings of the 58th Annual ACM Symposium on Theory of Computing*, pp. 584–594, 2026. doi: 10.1145/3798129.3800775. URL https://doi.org/10.1145/3798129.3800775.

Yeyuan Chen and Zihan Zhang. Explicit Folded Reed-Solomon and Multiplicity Codes Achieve Relaxed Generalized Singleton Bounds. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, pp. 1–12, 2025. doi: 10.1145/3717823.3718114. URL https://doi.org/10.1145/3717823.3718114.

Christian Coester and Tze-Yang Poon. Online 3-Taxi on General Metrics. In *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 6659–6673, 2026. doi: 10.1137/1.9781611978971.238. URL https://doi.org/10.1137/1.9781611978971.238.

Vincent Cohen-Addad, Dimitris Paparas, Ernest van Wijland, Max Springer, Julien Canitrot-Paradis, Honghao Lin, et al. TCS-BENCH: Benchmarking State-of-the-Art Generative AI Theoretical Computer Science Research Ability, 2026. URL https://arxiv.org/abs/2608.09538. arXiv:2608.09538.

Leonardo de Moura and Sebastian Ullrich. The Lean 4 Theorem Prover and Programming Language. In *Automated Deduction – CADE 28*, pp. 625–635, 2021. doi: 10.1007/978-3-030-79876-5_37. URL https://doi.org/10.1007/978-3-030-79876-5_37.

DeepSeek-AI. DeepSeek-V4.1-Flash: Pushing the Limits of KV Cache Compression, 2026. URL https://arxiv.org/abs/2609.19969. arXiv:2609.19969.

Iain Dunning, Swati Gupta, and John Silberholz. What Works Best When? A Systematic Evaluation of Heuristics for Max-Cut and QUBO. *INFORMS Journal on Computing*, 30(3):608–624, 2018. doi: 10.1287/ijoc.2017.0798. URL https://doi.org/10.1287/ijoc.2017.0798.

Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022. doi: 10.1038/s41586-022-05172-4. URL https://doi.org/10.1038/s41586-022-05172-4.

Moritz Firsching, Paul Lezeau, Salvatore Mercuri, Miklós Z. Horváth, Yaël Dillies, Calle Sönne, Eric Wieser, Fred Zhang, Thomas Hubert, Blaise Agüera y Arcas, and Pushmeet Kohli. Formal Conjectures: An Open and Evolving Benchmark for Verified Discovery in Mathematics, 2026. URL https://arxiv.org/abs/2605.13171. arXiv:2605.13171.

Frontier-CS Team. Frontier-CS 1.0 Release, 2026. URL https://frontier-cs.org/blog/feb-release/. Release announcement; accessed September 23, 2026.

FrontierCS Contributors. Frontier-CS 2.0 Problems, 2026. URL https://github.com/FrontierCS/Frontier-CS/tree/main/2.0. Dataset repository; accessed September 23, 2026.

Dion Gijswijt, Hans D. Mittelmann, and Alexander Schrijver. Semidefinite code bounds based on quadruple distances. *IEEE Transactions on Information Theory*, 58(5):2697–2705, 2012. URL https://ir.cwi.nl/pub/20724.

Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, et al. FrontierMath: A Benchmark for Evaluating Advanced Mathematical Reasoning in AI, 2024. URL https://arxiv.org/abs/2411.04872. arXiv:2411.04872.

Jan Goedgebeur, Jorik Jooken, Gwenaël Joret, and Tibo Van den Eede. Improved lower bounds on the maximum size of graphs with girth 5, 2025. URL https://arxiv.org/abs/2508.05562v1. arXiv:2508.05562v1; Table 1.

Rohan Goyal and Venkatesan Guruswami. Optimal Proximity Gaps for Subspace-Design Codes and (Random) Reed-Solomon Codes. Technical Report TR25-166, Electronic Colloquium on Computational Complexity, 2025. URL https://eccc.weizmann.ac.il/report/2025/166/. Revision 1, March 24, 2026.

Markus Grassl. Bounds on the minimum distance of quantum codes: $[[25, 7]]_2$, n.d. URL https://www.codetables.de/QECC/QECC.php?q=4&n=25&k=7. Online code table; accessed September 23, 2026.

Bernhard Haeupler, Richard Hladík, Václav Rozhoň, Robert E. Tarjan, and Jakub Tetěk. Universal Optimality of Dijkstra Via Beyond-Worst-Case Heaps. In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 2099–2130, 2024. doi: 10.1109/focs61266.2024.00125. URL https://doi.org/10.1109/focs61266.2024.00125.

Brandon Hanson and Giorgis Petridis. Refined estimates concerning sumsets contained in the roots of unity. *Proceedings of the London Mathematical Society*, 122(3):353–358, 2021. doi: 10.1112/plms.12322. URL https://doi.org/10.1112/plms.12322.

Harbor Framework Team. Harbor: A framework for evaluating and optimizing agents and models in container environments, 2026. URL https://github.com/harbor-framework/harbor. Software repository; accessed September 23, 2026.

Jiewen Hu, Thomas Zhu, and Sean Welleck. miniCTX: Neural Theorem Proving with (Long-)Contexts, 2024. URL https://arxiv.org/abs/2408.03350. arXiv:2408.03350.

IWLS 2023 Contest Organizers. IWLS 2023 Programming Contest: Logic Synthesis, 2023. URL https://github.com/alanminko/iwls2023-ls-contest. Official contest repository and results presentation; accessed September 23, 2026.

Michael Jaber, Yang P. Liu, Shachar Lovett, Anthony Ostuni, and Mehtaab Sawhney. Quasipolynomial Bounds for the Corners Theorem. In *2025 IEEE 66th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 2027–2036, 2025. doi: 10.1109/focs63196.2025.00108. URL https://doi.org/10.1109/focs63196.2025.00108.

Albert Q. Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Yuhuai Wu, and Guillaume Lample. Draft, Sketch, and Prove: Guiding Formal Theorem Provers with Informal Proofs, 2022. URL https://arxiv.org/abs/2210.12283. arXiv:2210.12283.

Sina Kalantarzadeh and Nikhil Kumar. Improved Lower Bounds on Multiflow-Multicut Gaps. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2025)*, pp. 14:1–14:18, 2025. doi: 10.4230/LIPIcs.APPROX/RANDOM.2025.14. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.APPROX/RANDOM.2025.14.

Kimi Team. Kimi K3: Open Frontier Intelligence, 2026. URL https://arxiv.org/abs/2607.24653. arXiv:2607.24653.

Pravesh K. Kothari and Peter Manohar. An Exponential Lower Bound for Linear 3-Query Locally Correctable Codes. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pp. 776–787, 2024. doi: 10.1145/3618260.3649640. URL https://doi.org/10.1145/3618260.3649640.

William Kretschmer, Luowen Qian, Makrand Sinha, and Avishay Tal. Quantum Cryptography in Algorithmica. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pp. 1589–1602, 2023. doi: 10.1145/3564246.3585225. URL https://doi.org/10.1145/3564246.3585225.

Junqi Liu, Xiaohan Lin, Jonas Bayer, Yael Dillies, Weijie Jiang, Xiaodan Liang, et al. CombiBench: Benchmarking LLM Capability for Combinatorial Mathematics, 2025. URL https://arxiv.org/abs/2505.03171. arXiv:2505.03171.

Alex Lombardi, Fermi Ma, and John Wright. A One-Query Lower Bound for Unitary Synthesis and Breaking Quantum Cryptography. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pp. 979–990, 2024. doi: 10.1145/3618260.3649650. URL https://doi.org/10.1145/3618260.3649650.

Qiuyang Mang, Wenhao Chai, Zhifei Li, Huanzhi Mao, Shang Zhou, Alexander Du, et al. FrontierCS: Evolving Challenges for Evolving Intelligence. In *Proceedings of the International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2512.15699.

Daniel J. Mankowitz, Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, et al. Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, 618(7964):257–263, 2023. doi: 10.1038/s41586-023-06004-9. URL https://doi.org/10.1038/s41586-023-06004-9.

Dor Minzer and Kai Zhe Zheng. Near Optimal Alphabet-Soundness Tradeoff PCPs. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pp. 15–23, 2024. doi: 10.1145/3618260.3649606. URL https://doi.org/10.1145/3618260.3649606.

János Nagy and Péter Pál Pach. The Alon–Jaeger–Tarsi conjecture via group ring identities. *Journal of the European Mathematical Society*, 2025. doi: 10.4171/JEMS/1640. URL https://doi.org/10.4171/JEMS/1640. Published online; preprint arXiv:2107.03956.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, et al. AlphaEvolve: A coding agent for scientific and algorithmic discovery, 2025. URL https://arxiv.org/abs/2506.13131. arXiv:2506.13131.

OpenAI. GPT-6 Astra System Card, 2026. URL https://deploymentsafety.openai.com/gpt-6-astra. System card, September 3, 2026.

phyter1. proving-ground: A benchmark for LLMs on provably unsolved problems, 2026. URL https://github.com/phyter1/proving-ground. Software repository; accessed September 23, 2026.

Gerhard Reinelt. TSPLIB—A Traveling Salesman Problem Library. *ORSA Journal on Computing*, 3(4):376–384, 1991. doi: 10.1287/ijoc.3.4.376. URL https://doi.org/10.1287/ijoc.3.4.376.

Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, et al. DeepSeek-Prover-V2: Advancing Formal Mathematical Reasoning via Reinforcement Learning for Subgoal Decomposition, 2025. URL https://arxiv.org/abs/2504.21801. arXiv:2504.21801.

ROADEF/EURO Challenge Organizers. ROADEF/EURO Challenge 2012: Machine Reassignment, Final Results, 2012. URL https://roadef.org/challenge/2012/en/results.php. Official competition results; accessed September 23, 2026.

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

Chendong Song, Zihan Wang, Frederick Pu, Haiming Wang, Xiaohan Lin, Junqi Liu, Jia Li, and Zhengying Liu. LeanGeo: Formalizing Competitional Geometry problems in Lean, 2025. URL https://arxiv.org/abs/2508.14644. arXiv:2508.14644.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, et al. PaperBench: Evaluating AI's Ability to Replicate AI Research, 2025. URL https://arxiv.org/abs/2504.01848. arXiv:2504.01848.

The mathlib Community. The Lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pp. 367–381, 2020. doi: 10.1145/3372885.3373824. URL https://doi.org/10.1145/3372885.3373824.

George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amitayush Thakur, and Swarat Chaudhuri. PutnamBench: Evaluating Neural Theorem-Provers on the Putnam Mathematical Competition, 2024. URL https://arxiv.org/abs/2407.11214. arXiv:2407.11214.

Tom van Dijk, Georg Loho, and Matthew T. Maat. The Worst-Case Complexity of Symmetric Strategy Improvement. In *32nd EACSL Annual Conference on Computer Science Logic (CSL 2024)*, pp. 24:1–24:19, 2024. doi: 10.4230/LIPIcs.CSL.2024.24. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CSL.2024.24.

Dingzirui Wang, Xuanliang Zhang, Keyan Xu, Qingfu Zhu, and Wanxiang Che. FormalTCS: Benchmarking End-to-End Frontier Formal Theoretical Computer Science Research of Large Language Models, 2026. URL https://arxiv.org/abs/2608.20153. arXiv:2608.20153.

Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, et al. Kimina-Prover Preview: Towards Large Formal Reasoning Models with Reinforcement Learning, 2025. URL https://arxiv.org/abs/2504.11354. arXiv:2504.11354.

Hjalmar Wijk, Tao Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, et al. RE-Bench: Evaluating frontier AI R&D capabilities of language model agents against human experts, 2024. URL https://arxiv.org/abs/2411.15114. arXiv:2411.15114.

xAI. Introducing Grok 4.6, 2026. URL https://x.ai/news/grok-4-6. Official release, August 12, 2026.

Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. LeanDojo: Theorem Proving with Retrieval-Augmented Language Models, 2023. URL https://arxiv.org/abs/2306.15626. arXiv:2306.15626.

Yang Yu. The permanent rank of a matrix. *Journal of Combinatorial Theory, Series A*, 85(2): 237–242, 1999. doi: 10.1006/jcta.1998.2904. URL https://doi.org/10.1006/jcta.1998.2904.

Z.ai. GLM-5.3 Model Card, 2026. URL https://huggingface.co/zai-org/GLM-5.3. Official model card; accessed September 23, 2026.

Zijian Zeng, Houde Liu, and Kuru Ratnavelu. Closing the permanent-rank boundary for the Alon–Jaeger–Tarsi conjecture, 2026. URL https://www.preprints.org/manuscript/202608.1194/v1. Preprints.org, version 1, August 18; not peer reviewed.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. MiniF2F: a cross-system benchmark for formal Olympiad-level mathematics, 2021. URL https://arxiv.org/abs/2109.00110. arXiv:2109.00110.

Deyao Zhu, Xin Zhou, Shengling Qin, Xuekai Zhu, Hangliang Ding, Shu Zhong, et al. EdgeBench: Unveiling Scaling Laws of Learning from Real-World Environments, 2026. URL https://arxiv.org/abs/2607.05155. arXiv:2607.05155.

A TASK CATALOGUE AND EXAMPLES

The catalogue records the sources and research questions represented by the 100-task collection. Task counts refer to problem packages; solver instances and construction parameter settings are listed separately in the final evaluation manifest. The references identify the mathematical sources; the evaluation protocol is specified in Appendix D.

A.1 THEOREM PROVING: 20 THEOREMS

Theorem	Source paper	Steps	Domain
`corners-quasipoly`	*Quasipolynomial Bounds for the Corners Theorem* (Jaber et al., 2025)	8	additive combinatorics
`dijkstra-universal`	*Universal Optimality of Dijkstra via Beyond-Worst-Case Heaps* (Haeupler et al., 2024)	8	algorithms
`discrepancy-beyond-banaszczyk`	*Decoupling via Affine Spectral-Independence: Beck-Fiala and Komlós Bounds Beyond Banaszczyk* (Bansal & Jiang, 2025)	7	discrepancy theory
`discrete-bidding-games`	*Mean-Payoff and Energy Discrete-Bidding Games* (Avni & Sadhukhan, 2026)	8	games
`folded-rs-gsb`	*Explicit Folded Reed-Solomon and Multiplicity Codes Achieve Relaxed Generalized Singleton Bounds* (Chen & Zhang, 2025)	8	coding theory
`gg26-proximity`	*Optimal Proximity Gaps for Subspace-Design Codes and (Random) Reed-Solomon Codes* (Goyal & Guruswami, 2025)	6	coding theory
`lcc-exp-lb`	*An Exponential Lower Bound for Linear 3-Query Locally Correctable Codes* (Kothari & Manohar, 2024)	6	coding theory
`monotonicity-test-lb`	*Boolean Function Monotonicity Testing Requires (Almost) n^{1/2} Queries* (Chen et al., 2026)	8	property testing
`mts-diamond-lb`	*The Randomized k-Server Conjecture is False!* (Bubeck et al., 2023)	8	online algorithms
`multiflow-gap-planar`	*Improved Lower Bounds on Multiflow-Multicut Gaps* (Kalantarzadeh & Kumar, 2025)	8	approximation
`negations-small-depth`	*Negations Are Powerful Even in Small Depth* (Cavalar et al., 2026)	7	circuit complexity
`pcp-alphabet-soundness`	*Near Optimal Alphabet-Soundness Tradeoff PCPs* (Minzer & Zheng, 2024)	7	PCP
`planar-is-reconfig-kernel`	*A Linear Kernel for Independent Set Reconfiguration in Planar Graphs* (Bousquet & Cranston, 2026)	7	parameterized complexity
`posbody-chasing`	*Chasing Positive Bodies* (Bhattacharya et al., 2023)	8	online algorithms
`qma-qcma-oracle`	*Separating QMA from QCMA with a Classical Oracle* (Bostanci et al., 2026)	8	quantum
`strategy-improvement-lb`	*The Worst-Case Complexity of Symmetric Strategy Improvement* (van Dijk et al., 2024)	8	games
`three-taxi`	*Online 3-Taxi on General Metrics* (Coester & Poon, 2026)	8	online algorithms
`unitary-synthesis-lb`	*A One-Query Lower Bound for Unitary Synthesis and Breaking Quantum Cryptography* (Lombardi et al., 2024)	8	quantum
`xor3-parallel-repetition`	*Parallel Repetition for 3-Player XOR Games* (Bhangale et al., 2024)	8	PCP
`quantum-crypto-algorithmica`	*Quantum cryptography in Algorithmica* (Kretschmer et al., 2023)	8	quantum computing

A.2 SOLVER ENGINEERING: 30 PROBLEMS

Task	Problem	Source	Objective
`ahc058-atcoder`	Hierarchical production planning	AtCoder Heuristic Contest 058: Apple Incremental Game	maximize
`bpp-bpplib`	One-dimensional bin packing	BPPLIB	minimize
`ce-pace2021`	Cluster editing	PACE 2021 (heuristic track)	minimize
`celar-rlfap`	Radio link frequency assignment (RLFAP)	CALMA/CELAR benchmarks (ZIB archive; MiniZinc Challenge)	minimize
`christmas-tree-ahc041`	Christmas-tree forest partitioning	AtCoder Heuristic Contest 041: Christmas Tree Cutting	maximize
`coloring-dimacs`	Graph coloring	DIMACS / COLOR02 benchmark (+ gc_instances)	minimize
`curriculum-itc2007`	Curriculum-based course timetabling	ITC 2007 Track 3	minimize
`cvrp-cvrplib`	Capacitated vehicle routing	CVRPLIB	minimize
`dfvs-pace2022`	Directed feedback vertex set	PACE 2022	minimize
`ds-pace2025`	Dominating set	PACE 2025 (heuristic track)	minimize
`glasscut-roadef2018`	2D guillotine glass cutting with defects	ROADEF/EURO Challenge 2018	minimize
`gp-dimacs10`	Balanced graph partitioning (k=4, 3% balance cap)	DIMACS10 / Walshaw partition archive (34 graphs + 9 generated)	minimize
`hs-pace2025`	Hitting set	PACE 2025 (heuristic track)	minimize

Task	Problem	Source	Objective
itc2021-sports-timetabling	Sports timetabling	ITC 2021 (5th International Timetabling Competition)	minimize
jsplib-jssp	Job-shop scheduling (makespan)	JSPLib (252 classic + 31 generated)	minimize
logic-syn-iwls2023	Logic synthesis (minimum AIG)	IWLS 2023 Programming Contest (IWLS 2023 Contest Organizers, 2023)	minimize
maxclique-dimacs	Maximum clique	DIMACS clique benchmark (classic 80-instance set)	maximize
maxcut-mqlib	MaxCut	MQLib (Dunning et al., 2018)	maximize
maxsat-mse	Weighted MaxSAT	MaxSAT Evaluation 2026 (anytime weighted set)	minimize
mr-roadef2012	Machine reassignment	ROADEF/EURO Challenge 2012 (ROADEF/EURO Challenge Organizers, 2012)	minimize
nurse-inrc	Nurse rostering	INRC-I (First International Nurse Rostering Competition, 2010)	minimize
oscm-pace2024	One-sided crossing minimization	PACE 2024 (exact track)	minimize
qap-qaplib	Quadratic assignment	QAPLIB (Burkard et al., 1997)	minimize
qbf-qbfeval	QBF solving with Q-resolution certificates	QBFEVAL'20 (PCNF track)	decision
radiation-mznchallenge	Radiotherapy planning	MiniZinc Challenge benchmarks	minimize
scp-orlib	Set covering	OR-Library (full official 87-instance set) (Beasley, 1990)	minimize
sygus-comp	Program synthesis (SyGuS)	SyGuS-Comp 2017-2019 benchmarks (deduplicated)	decision
tsp-tsplib	Traveling salesman (symmetric)	TSPLIB (Reinelt, 1991)	minimize
tw-heuristic	Treewidth decomposition	PACE 2017 (heuristic track)	minimize
vc-pace2019	Vertex cover (exact)	PACE 2019 (exact track)	decision

A.3 CONSTRUCTIONS: 20 TASKS

Task	Construction	Reference in source snapshot	Domain
ramsey-r55	R(5,5) lower-bound construction	N = 42 (R(5,5) in [43, 46]) (Exoo, J. Graph Theory 13(1):97-98, 1989)	Ramsey theory
ramsey-r46	R(4,6) lower-bound construction	N = 35 (R(4,6) in [36, 40]) (Exoo, Electronic Journal of Combinatorics 19(1), 2012)	Ramsey theory
ramsey-r3-10	R(3,10) lower-bound construction	N = 39 (R(3,10) in [40, 41]) (Exoo, SIAM Journal on Discrete Mathematics 2(4):488–490, 1989)	Ramsey theory
vdw-w27	van der Waerden number W(2,7) lower bound	n = 3703 (Rabung-Lotts, 2012)	Ramsey theory
capset-d7	Maximum cap set in dimension 7	236 points (upper bounds: 288, Thackeray 2022 preprint; 291, Versluis 2017 BSc thesis) (Calderbank-Fishburn, Designs, Codes and Cryptography, 1994)	additive combinatorics
covering-code-k10	Binary covering code K(10,1) (minimize codewords)	120 codewords (K(10,1) in [107, 120]) (Östergård, IEEE Trans. Information Theory, 1991)	coding theory
binary-code-a22-10	A(22,10) maximum binary code	64 codewords (A(22,10) in [64, 84]) (Östergård, 2005)	coding theory
binary-code-a24-10	A(24,10) maximum binary code	136 codewords (A(24,10) in [136, 268]) (Laaksonen-Östergård, arXiv:1604.06022, 2016)	coding theory
quantum-code-25-7	Quantum stabilizer code [[25,7]] maximum minimum distance	d = 5 (upper bound 7) (codetables.de (data stamp 2005-06-24), 2005) (Grassl, n.d.)	quantum codes
mols-n10	Mutually orthogonal Latin squares, per-n records (10 cells)	n={10,12,14,15,18,20,21,22,24,26} -> k={2,5,4,4,5,4,5,3,7,4} (Miller et al., Symmetry 16(12):1678, 2024)	combinatorial designs
cage-girth	(k,g)-cage minimum order (cells (3,13)/(4,10)/(3,16), minimize N)	272 / 320 / 936 (lower bounds 202/243/512) (Biggs 1989 ((3,13)); EJC DS16 survey + arXiv:2511.07247 (EJC 33(3) 2026) + House of Graphs (2026), 1989/2026)	extremal graph theory
sorting-network-size	Sorting-network minimum comparators (8 cells n=13..20, minimize)	45/51/56/60/71/77/85/91 (16-wire 60 = Green 1969) (Dobbelaere sorting-network table (updated 2025-11), 1969/2025)	algorithms
ex-c3c4-maxedges	Erdős girth-5 extremal problem ex(n,{C3,C4}) (cells n=80/124/175/198)	321/629/1012/1166 edges (arXiv:2508.05562v1, Table 1 + AGT-Kulak data repository, 2025) (Goedgebeur et al., 2025)	extremal graph theory
kissing-d9	9-dimensional kissing number lower bound	306 points (d=9 in [306, 363]) (Leech-Sloane, 1971)	sphere packing
kissing-d13	13-dimensional kissing number lower bound	1154 points (d=13 in [1154, 2064]) (Zinoviev-Ericson, 1999)	sphere packing
circles-in-square-n100	100 non-overlapping equal circles in a unit square (maximize radius)	r = 0.051401071774 (Packomania csq table, accessed 26 September 2026)	packing geometry
spherical-code-d3-n50	Tammes problem, 50 points on the sphere (minimize max inner product)	max inner product 0.868173209127 (Székely, 1974)	spherical codes

Task	Construction	Reference in source snapshot	Domain
tsp-gap	Metric TSP subtour-LP integrality gap (n=15..20)	Benoit-Boyd family per-n: 15: 11/9, 16: 27/22, 17: 53/43, 18: 26/21, 19: 77/62, 20: 76/61 (Benoit-Boyd, Mathematics of Operations Research 33(4):921–931, 2008)	approximation
graph-tsp-gap	graph-TSP subtour-LP integrality gap (n=11..20)	Desjardins family per-n values, window max 26/21 (Desjardins, University of Ottawa master's thesis, Conjecture 5.3.1, 2017)	approximation
rectilinear-tsp-gap	Rectilinear (L1) TSP subtour-LP integrality gap (n=11..16)	Zhong family per-n: 11: 17/15, 12: 8/7, 13: 23/20, 14: 22/19, 15: 7/6, 16: 142/121 (recomputed 2026-09-17) (Zhong, Discrete Applied Mathematics 365, Section 5.3, 2025)	approximation

A.4 Open conjectures: 30 conjectures

Task	Conjecture	Proposer / source	Domain
aaronson-ambainis	Aaronson-Ambainis conjecture	Aaronson & Ambainis (2014) [Theory of Computing 10(6), Conjecture 1.7; arXiv 2009]	quantum computing
alon-jaeger-tarsi	Alon-Jaeger-Tarsi conjecture (prime-field form)	Alon-Jaeger-Tarsi (late 1980s) [non-prime fields settled by Alon-Tarsi 1989]	combinatorics
bin-packing-mirup	MIRUP conjecture (modified integer round-up property, 1-D bin packing)	Scheithauer-Terno (1995) [European Journal of Operational Research]	integer programming
caccetta-haggkvist	Caccetta-Häggkvist conjecture	Caccetta & Häggkvist (1978)	graph theory
d-vs-r-total-functions	D(f) $\leq$ C·R(f)^2 for all total Boolean functions	Nisan (1991)	query complexity
evasiveness-graph-properties	Evasiveness conjecture (Karp's conjecture)	Karp (early 1970s) [weaker form: Rosenberg 1973]	query complexity
explicit-ramsey-graph-log	Paley-graph Ramsey conjecture (polylogarithmic form)	no single proposer (explicit Ramsey constructions goal since Erdős 1947) (-) [O(log q) strong form refuted by Graham-Ringrose 1990; benchmark uses polylog form]	pseudo-randomness
explicit-rigid-matrix	Valiant's matrix-rigidity conjecture (Hilbert-matrix family)	Valiant (1977) [MFCS 1977]	algebraic complexity
fourier-entropy-influence	Fourier Entropy-Influence conjecture (FEI)	Friedgut-Kalai (1996)	Boolean analysis
frege-not-pbounded	Frege systems are not polynomially bounded	Cook-Reckhow (1979)	proof complexity
frege-vs-extended-frege	Frege does not polynomially simulate Extended Frege	Cook-Reckhow (1979)	proof complexity
krw-composition-conjecture	KRW composition conjecture	Karchmer-Raz-Wigderson (1995) [k$\geq$2 fraction formulation (GMWW 2017 Conjecture 1.3)]	communication complexity
log-rank-conjecture	Log-rank conjecture	Lovász-Saks (1988) [FOCS 1988]	communication complexity
lovasz-hamiltonicity	Lovász's Hamiltonicity conjecture (connected vertex-transitive graphs have a Hamiltonian path)	Lovász (1969)	graph theory
mansour-conjecture	Mansour's conjecture (DNF Fourier concentration)	Mansour (1992/1995) [COLT 1992 / JCSS 1995]	learning theory
mds-conjecture-nonprime	MDS main conjecture over non-prime finite fields	Segre (1955) [prime case settled by Ball 2012]	coding theory
nexp-vs-tc0	NEXP not in TC0	no single proposer (-) [background: Williams 2011 (NEXP not in ACC); Chen-Tell STOC 2019]	circuit complexity
prg-constant-width-robp	o(log^2 n)-seed PRG for width-4 read-once branching programs	rooted in Nisan 1992; explicit width-4 form Braverman-Rao-Raz-Yehudayoff / Forbes-Kelley (1992/2010/2018) [definition layer repaired 2026-09-18 (seed-length computability)]	pseudo-randomness
np-vs-acc	NP not in ACC	no single proposer (-) [background frontier: Williams 2011; Murray-Williams 2018]	circuit complexity
omega-2	Matrix multiplication exponent omega = 2	no single proposer (central open problem since Strassen 1969) (-)	algebraic complexity
online-bin-packing-ratio	Optimal asymptotic competitive ratio of online bin packing (benchmark threshold R* $\leq$ 1.56)	classic problem, no single proposer (1970s-) [threshold pinned between lower bound 1.54278+ (2021) and upper bound 1.57829 (ESA 2018)]	online algorithms
online-matching-random-order	Competitive ratio of online bipartite matching under random arrival order	Karp-Vazirani-Vazirani (model); Goel-Mehta (random order) (1990/2008) [STOC 1990; SODA 2008]	online algorithms
permanent-vs-determinant	Permanent cannot be written as a polynomial-size determinant	Valiant (1979)	algebraic complexity
pfr-integers	Polynomial Freiman-Ruzsa conjecture over the integers (Marton's conjecture)	Ruzsa; integer convex-progression form Lovett-Regev (1990s/2017) [Lovett-Regev, Discrete Analysis 2017:8, Conjecture 1.2]	additive combinatorics

Task	Conjecture	Proposer / source	Domain
`planar-l1-gnrs`	GNRS conjecture (planar-graph l1 embedding with constant distortion)	Gupta-Newman-Rabinovich-Sinclair (1999) [FOCS 1999; journal version Combinatorica 2004]	metric embeddings
`polar-scaling-exponent`	Polar-code scaling-exponent conjecture (mu = 3.627)	Arikan (polar codes); mu=3.627 conjectured by Korada-Montanari-Telatar-Urbanke / Mondelli-Hassani-Urbanke (2009/2010/2016)	coding theory
`ryser-conjecture`	Ryser's conjecture (tau(H) $\leq$ (r-1)*nu(H) for r-partite r-uniform hypergraphs)	Ryser (c. 1970)	combinatorics
`total-coloring-conjecture`	Total Coloring Conjecture (chi"(G) $\leq$ Delta(G)+2)	Behzad; Vizing (independently) (1965/1968)	graph theory
`tuza-conjecture`	Tuza's conjecture (triangle edge cover $\leq$ 2x triangle edge packing)	Tuza (1981) [published 1984, Proc. Colloq. Math. Soc. János Bolyai 37]	graph theory
`xor-log-rank-conjecture`	XOR log-rank conjecture	Montanaro-Osborne; parity-decision-tree form Tsang-Wong-Xie-Zhang (2009/2013) [arXiv:0909.3392; FOCS 2013]	communication complexity

A.5 Reading the four task interfaces

The following examples explain the submission and evaluation unit. Full statements, input formats, and checker configurations belong to the versioned task packages. These descriptions do not assert an experimental outcome.

Theorem proving: corners-quasipoly. The task concerns the quantitative bound for corner-free sets of Jaber et al. (2025) and provides formal definitions and a fixed target theorem. Bare attempts must prove the target from the supplied mathematical context. Ladder and guided attempts additionally receive intermediate statements; guided attempts receive a proof idea. A local checkpoint proof is evaluated under the other checkpoint statements, whereas the complete-proof check excludes unproved checkpoint assumptions. The fixed target declaration and supplied assumptions determine the statement whose complete proof is scored.

Solver engineering: maxcut-mqlib. A submitted program receives a weighted graph from the MQLib collection (Dunning et al., 2018) and returns a vertex partition. The checker validates the partition and recomputes the total weight of edges crossing the cut. Final judging applies the same program across the designated instance family under the task's resource limits. This evaluates a reusable procedure; a saved cut for one graph is a different object. The final manifest specifies public and held-out instance counts and the limits used for each set.

Constructions: capset-d7. A cap set in $(\mathbb{Z}/3\mathbb{Z})^7$ contains no three distinct points whose sum is zero. The submission supplies a set of points and the required certificate. Verification checks membership, distinctness, and the cap-set property, then recomputes cardinality. Cardinality is the underlying objective; the numerical score transformation does not change the mathematical parameter being optimized.

Open conjectures: log-rank-conjecture. The task asks for a proof or refutation of a formal relationship between deterministic communication complexity and the rank of the communication matrix. Two slots hold proofs of P and $\neg P$. Each is checked without access to the unproved opposite slot. A special case can be retained as an auxiliary result, but receives no complete-resolution credit unless it proves or refutes the target as stated.

B Scoring and Verification

The task version fixes the submission interface, legal operations, checker, reference values, and resource limits. A benchmark score records an accepted submission under those rules. A proof or object that is mathematically valid but violates the submission interface is reported separately from the official score; mathematical interest does not override the scoring protocol.

B.1 THEOREM PROVING: CONDITIONAL COVERAGE AND COMPLETE PROOFS

For a task with m checkpoints, let I_j indicate that checkpoint j passes its isolated check. All checkpoints within a task receive equal credit, so conditional checkpoint coverage is

$$C = \frac{1}{m} \sum_{j=1}^{m} I_j. \tag{1}$$

When checking checkpoint j, the verifier supplies the other checkpoints' reference statements as assumptions. Passing a checkpoint establishes that obligation under the supplied assumptions. This rule applies equally to an intermediate lemma and to the main-theorem checkpoint. Coverage cannot by itself establish that the full theorem has been proved.

Complete-proof validation uses a separate check. The `assembly_full` path verifies the assembled proof; the `main_direct` path checks a direct proof of the main theorem. Acceptance requires the permitted fixed context and excludes unproved checkpoint assumptions and `sorryAx`. For guided and ladder tasks, an accepted complete proof receives full credit: $S = \max(F, C)$, where F is the complete-proof indicator. Complete-proof outcomes use the separately recorded indicator F; the guidance analysis reports both complete-proof success and partial-credit task scores. Bare tasks have one complete target and use the binary complete-proof result.

Future work may examine nonuniform checkpoint weights; all reported evaluations use equal weights within each task.

Formal tasks use Lean 4 (de Moura & Ullrich, 2021) and, where required, the mathlib library (The mathlib Community, 2020); task manifests pin their versions. The verifier also checks fixed statements and writable regions and inspects proof dependencies. A successful compiler exit alone is insufficient if a submission changes the problem or depends on disallowed assumptions. Reports retain the check mode, declaration name, permitted assumptions, and the dependency output needed to interpret acceptance.

B.2 SOLVER ENGINEERING: INSTANCE PERFORMANCE AND REFERENCE-MODEL COMPARISON

For an optimization instance with attained value A, naive-baseline value B, and reference value K, the ordinary case with a feasible baseline and a strictly better reference uses

$$s(A; B, K) = \frac{A - B}{K - B}. \tag{2}$$

The expression applies to both optimization directions when K is better than B. It assigns 0 at B and 1 at K, and is unbounded in either direction. Checkers validate feasibility and recompute A; invalid solutions, solver crashes, and budget overruns receive the task's failure score, which is zero for the reported optimization tasks. A valid solution worse than B can therefore score below an invalid submission; failure rates and raw objectives distinguish these outcomes. Decision tasks use their declared solve-rate or certificate-based rule.

Instances with equal baseline and reference values, a reference worse than the baseline, or no feasible baseline do not enter the ratio-score mean. Their raw outcomes remain in the evaluation records and may enter the Kimi-K3-relative comparison. An absent reference value is distinct from an absent model outcome. Instances are not discarded because a submitted solver fails, and missing judging records are tracked separately from failed solutions.

The ratio score measures improvement relative to the chosen baseline and reference. Its sensitivity is $|\Delta s| = |\Delta A|/|K - B|$: a wide baseline–reference difference can make distinct outcomes near K look numerically similar, while a narrow difference can yield very large contributions. The Kimi-K3-relative score provides a second comparison using the raw outcomes of the three Kimi-K3 attempts on each instance. Its scale depends on those observed outcomes and the floor below, rather than on $K - B$; the bounded transform limits the contribution of any one instance to the relative mean.

For optimization, set $a = A$ for maximization and $a = -A$ for minimization; for decision tasks, a is the binary solve indicator. Let μ_i and σ_i be the reference mean and sample standard deviation on

instance i, with variance denominator $n_i - 1$. The comparison uses

$$\widetilde{\sigma}_i = \max(\sigma_i,\, 0.02|\mu_i|,\, 10^{-9}), \qquad z_i = \frac{a_i - \mu_i}{\widetilde{\sigma}_i}, \qquad q_i = \Phi(z_i), \tag{3}$$

where Φ is the standard Gaussian cumulative distribution function. The score lies in $[0, 1]$; values multiplied by 100 are the same metric on a percentage scale. A valid candidate outcome equal to the reference mean receives 0.5. The average of Kimi-K3's individual transformed outcomes need not equal 0.5. The transform is a standardized comparison, not an empirical percentile or a probability of winning. It is most sensitive near the reference mean and compresses large deviations in either tail. The scale floor also affects sensitivity when repeated reference outcomes vary little. We therefore retain raw objectives and ratio scores alongside it.

For optimization instances, failed Kimi-K3 attempts supply no numerical reference value. The comparison includes only instances with at least two valid reference values; the same eligible instance set is used for all models. Decision failures supply a reference outcome of zero. A failed candidate solution receives relative score zero directly, rather than being interpreted as a zero-cost solution. Missing judging records are reported separately and are not filled with numerical zeros. We transform each instance outcome before averaging over instances, then average attempts within tasks and give the 30 tasks equal weight. Reference coverage, feasibility outcomes, and the unbounded ratio scores accompany this comparison.

Final solver execution follows the task's declared budget. Per-instance tasks limit each instance separately. Time-pool tasks permit allocation of a single total budget across the family; evaluation must preserve that allocation opportunity. Certificate requirements are also task-specific: for example, SAT unsatisfiability claims require the designated proof certificate and certificate checker. Final results use the final submitted program on the designated evaluation set.

B.3 Constructions: mathematical parameters and shaped scores

The checker verifies the submitted object and recomputes its mathematical parameter X. Let B_t and R_t denote the task's trivial baseline and documented reference. The task's calibrated attainment $u_t(X)$ assigns 0 at the baseline and 1 at the reference. For an ordinary linear calibration,

$$u_t(X) = \max\left(0, \frac{X - B_t}{R_t - B_t}\right), \qquad S_t(X) = u_t(X)^{p_t}. \tag{4}$$

The same ratio accommodates minimization when $R_t < B_t$. Task-specific nonlinear calibrations replace u_t where required. The scoring design uses $p_t = 3$ for 14 tasks and $p_t = 1$ for six, with no upper cap at the reference. Multi-parameter tasks apply their declared aggregation over all required settings, including missing submissions. A score exceeding 1 is evidence of exceeding the versioned reference only after validity and tolerance checks; its numerical excess is not the percentage improvement in X.

B.4 Open conjectures: full proof or refutation

A task supplies a formal proposition P and slots for P and $\neg P$. A valid proof in either direction earns 1; otherwise the complete-resolution score is 0. Each main slot is checked in isolation after removing its contradictory twin, without treating either unproved main statement as an assumption. Auxiliary statements may support the submitted proof only within the permitted verification context. Special cases, reductions, and counterexamples to related stronger statements are reported with their precise scope and do not change the binary target score.

C Task Development and Fidelity Audits

C.1 From a research source to a task

Agent-assisted source reviews identify the intended research claim, its quantifiers and parameter range, and the result or object accepted by the task. For formal statements, component reconciliation maps each source definition and assumption to the corresponding Lean declaration. Small examples and boundary cases check that definitions behave as intended. Re-translation provides an additional

comparison between the source and its formalization. Some re-translation passes were performed by the authoring agent, while other reviews used a fresh context. Shared definitions and statements are checked across the three theorem-disclosure levels.

Checker review exercises both valid and invalid submissions: empty outputs, malformed objects, incorrect objectives, incomplete certificates, and attempts to change fixed statements. Reference outputs are used where available. Open conjectures do not have a known positive solution merely because the task has passed development checks; their verifiers require independent checks of the proof-or-refutation interface and negative cases.

Table 7: Task-validation checks and the evidence used to assess each failure mode.

Check	Evidence	Failure addressed
Source correspondence	Source location, formal statement, reconciliation	Omitted assumptions or changed quantifiers
Definition behavior	Hand-checkable examples and executed checks	Degenerate or incorrectly encoded objects
Submission integrity	Fixed-region and proof-dependency checks	Changing the task or assuming the result
Checker behavior	Valid, invalid, and empty test outputs	Accepting an invalid answer or rejecting the documented interface
Version consistency	Versioned statements, environments, and scoring rules	Mixing incompatible statements or score definitions

C.2 Review triggered by unexpected outcomes

A full proof or a value beyond the reference triggers agent-assisted re-examination of the rollout, submitted artifact, and task statement. The review checks adherence to the task rules, the accepted proof's assumptions or the object's verified properties, and the scope of the claimed result. For numerical improvements, source comparison checks the prior reference, problem parameters, instance set, and resource conditions. Mathematical validity and literature novelty are documented separately. If a statement or checker changes, the corrected task receives a new version; results on different versions retain their original scope.

Potential inconsistencies in research sources. Development checks sometimes identified apparent missing boundary conditions, discrepancies between definitions and their use in proofs, or potential gaps in arguments. We treated these as concerns requiring source reconciliation, including checks for mistakes in our own formalization. Candidate tasks were excluded when such concerns could not be resolved. These agent-assisted findings informed task selection; they are not presented here as definitive judgments about the source publications.

C.3 Curation and evaluation population

The theorem collection was screened through pilot evaluations, with over 40 theorem problems retired across development rounds. The evaluation records retain the screening model and configuration, candidate and included task lists, and reasons for retirement. Results on the selected set characterize this research-oriented collection; performance-conditioned selection can affect comparisons involving the screening system. Task additions and retirements are recorded without changing historical evaluation manifests.

D Experimental Protocol and Reproduction

D.1 Systems, versions, and budgets

The evaluation compares six model–harness systems. Each attempt has a development budget of up to 12 hours. Network access is disabled for theorem and solver tasks and enabled for construction and conjecture tasks. Models use the supplied tools and libraries within the task rules. Solver judging has separate task-specific time and hardware limits.

Table 8: Models and primary harness configurations. Attempt counts apply to each task in the four tracks. Qwen3.8-max denotes the 0902 version; DeepSeek-Flash denotes v4.1 Flash.

Model	Harness	Effort	Attempts per task
GPT-6-Astra	Codex 0.154.0	max	1
Kimi-K3	Kimi Code 0.43.0	max	3
GLM-5.3	Claude Code 2.1.257	max	1
Qwen3.8-max	Claude Code 2.1.257	max	1
Grok-4.6	Grok Build 1.0.41	xhigh	1
DeepSeek-Flash	Claude Code 2.1.257	max	3

Grok-4.6 uses the Grok Build harness with thinking enabled and reasoning effort set to xhigh. Model development budgets and final solver execution budgets are recorded separately. Each result is associated with its task definition, submitted output, verification outcome, and resource use.

D.2 Solver task construction and execution

Sources and evaluation sets. The solver collection comprises 19 competition-derived tasks and 11 tasks based on instance libraries. Source type and execution budget are separate choices: either type can use per-instance limits or a shared time pool. Each task specifies a public development set, a final evaluation set, an executable submission interface, and a checker. Competition-derived tasks can retain the source's development/test distinction or define a new split of released instances. Library-derived tasks define their own splits; these are fixed in the task version rather than selected from model outcomes. Table 9 records the evaluated configurations.

Public instances and self-test results are available during program development. Hidden instance files and transformation parameters remain on the judging side, outside the model's offline development environment. In two tasks, the final set also includes public instances: maximum satisfiability uses 200 hidden and 11 public instances; radiation therapy uses 36 hidden and 11 public instances. Thus the final collections contain 1,892 judged instances, of which 1,870 are hidden and 22 overlap with the public sets. These overlaps are disclosed in the task instructions and retained in the reported scores.

Adapting an instance library. The quadratic-assignment task illustrates the conversion of a public library into a solver evaluation. From 136 QAPLIB instances (Burkard et al., 1997), the versioned split assigns 41 to hidden evaluation, 75 to public development, and 20 to neither set. Hidden selection is stratified by instance family and size; withholding the third group prevents the public complement from identifying the evaluation set exactly. Hidden instances independently permute facility and location indices. If π relabels facilities and σ relabels locations, a feasible assignment p becomes $p' = \sigma \circ p \circ \pi^{-1}$. Applying the same relabelings to the flow and distance matrices preserves every assignment's cost and the optimal value. The checker validates the submitted permutation and recomputes its cost on the transformed instance.

Other transformations follow the problem's structure, such as vertex relabeling for graph problems and item reordering for bin packing. These changes obstruct direct reuse of answers in the original indexing. Publicly released source instances may nevertheless have appeared in model training; the protocol does not establish their absence from training data. The submission rules require computation during judging and prohibit replay of stored final answers. Hidden access controls, transformations, and rollout audits address different ways a submitted program could bypass that requirement.

Development and execution budgets. The 12-hour budget covers model interaction and program development. Afterward, the frozen program is rebuilt and evaluated under the task's separate execution limits. Twenty-four tasks impose limits independently on each instance. Six tasks instead share one total time pool: bin packing, graph partitioning, sports timetabling, logic synthesis, quadratic assignment, and set cover. The judge processes the evaluation family while tracking the remaining pool, which is supplied to the program on each invocation. Time saved on one instance is available to later instances; these tasks are evaluated as one group to preserve this allocation rule. For exam-

ple, quadratic assignment has 1,230 seconds across its 41 final instances, with no separate 30-second cutoff per instance. Glass cutting uses two runs per instance, at 180 and 3,600 seconds, with weights 0.1 and 0.9. The limits in Table 9 describe our evaluation protocol; a competition source does not imply that all its original resource conditions are reproduced.

Parallel evaluation. For per-instance evaluation, we use up to ten isolated sandboxes concurrently, using ten whenever the number of instances and available resources permit. Each instance retains the specified time, CPU, and memory limits. Shared-pool tasks remain in one evaluation group to preserve cross-instance time allocation.

Table 9: Solver task sources, development instances, and final execution budgets. Public instances are available during development; hidden instances are withheld; overlap counts public instances also included in final judging. Times are seconds. *Each* denotes a separate limit per instance; *Pool* denotes one total across the final set.

Task	Source	Public	Hidden	Overlap	Budget
AtCoder AHC058	Competition	105	45	0	Each 2
Bin packing	Library	90	39	0	Pool 1,170
Cluster editing	Competition	100	100	0	Each 600
Frequency assignment	Library	13	8	0	Each 1,200
AtCoder AHC041	Competition	105	45	0	Each 2
Graph coloring	Library	132	72	0	Each 180
Course timetabling	Competition	12	6	0	Each 240
Vehicle routing	Library	80	34	0	Each 180
Directed feedback vertex set	Competition	100	200	0	Each 1,800
Dominating set	Competition	100	100	0	Each 300
Glass cutting	Competition	35	15	0	Each 180 / 3,600
Graph partitioning	Library	24	13	0	Pool 780
Hitting set	Competition	100	100	0	Each 300
Sports timetabling	Competition	29	16	0	Pool 960
Job-shop scheduling	Library	198	85	0	Each 600
Logic synthesis	Competition	100	100	0	Pool 4,000
Maximum clique	Library	44	24	0	Each 180
Maximum cut	Library	105	45	0	Each 600
Maximum satisfiability	Competition	11	200	11	Each 300
Machine reassignment	Competition	20	10	0	Each 300
Nurse rostering	Competition	9	60	0	Each 600
Crossing minimization	Competition	100	100	0	Each 1,800
Quadratic assignment	Library	75	41	0	Pool 1,230
Quantified Boolean formulas	Competition	13	57	0	Each 900
Radiation therapy	Competition	15	36	11	Each 1,200
Set cover	Library	48	26	0	Pool 780
Syntax-guided synthesis	Competition	100	60	0	Each 3,600
Traveling salesperson	Library	61	33	0	Each 600
Treewidth	Competition	100	100	0	Each 1,800
Vertex-cover certificates	Competition	100	100	0	Each 1,800

Glass cutting evaluates separate 180- and 3,600-second runs, weighted 0.1 and 0.9. Named source collections are listed in Appendix A.

D.3 Repetitions and aggregation

For model m, task t, and reported attempt r, let s_{mtr} be the score and R_m the number of attempts per task. Across all four tracks, $R_m = 3$ for Kimi-K3 and DeepSeek-Flash and $R_m = 1$ for the other systems. Within a track with task set $\mathcal{T}$, we report

$$\bar{s}_m = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \left(\frac{1}{R_m} \sum_{r=1}^{R_m} s_{mtr} \right). \tag{5}$$

Task-specific aggregation over instances or parameter settings precedes this calculation. Every task has equal weight. For binary theorem outcomes, the formula equals the number of complete proofs

divided by $20R_m$, giving denominators of 20 or 60. Case studies examine individual submissions; the main table retains average performance across attempts.

Solver evaluation. The solver evaluation uses the repetition counts in Table 8. The final evaluation contains 1,892 instances across 30 problems and 18,920 instance outcomes. Public development tests do not contribute to final scores; public overlap with evaluation instances is specified in Appendix D.2. Each submitted program is evaluated under the task's time and hardware limits. Independent checks of the evaluation records verify instance membership, solution validity, objective values, and aggregation.

Proof-guidance evaluation. Kimi-K3 and DeepSeek-Flash each contribute three attempts in each of the three settings on 20 theorems, giving 60 bare, 60 ladder, and 60 guided attempts per model. Guided and ladder share their task definitions and main targets, with guided additionally supplying proof ideas.

An accepted assembled proof or direct main proof counts as a complete-proof success. The per-attempt rate divides successful attempts by 60; theorem coverage counts targets with at least one success among the three attempts. All submitted proofs are checked using the task's fixed definitions and permitted axioms. Complete-proof outcomes and partial-credit task scores are reported separately.

Cross-setting case analyses also check that the main theorem and its definitions agree. The bare setting can differ in its supporting definitions, prompt wording, and verifier; Appendix E.3 details these differences for the selected cases.

Construction evaluation. All six models receive construction prompts that describe published reference bounds but omit the numerical score transformation. The 12-hour development budget applies to producing the submitted object and certificate. Post-submission verification evaluates mathematical validity without a fixed compilation-time cutoff; this rule applies to every model. Acceptance requires the unchanged submitted artifact to satisfy the task statement, fixed-context checks, and permitted proof dependencies. Scores use these verified outcomes, and reference improvements are distinguished from matching the reference within the stated numerical tolerance.

Open-conjecture evaluation. Each model attempts all 30 conjectures with the repetitions specified in Table 8. Success requires a complete proof or refutation of the stated conjecture; all reported scores are zero. Verified auxiliary results and special cases are analyzed separately as mathematical outputs. Auxiliary and generated file counts are not used as measures of mathematical progress.

D.4 Interpreting variation

The theorem and construction plots display variation across tasks in the selected collection. Figure 6 shows selected task-mean differences and individual Kimi-K3 outcomes across proof settings. The selection highlights large observed differences; the aggregate analysis retains all 20 paired theorems. Attempt indices across settings do not denote matched random seeds. Neither a task distribution nor a single attempt estimates run-to-run uncertainty for systems evaluated once. The systems use different harnesses, and network-enabled tasks can draw on external literature. Comparisons therefore describe the recorded evaluation conditions; they do not isolate the effect of a model's training procedure or establish coverage of TCS as a whole.

D.5 Recorded token usage and duration

Table 10 reports accumulated input and output tokens and recorded duration per task attempt. Each mean uses only attempts with an observed value for that field, with coverage reported alongside it. Missing telemetry does not remove an attempt from the performance evaluation.

Usage measurements primarily use the recorded API usage totals. Where API totals are unavailable, counts are reconstructed from request traces or estimated from recorded text. Accumulated input totals can count context repeatedly across requests; they do not measure the number of distinct tokens available to the model. Provider tokenization and accounting conventions can differ, and

the records do not consistently separate cached input or reasoning tokens. We therefore use these measurements to describe the evaluation, without converting them into monetary costs or a model-efficiency ranking.

Solver durations are estimated from execution logs and output timestamps, excluding final judging delays where these can be identified; budget-exhausting attempts are recorded at 12 hours. These estimates are reported separately by track and do not establish precise model execution time.

Attempts can finish before the 12-hour development budget is exhausted. Recorded durations can also include termination and reporting overhead beyond the nominal limit.

Table 10: Recorded resource use per attempt. Means use observed values only; n/N reports measurement coverage. M and k denote millions and thousands of tokens. Solver times are estimates from platform and artifact timestamps. Proof-guidance runs are excluded.

Model	Input (M)	n/N	Output (k)	n/N	Time (h)	n/N
Theorem proving (bare)						
GPT-6-Astra	6.51	20/20	91.1	20/20	0.68	20/20
Kimi-K3	183.66	60/60	1599.8	60/60	9.32	60/60
GLM-5.3	29.34	19/20	586.0	20/20	4.26	20/20
Qwen3.8-max	43.03	15/20	655.3	15/20	8.22	20/20
Grok-4.6	93.06	20/20	478.6	20/20	3.28	20/20
DeepSeek-Flash	15.48	58/60	553.5	59/60	1.24	60/60
Solver engineering						
GPT-6-Astra	58.00	30/30	398.1	30/30	5.22	30/30
Kimi-K3	82.69	90/90	294.6	90/90	7.34	90/90
GLM-5.3	89.89	30/30	298.3	30/30	8.01	30/30
Qwen3.8-max	67.35	30/30	349.0	30/30	8.38	30/30
Grok-4.6	14.37	30/30	121.5	30/30	2.26	30/30
DeepSeek-Flash	79.45	90/90	306.9	90/90	5.09	90/90
Constructions						
GPT-6-Astra	89.95	20/20	441.6	20/20	6.56	20/20
Kimi-K3	59.50	60/60	297.6	60/60	8.66	60/60
GLM-5.3	84.23	20/20	531.0	20/20	7.16	20/20
Qwen3.8-max	81.15	20/20	931.8	20/20	10.02	20/20
Grok-4.6	19.93	19/20	142.4	19/20	3.72	20/20
DeepSeek-Flash	59.59	59/60	525.1	59/60	3.91	60/60
Open conjectures						
GPT-6-Astra	60.36	30/30	468.1	30/30	4.58	30/30
Kimi-K3	76.66	90/90	461.8	90/90	5.16	90/90
GLM-5.3	93.83	30/30	447.9	30/30	3.92	30/30
Qwen3.8-max	97.68	30/30	591.1	30/30	7.62	30/30
Grok-4.6	13.47	30/30	111.6	30/30	0.96	30/30
DeepSeek-Flash	53.34	90/90	427.1	90/90	1.65	90/90

E Complete Results and Diagnostics

E.1 Task-level performance and repetitions

The following tables use the same experiment records as Table 2. All scheduled bare-theorem attempts are included using their recorded checker outcomes. Construction entries average task scores over attempts; their numerical differences do not measure proportional changes in mathematical parameters. Construction evaluation conditions are documented in Appendix D. Every system has zero complete resolutions on all 30 open conjectures, across three attempts per task for Kimi-K3 and DeepSeek-Flash and one for each other system.

Figures 8 and 9 display individual attempts underlying the task means. For theorem proving, Kimi-K3 completes the negations theorem in all three attempts, and universal Dijkstra and discrete bidding games in two attempts each. Its seven successful attempts therefore cover three distinct theorems.

Grok-4.6 and GLM-5.3 each complete the discrete-bidding theorem, while Qwen3.8-max completes the negations theorem. For constructions, the matrix distinguishes repeated attainment of the reference score from variation across attempts, including the occasional scores above 1. An above-reference score is interpreted under that task's scoring function and version; it does not by itself establish literature novelty.

Table 11: Bare theorem results. Each cell is complete-proof successes / scheduled attempts. All scheduled attempts are included. Theorem names match the individual-attempt matrix in Figure 8.

#	Theorem	Astra	Kimi-K3	GLM-5.3	Qwen	Grok-4.6	DeepSeek
1	Corners theorem	0/1	0/3	0/1	0/1	0/1	0/3
2	Negations in small-depth circuits	1/1	3/3	0/1	1/1	0/1	0/3
3	Universal Dijkstra	1/1	2/3	0/1	0/1	0/1	0/3
4	Three taxis	1/1	0/3	0/1	0/1	0/1	0/3
5	Monotonicity testing	1/1	0/3	0/1	0/1	0/1	0/3
6	QMA–QCMA oracle separation	1/1	0/3	0/1	0/1	0/1	0/3
7	Discrepancy	0/1	0/3	0/1	0/1	0/1	0/3
8	PCP alphabet soundness	0/1	0/3	0/1	0/1	0/1	0/3
9	Proximity testing	1/1	0/3	0/1	0/1	0/1	0/3
10	Folded Reed–Solomon codes	1/1	0/3	0/1	0/1	0/1	0/3
11	Metrical task systems	1/1	0/3	0/1	0/1	0/1	0/3
12	XOR parallel repetition	0/1	0/3	0/1	0/1	0/1	0/3
13	Unitary synthesis	1/1	0/3	0/1	0/1	0/1	0/3
14	Planar independent-set reconfiguration	1/1	0/3	0/1	0/1	0/1	0/3
15	Positive-body chasing	1/1	0/3	0/1	0/1	0/1	0/3
16	Locally correctable codes	0/1	0/3	0/1	0/1	0/1	0/3
17	Strategy improvement	1/1	0/3	0/1	0/1	0/1	0/3
18	Planar multiflow	1/1	0/3	0/1	0/1	0/1	0/3
19	Discrete bidding games	1/1	2/3	1/1	0/1	1/1	0/3
20	Quantum cryptography	0/1	0/3	0/1	0/1	0/1	0/3

Table 12: Construction task means over scheduled attempts. Verification conditions are documented in Appendix D.3.

Task	Astra	Kimi-K3	GLM-5.3	Qwen	Grok-4.6	DeepSeek
binary-code-a22-10	1.0160	0.8028	1.0000	1.0000	0.5079	1.0000
binary-code-a24-10	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
cage-girth	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
capset-d7	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
circles-in-square-n100	1.0000	1.0000	1.0000	1.0000	1.0000	0.6667
covering-code-k10	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
ex-c3c4-maxedges	1.0207	1.0138	1.0000	1.0207	1.0207	1.0069
graph-tsp-gap	1.0000	1.0000	1.0000	1.0000	1.0000	0.6667
kissing-d13	1.0000	0.9756	0.9375	0.9947	0.9426	0.9756
kissing-d9	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
mols-n10	1.0000	1.0000	1.0000	1.0000	1.0000	0.6667
quantum-code-25-7	1.9531	1.3177	1.0000	1.0000	1.0000	1.0000
ramsey-r3-10	1.0000	1.0000	1.0000	1.0000	1.0000	0.6667
ramsey-r46	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
ramsey-r55	1.0000	1.0000	1.0000	1.0000	0.0000	1.0000
rectilinear-tsp-gap	1.0000	1.0000	1.0000	1.0000	0.7683	1.0000
sorting-network-size	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
spherical-code-d3-n50	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
tsp-gap	1.0000	0.0000	0.5997	0.0000	0.0000	0.0000
vdw-w27	1.0000	1.0000	1.0000	1.0000	1.0000	0.6667

Table 13: Proof-guidance comparison on 20 paired theorems, with three attempts per model and level. Scores are percentages; Δ is guided minus ladder in percentage points. The evaluation protocol is described in Appendix D.3.

		Kimi-K3			DeepSeek-Flash		
#	Theorem	Guided	Ladder	Δ	Guided	Ladder	Δ
1	Corners theorem	25.00	20.83	+4.17	25.00	25.00	+0.00
2	Negations in small-depth circuits	100.00	80.95	+19.05	61.86	57.13	+4.73
3	Universal Dijkstra	100.00	79.17	+20.83	41.67	41.83	-0.17
4	Three taxis	83.67	75.00	+8.67	79.17	66.67	+12.50
5	Monotonicity testing	91.67	58.33	+33.33	70.67	62.50	+8.17
6	QMA–QCMA oracle separation	87.50	83.33	+4.17	58.33	70.83	-12.50
7	Discrepancy	61.90	33.33	+28.57	33.33	28.57	+4.76
8	PCP alphabet soundness	23.81	71.43	-47.62	76.18	66.67	+9.51
9	Proximity testing	100.00	100.00	+0.00	100.00	83.33	+16.67
10	Folded Reed–Solomon codes	100.00	66.67	+33.33	91.67	91.67	+0.00
11	Metrical task systems	12.50	58.33	-45.83	41.67	33.33	+8.33
12	XOR parallel repetition	54.00	50.00	+4.00	12.50	20.83	-8.33
13	Unitary synthesis	83.33	54.17	+29.17	71.00	66.67	+4.33
14	Planar independent-set reconfiguration	47.62	71.43	-23.81	71.42	76.19	-4.77
15	Positive-body chasing	100.00	88.00	+12.00	100.00	54.17	+45.83
16	Locally correctable codes	27.78	33.33	-5.55	22.22	16.67	+5.56
17	Strategy improvement	66.67	37.50	+29.17	45.83	25.00	+20.83
18	Planar multiflow	66.67	41.67	+25.00	41.67	29.17	+12.50
19	Discrete bidding games	20.67	50.33	-29.67	37.50	20.83	+16.67
20	Quantum cryptography	87.67	54.00	+33.67	75.00	45.83	+29.17
	Mean	67.02	60.39	+6.63	57.83	49.14	+8.69

E.2 Solver objectives and score sensitivity

Table 14 reports both solver metrics and failed outcomes; Table 16 gives all 30 task means, and Table 17 retains every attempt's original score. Figure 11 shows the 300 relative scores before averaging attempts.

Table 14: Solver metrics and failed outcomes. Relative scores are percentages; task scores are unbounded difference ratios or the task's solve rate. Failed outcomes count rejected solutions, crashes, and timeouts across all instances, including those outside relative-score coverage.

Model	Relative (%)	Task score	Failed / judged
GPT-6-Astra	64.64	1.0711	44/1892
Kimi-K3	48.73	0.7814	314/5676
GLM-5.3	48.97	0.8112	108/1892
Qwen3.8-max	49.19	0.6473	86/1892
Grok-4.6	43.72	0.7468	227/1892
DeepSeek-Flash	48.20	0.7795	347/5676

Coverage across an instance family. Table 15 separates producing valid optimization outputs from meeting a certificate-based target. For each optimization attempt, we compute the fraction of judged instances with accepted feasible solutions, then average attempts within tasks and tasks equally. An attempt contributes to the full-set count only if every instance has a valid result. Across the 27 optimization tasks, Astra's mean validity is 99.29%, compared with 93.92–97.50% for the other models; it returns valid solutions on every instance in 24 tasks. The three certificate-based tasks instead require the designated correctness or target certificate; their binary completion rates are reported separately. These statistics describe reliability across the instances of each problem, while objective values measure solution quality.

Table 15: Coverage of the submitted solvers across instance families. Validity is averaged over instances, attempts, and the 27 optimization tasks, giving tasks equal weight. Full sets count attempts with a valid result on every instance. Certified completion averages the three certificate-based tasks: quantified Boolean formulas, syntax-guided synthesis, and vertex-cover certificates.

Model	Validity (%)	Full sets / attempts	Certified (%)
GPT-6-Astra	99.29	24/27	90.12
Kimi-K3	97.21	75/81	72.07
GLM-5.3	97.50	25/27	68.77
Qwen3.8-max	97.21	25/27	77.69
Grok-4.6	93.92	23/27	81.60
DeepSeek-Flash	97.23	74/81	68.19

Reference coverage and exclusions. The Kimi-K3-relative comparison uses a common set of 1,860 judged instances for every model. The excluded 32 instances lack two valid numerical Kimi-K3 outcomes, which are required to estimate the sample standard deviation in Appendix B.2. On 23 maximum-satisfiability instances, all three attempts are rejected because the output lacks the required objective-value line. Among nine sports-timetabling instances, eight have no feasible Kimi-K3 solution and one has only a single feasible solution; the other submissions violate hard constraints. These are recorded unsuccessful solutions, with no missing judging records. We exclude these instances from every model's relative score to maintain a common comparison set. Their outcomes remain in the raw records and failure counts, and the original task-score rules remain unchanged.

All other tasks have full relative-score coverage. The scale floor is active on 1,424 of the 1,860 included instances, so the comparison frequently uses the specified minimum scale rather than the empirical standard deviation alone. Kimi-K3's overall score is 48.73%, rather than a forced 50%: failures receive zero and the nonlinear transformation of repeated values need not average to 0.5.

Table 16: Solver task means on the Kimi-K3-relative scale (%). Coverage counts instances with at least two valid Kimi-K3 reference outcomes over all judged instances. Attempts are averaged within each task; ranks use unrounded scores.

#	Task	Coverage	Astra	Kimi-K3	GLM-5.3	Qwen	Grok-4.6	DeepSeek
1	AtCoder AHC058	45/45	83.81	50.12	48.34	52.35	45.45	67.34
2	Bin packing	39/39	56.15	50.00	48.41	44.16	44.53	48.40
3	Cluster editing	100/100	50.29	50.00	50.21	49.99	49.13	50.05
4	Frequency assignment	8/8	53.79	50.24	52.50	53.79	53.79	50.61
5	AtCoder AHC041	45/45	85.36	50.03	54.06	69.49	23.16	19.00
6	Graph coloring	72/72	54.90	50.07	52.56	49.00	50.21	51.50
7	Course timetabling	6/6	88.90	49.63	28.50	35.78	33.72	60.38
8	Capacitated vehicle routing	34/34	63.46	50.21	50.68	55.01	53.11	61.17
9	Directed feedback vertex set	200/200	52.06	50.11	45.18	47.30	15.23	48.44
10	Dominating set	100/100	45.09	50.01	21.06	31.00	25.39	43.40
11	Glass cutting	15/15	98.49	50.23	3.18	1.21	0.01	2.68
12	Graph partitioning	13/13	58.37	49.89	55.99	49.41	41.90	52.57
13	Hitting set	100/100	56.52	50.05	60.26	54.56	42.23	45.01
14	Sports timetabling	7/16	94.45	42.92	45.33	44.04	44.03	58.81
15	Job-shop scheduling	85/85	60.40	50.09	53.79	43.49	34.89	40.13
16	Logic synthesis	100/100	90.21	50.16	66.34	67.23	63.95	68.74
17	Maximum clique	24/24	53.69	50.26	53.45	51.63	49.62	51.20
18	Maximum cut	45/45	53.70	50.00	51.15	52.01	50.54	45.88
19	Maximum satisfiability	188/211	73.32	47.99	61.73	59.65	43.58	55.01
20	Machine reassignment	10/10	70.87	50.43	68.75	67.33	45.93	50.64
21	Nurse rostering	60/60	65.68	50.20	64.52	65.31	54.78	54.29
22	One-sided crossing minimization	100/100	50.00	50.00	50.00	50.00	49.98	49.99
23	Quadratic assignment	41/41	50.83	50.00	50.68	49.87	50.40	50.18
24	Quantified Boolean formulas	57/57	48.63	30.70	33.76	36.01	31.24	34.68
25	Radiation therapy	47/47	60.76	50.55	60.76	51.61	57.97	54.62

#	Task	Coverage	Astra	Kimi-K3	GLM-5.3	Qwen	Grok-4.6	DeepSeek
26	Set cover	26/26	59.76	50.18	55.57	48.82	50.93	55.49
27	Syntax-guided synthesis	60/60	48.33	48.33	48.33	48.33	48.33	48.33
28	Traveling salesperson	33/33	53.17	50.00	51.85	51.73	43.61	51.40
29	Treewidth	100/100	87.71	49.98	53.52	46.34	48.57	48.56
30	Vertex-cover certificates	100/100	70.54	39.43	28.50	49.12	65.35	27.59

Table 17: Original solver task scores by attempt. Difference ratios are unbounded; decision tasks use solve rates.

Task	Astra	Kimi-K3			GLM	Qwen	Grok	DeepSeek		
	1	1	2	3	1	1	1	1	2	3
AtCoder AHC058	1.001	0.963	0.973	0.972	0.967	0.971	0.959	0.988	0.978	0.985
Bin packing	1.000	0.788	0.791	0.788	0.735	0.686	0.690	0.691	0.782	0.779
Cluster editing	1.004	1.004	1.003	1.002	1.004	1.002	1.000	1.003	1.003	1.003
Frequency assignment	1.003	0.991	1.003	1.002	1.001	1.003	1.003	1.000	1.003	1.003
AtCoder AHC041	1.013	0.996	0.991	0.982	0.992	1.001	0.973	0.921	0.968	0.984
Graph coloring	1.097	0.934	0.812	0.854	0.940	0.827	0.847	0.862	0.909	0.944
Course timetabling	0.919	0.477	0.413	0.622	0.350	0.448	0.313	0.756	0.647	-0.029
Capacitated vehicle routing	0.975	0.825	0.874	0.640	0.761	0.860	0.841	0.916	0.956	0.964
Directed feedback vertex set	1.000	0.990	0.993	0.996	0.976	0.961	0.326	0.996	0.974	0.980
Dominating set	0.878	0.904	0.974	0.838	0.592	0.736	0.666	0.758	0.913	0.858
Glass cutting	1.006	0.890	0.927	0.956	0.844	0.758	0.621	0.787	0.782	0.829
Graph partitioning	1.005	0.972	0.916	0.945	0.983	0.730	0.849	0.979	0.969	0.928
Hitting set	0.924	0.951	0.856	0.747	0.958	0.878	0.765	0.754	0.676	1.007
Sports timetabling	0.922	0.407	0.359	0.357	0.385	0.638	0.381	0.419	0.646	-0.101
Job-shop scheduling	1.001	0.979	0.964	0.960	0.980	0.952	0.913	0.951	0.951	0.877
Logic synthesis	0.982	0.601	0.493	0.760	0.726	-3.865	0.704	0.860	0.851	0.532
Maximum clique	1.000	1.000	0.774	1.000	1.000	0.981	0.906	1.000	0.842	0.925
Maximum cut	1.024	0.963	0.965	0.961	0.973	0.988	0.969	0.934	0.805	0.814
Maximum satisfiability	0.807	0.239	-0.071	0.227	0.411	0.347	0.156	0.328	0.097	0.228
Machine reassignment	1.121	0.801	0.782	1.058	1.030	1.026	0.952	0.574	0.612	0.685
Nurse rostering	0.997	0.983	0.987	0.979	0.995	0.995	0.985	0.991	0.989	0.975
One-sided crossing minimization	1.000	0.999	0.980	0.999	1.000	1.000	0.925	1.000	0.997	0.961
Quadratic assignment	0.991	0.953	0.959	0.958	0.983	0.953	0.980	0.978	0.964	0.961
Quantified Boolean formulas	0.667	0.333	0.333	0.125	0.375	0.417	0.292	0.375	0.417	0.375
Radiation therapy	1.013	1.005	1.012	1.011	1.013	1.010	1.010	1.010	1.013	0.980
Set cover	0.998	0.968	0.928	0.898	0.967	0.898	0.952	0.943	0.976	0.969
Syntax-guided synthesis	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Traveling salesperson	0.993	0.465	0.811	-0.251	0.777	0.287	-0.453	0.706	0.802	0.164
Treewidth	3.792	0.158	0.153	-0.053	0.312	0.279	0.993	0.002	0.327	0.425
Vertex-cover certificates	1.000	0.292	0.222	0.986	0.306	0.653	0.889	0.181	0.083	0.556

Table 18 illustrates how raw instance outcomes enter the unbounded ratio score. On treewidth instance `he134`, Astra attains width 201 against a locally measured FlowCutter reference of 248. Because the baseline exceeds this reference by only one, the score is 48. This internal reference supports a scoring diagnostic, without establishing an improvement over a published record. On logic-synthesis instance `ex37`, Qwen's submitted circuit passes validation but has a much larger objective than the baseline, producing a large negative contribution. These values are retained in the task mean.

Table 18: Raw objectives and ratio scores for two solver instances. Both are minimization tasks with valid submitted solutions. B is the baseline, K the task reference, and A the attained objective. The treewidth reference is locally measured; these examples illustrate scoring behavior.

Task	Model	Instance	B	K	A	$s(A; B, K)$
Treewidth	GPT-6-Astra	he134	249	248	201	48.00
Logic synthesis	Qwen3.8-max	ex37	698	106	273 020	-460.00

Equal weighting of tasks does not eliminate the influence of an extreme contribution within a task. The Kimi-K3-relative comparison uses the reference model's raw outcomes to provide a separate scale, while the raw objectives preserve the size and direction of the observed improvements. Feasibility and solve outcomes remain meaningful when the Kimi-K3 reference has too few valid numerical observations to define the relative score. Neither a ratio score above 1 nor a single improved instance establishes that a solver improves every instance in its family.

E.3 PROOF-GUIDANCE SCORES

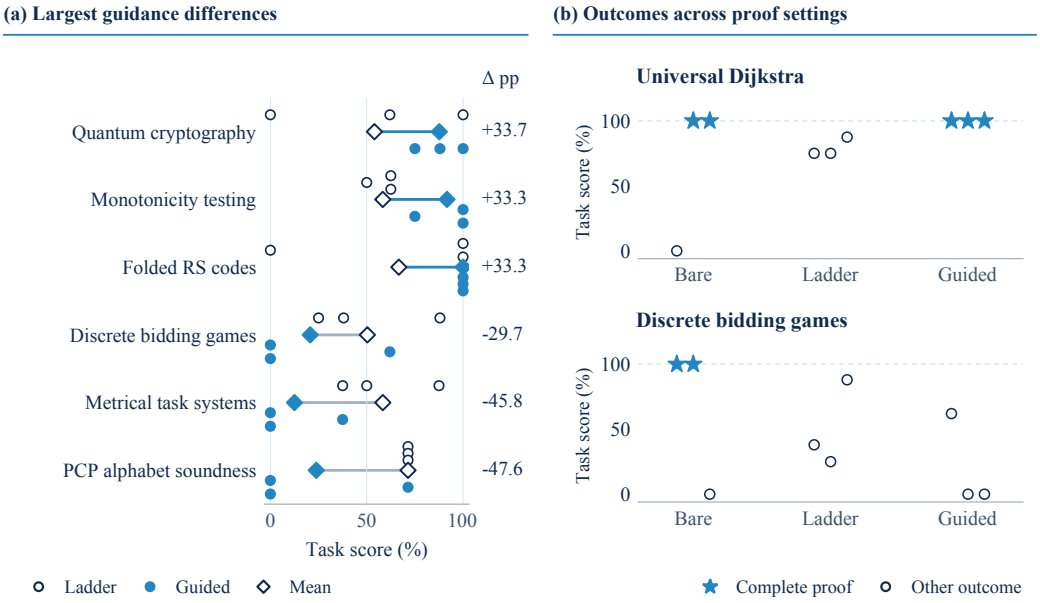


Figure 6: **Kimi-K3 performance across proof settings.** (a) The six largest absolute guided-minus-ladder mean differences among 20 theorems. Circles show individual attempts; diamonds mark means, with differences in percentage points. (b) Universal Dijkstra and discrete bidding games, which retains bare success with incomplete ladder/guided attempts. Each marker is an independent attempt; stars identify complete proofs. Bare receives binary credit, while ladder/guided also award partial checkpoint credit. Horizontal offsets separate the three attempts. Full results and checkpoint details appear in Appendix E.3.

Table 19 compares complete-proof outcomes on the same 20 targets. From bare to ladder to guided, the numbers of theorems solved at least once increase from 3 to 4 to 7 for Kimi-K3, and from 0 to 2 to 6 for DeepSeek-Flash. Kimi-K3 has equal per-attempt success under bare and ladder (7/60), then rises to 18/60 under guided; DeepSeek-Flash has 0/60, 2/60, and 11/60, respectively. Each setting uses three attempts per theorem.

Table 19: Complete proofs across three disclosure settings. Each model has three attempts on each of the same 20 theorems. A theorem is solved if at least one attempt proves it completely. The evaluation protocol is described in Appendix D.3.

Model	Outcome	Bare	Ladder	Guided
Kimi-K3	Successful attempts	7/60 (11.7%)	7/60 (11.7%)	18/60 (30.0%)
	Theorems solved	3/20	4/20	7/20
DeepSeek-Flash	Successful attempts	0/60 (0.0%)	2/60 (3.3%)	11/60 (18.3%)
	Theorems solved	0/20	2/20	6/20

Table 13 reports task scores, including partial credit, on all 20 theorems. Kimi-K3 averages 67.02% under guided and 60.39% under ladder, a difference of 6.63 percentage points; DeepSeek-Flash averages 57.83% and 49.14%, a difference of 8.69 points. Guided has a higher mean on 14 theorems for each model; Kimi-K3 has five decreases and one tie, while DeepSeek-Flash has four decreases and two ties. Each theorem contributes equally after averaging its three attempts. Pooling all checkpoints across tasks would instead give more weight to theorems with more checkpoints. Appendix D.3 describes the evaluation protocol; the figures below examine Kimi-K3 in more detail.

Figure 10 separates the three attempts within each level. For example, monotonicity testing improves from 62.5%, 62.5%, and 50% under ladder to 100%, 100%, and 75% under guided. In universal

Dijkstra, ladder scores are 75%, 75%, and 87.5%, while all three guided scores are 100%. These are independent attempts, without pairing by random seed. Thus a difference in task means can reflect either a shift across several attempts or a single unusually low outcome. These observations describe the recorded attempts and do not identify the cause of a particular failure.

Guided and ladder checkpoints are checked in isolation, with other checkpoint statements supplied as assumptions. This includes the main-theorem checkpoint. Consequently, success at that checkpoint does not establish a complete proof. A full-proof claim requires a separate check of the assembled or direct proof without unproved checkpoint assumptions. Table 19 reports complete-proof success independently of partial-credit scores; the selected cases below show the underlying acceptance patterns.

Bare success across proof settings. Table 20 lists all three theorems solved by Kimi-K3 in at least one bare attempt. Discrete bidding games retains bare success with no full-score ladder or guided attempt. Dijkstra has three accepted guided proofs; both cases appear in Figure 6b. Small-depth negations provides a contrasting case: bare succeeds three times, ladder reaches full credit once, and guided reaches it three times. Values are taken from the reported package scores, retaining their source precision.

Table 20: Kimi-K3 outcomes on every theorem with a successful bare attempt. Bare counts complete proofs; ladder/guided entries give the three task scores in percent. Attempt order within each setting does not imply pairing across settings.

Theorem	Bare	Ladder (%)	Guided (%)
Universal Dijkstra	2/3	75, 75, 87.5	100, 100, 100
Discrete bidding games	2/3	38, 25, 88	62, 0, 0
Small-depth negations	3/3	57.14, 85.71, 100	100, 100, 100

Statement and scoring checks. For Dijkstra, static comparison across the bare and scaffolded task versions confirms that the main statement and shared definitions agree after renaming the external interval-DAG bound and its bound variable; the scaffolded tasks additionally define the heap operations used by their checkpoints. For bidding games, the main statement and shared definitions agree, with additional helper declarations in the scaffolded tasks. The negations statements and shared definitions agree without renaming. These checks establish a common formal target for each case while retaining the differences in supplied context, prompts, and verifiers. In both selected tasks, the source verifiers award full credit to an accepted direct main proof as well as to an assembled proof. The sub-full scores therefore cannot be explained solely by a requirement to complete every prescribed intermediate slot.

Checkpoint evidence. Figure 7 separates complete-proof acceptance from isolated checkpoint acceptance. All three Dijkstra guided submissions have accepted complete proofs. The first two also pass all eight isolated checkpoints; the third passes only one because extracting the checkpoint bodies disrupts its Lean scopes. Thus a full task score need not equal the fraction of accepted isolated checkpoints. Bidding's second ladder attempt passes two checkpoints and receives a score of 0.25, without an accepted complete proof. Its third ladder attempt passes all intermediate checkpoints but leaves the main-theorem checkpoint unproved. These distinctions separate completion of the target theorem from progress along the supplied proof structure.

Bidding-game case: a direct invariant proof. The target asks for a budget-agnostic winning strategy, whose action depends on the current vertex and tie-breaking advantage rather than the exact available budget. Theorem 27 of Avni & Sadhukhan (2026) constructs this strategy by applying value iteration at a trimmed threshold budget. Its proof tracks the excess budget, which is nondecreasing and bounded, and partitions play into intervals on which this quantity stays constant. This gives the initial energy bound $M = (k+1)(W+H)$, where k is the total integer budget, W bounds absolute edge weights, and H is the maximum energy threshold at the threshold budgets.

One successful bare submission uses the same strategy and bound but organizes the argument as a single inductive invariant. For configuration $c = (v, B)$, let $e(c)$ be the integer budget above the

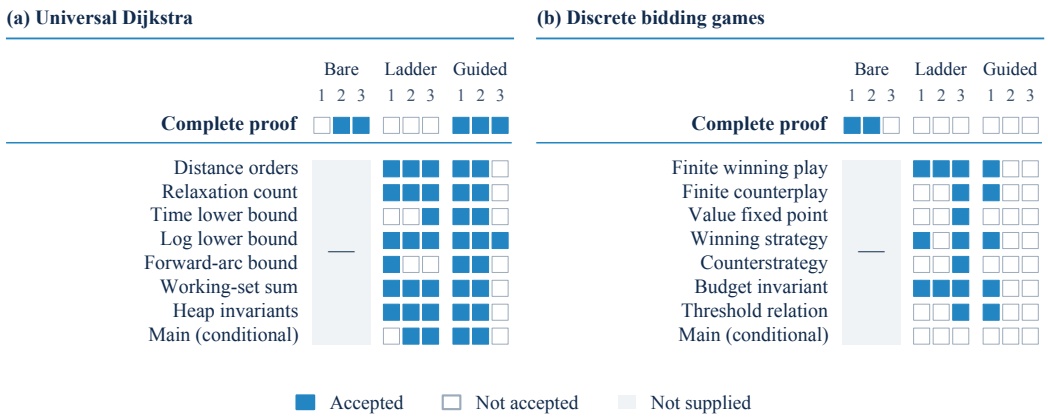


Figure 7: **Complete-proof and checkpoint acceptance for the Kimi-K3 cases.** The top row records complete proofs. Each lower row is a conditional checkpoint, including the main-theorem slot. Columns number independent attempts within each setting; gray denotes checkpoints absent from bare. Dijkstra's third guided submission has an accepted complete proof despite failures in seven isolated checkpoint checks. In bidding's third ladder attempt, all intermediate checkpoints pass while the main slot does not.

trimmed threshold, and let $u(c)$ be the value-iteration energy threshold at that trimmed budget. Its helper lemma proves, for every legal transition $c \to c'$ consistent with the strategy,

$$u(c) + w(v, v') + (W + H)\big(e(c') - e(c)\big) \geq u(c'). \tag{6}$$

Writing S_n for the sum of the first n edge weights, induction yields

$$M + S_n \geq u(c_n) + M - (W + H)e(c_n) - H \geq 0, \tag{7}$$

where the last step uses $e(c_n) \leq k$ and $u(c_n) \geq 0$. This establishes the energy guarantee directly for every play prefix. The submitted proof defines its own supporting lemmas rather than completing the supplied checkpoint sequence. Independent rechecks accept both successful bare submissions with only Lean's standard axioms. The strategy and energy bound agree with the source theorem; the evidence supports a difference in proof organization, without establishing a new mathematical result or a shorter proof overall.

Seven accepted checkpoints with the target unfinished. The third ladder attempt provides a concrete comparison. Its final judge log accepts all seven intermediate checkpoints, including the value-iteration fixed point, the budget invariant, and the threshold-average property. The main theorem remains a `sorry` placeholder, and both complete-proof checks fail; the exact reward is $7/8 = 0.875$ (shown as 88% in Table 20). Near the end of the 12-hour budget, the last build still reports the unfinished declaration and the final response records the target as incomplete. Thus this attempt's partial credit comes from intermediate obligations that were accepted conditionally, while the final strategy-existence argument was not completed. The successful bare proof and this incomplete ladder proof share the formal target but use independent attempts and different supplied contexts. Their contrast identifies the remaining obligation; it does not establish that guidance caused the failure. In the second guided attempt, the model proposed the auxiliary lemma `muN_kStar_le_xN`, then identified a counterexample involving tie-breaking. The lemma was absent from the supplied task; the attempt exhausted its budget while revising the proof. A third guided attempt worked in auxiliary Lean files but left the designated submission unchanged, yielding no accepted checkpoints. This distinguishes unfinished mathematical obligations from failure to integrate working files into the submitted artifact.

E.4 Interpreting intermediate conjecture outputs

The AJT case study establishes a bounded-dimensional theorem through independent Lean verification and exact certificate checks (Appendix F.3). This special case leaves the general conjecture

unresolved. The Paley case study examines an existing bound represented in Lean, an implication conditional on an unproved uniform estimate, and a finite clique witness. For each, we identify the exact statement, its relation to the conjecture, and the supporting source or computation. Appendix F.5 gives the details, including an independent check of the clique's 36 edges.

These observations explain the scope of one unsuccessful attempt. They do not estimate how frequently the systems produce such outputs across the conjecture collection. In particular, a conditional proof leaves its hypothesis as a mathematical obligation, and a counterexample to a stronger claim does not refute the stated target. The complete-resolution score remains zero for this attempt.

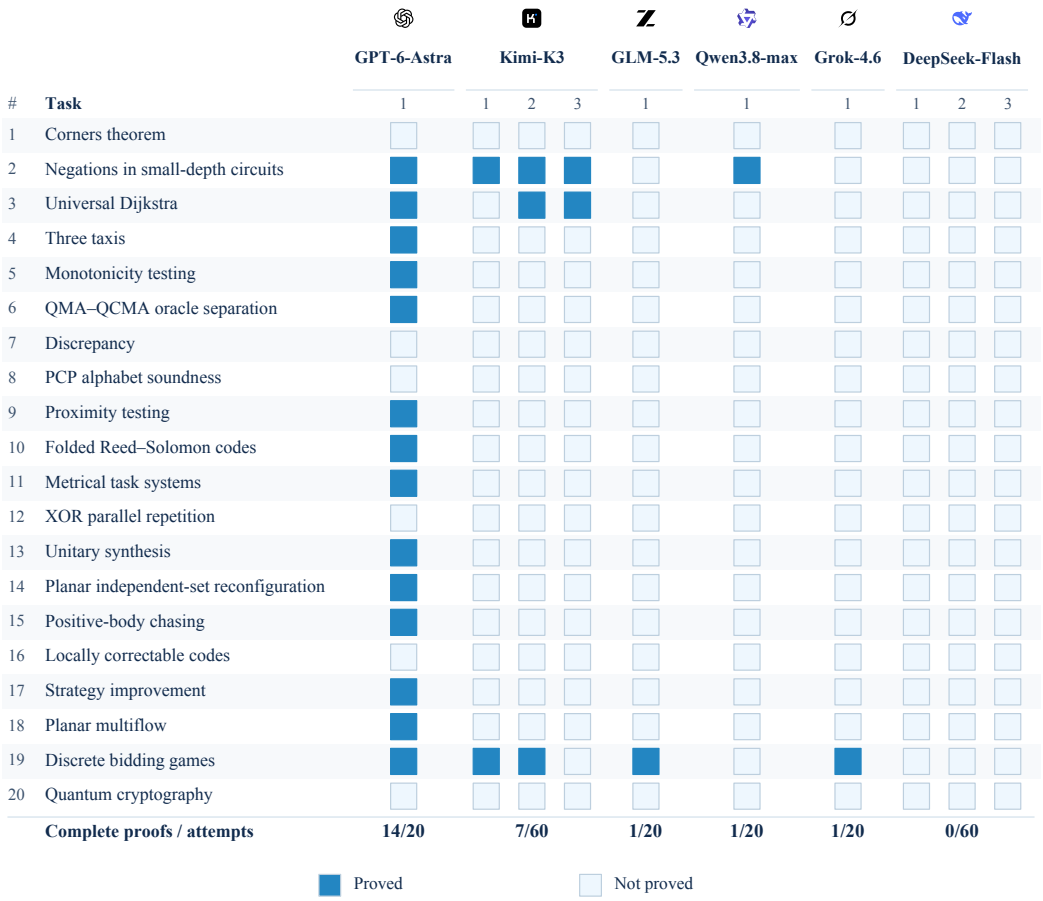


Figure 8: **Complete theorem proofs across individual attempts.** Each square represents one scheduled attempt. Filled squares denote accepted complete proofs; unfilled squares denote submissions without an accepted complete proof. The six systems contribute 200 attempts on the same 20 theorems. Columns are numbered within each model; they do not imply shared random seeds across systems. Task indices and theorem names match Table 11.

		GPT-6-Astra	Kimi-K3			GLM-5.3	Qwen3.8-max	Grok-4.6	DeepSeek-Flash		
#	**Task**	1	1	2	3	1	1	1	1	2	3
1	Binary codes: A(22, 10)	1.016	1	1	0.408	1	1	0.508	1	1	1
2	Binary codes: A(24, 10)	1	1	1	1	1	1	1	1	1	1
3	Cages and girth	1	1	1	1	1	1	1	1	1	1
4	Cap sets: dimension 7	1	1	1	1	1	1	1	1	1	1
5	Circle packing: 100 circles	1	1	1	1	1	1	1	1	0	1
6	Covering codes: K(10, 1)	1	1	1	1	1	1	1	1	1	1
7	Graphs without 3- or 4-cycles	1.021	1	1.021	1.021	1	1.021	1.021	1.021	1	1
8	Graph TSP	1	1	1	1	1	1	1	1	1	0
9	Kissing number: dimension 13	1	0.938	0.995	0.995	0.938	0.995	0.943	0.938	0.995	0.995
10	Kissing number: dimension 9	1	1	1	1	1	1	1	1	1	1
11	Mutually orthogonal Latin squares	1	1	1	1	1	1	1	0	1	1
12	Quantum codes: n = 25, k = 7	1.953	1	1.953	1	1	1	1	1	1	1
13	Ramsey: R(3, 10)	1	1	1	1	1	1	1	1	1	0
14	Ramsey: R(4, 6)	1	1	1	1	1	1	1	1	1	1
15	Ramsey: R(5, 5)	1	1	1	1	1	1	0	1	1	1
16	Rectilinear TSP	1	1	1	1	1	1	0.768	1	1	1
17	Sorting networks	1	1	1	1	1	1	1	1	1	1
18	Spherical codes: 50 points in 3D	1	1	1	1	1	1	1	1	1	1
19	Metric TSP	1	0	0	0	0.600	0	0	0	0	0
20	Van der Waerden: W(2, 7)	1	1	1	1	1	1	1	1	1	0

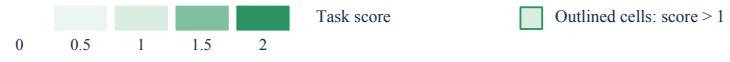


Figure 9: **Construction scores across individual attempts.** The matrix contains all 200 scheduled attempts. Cells show task scores rounded to three decimal places; outlines identify scores above the versioned reference of 1. Color follows a common score scale, while mathematical parameters and scoring functions differ by task. Evaluation conditions are documented in Appendix D.

Kimi-K3

		Ladder			Guided			Mean difference
#	Task	1	2	3	1	2	3	Guided − ladder (pp)
1	Corners theorem	25	12	25	25	12	38	+4.17
2	Negations in small-depth circuits	57	86	100	100	100	100	+19.05
3	Universal Dijkstra	75	75	88	100	100	100	+20.83
4	Three taxis	75	75	75	88	88	75	+8.67
5	Monotonicity testing	62	62	50	100	100	75	+33.33
6	QMA–QCMA oracle separation	88	88	75	88	88	88	+4.17
7	Discrepancy	57	43	0	57	57	71	+28.57
8	PCP alphabet soundness	71	71	71	71	0	0	-47.62
9	Proximity testing	100	100	100	100	100	100	+0.00
10	Folded Reed–Solomon codes	0	100	100	100	100	100	+33.33
11	Metrical task systems	88	38	50	38	0	0	-45.83
12	XOR parallel repetition	50	62	38	50	50	62	+4.00
13	Unitary synthesis	88	75	0	75	88	88	+29.17
14	Planar independent-set reconfiguration	71	71	71	0	71	71	-23.81
15	Positive-body chasing	88	88	88	100	100	100	+12.00
16	Locally correctable codes	33	33	33	33	17	33	-5.55
17	Strategy improvement	50	0	62	75	62	62	+29.17
18	Planar multiflow	38	50	38	75	75	50	+25.00
19	Discrete bidding games	38	25	88	62	0	0	-29.67
20	Quantum cryptography	100	0	62	75	88	100	+33.67

Task score (%)

0 25 50 75 100

Figure 10: **Kimi-K3 task scores across individual attempts.** Each cell shows a package score as a percentage, rounded to an integer. Mean differences use the unrounded scores and are given in percentage points. The 20 paired theorems contribute 60 attempts per setting. Scores combine partial checkpoint credit with full credit for accepted complete proofs; complete-proof outcomes are shown separately for selected cases in Figure 7.

		GPT-6-Astra	Kimi-K3			GLM-5.3	Qwen3.8-max	Grok-4.6	DeepSeek-Flash		
#	Task	1	1	2	3	1	1	1	1	2	3
1	AtCoder AHC058	83.8	41.8	54.1	54.5	48.3	52.4	45.4	72.3	60.2	69.6
2	Bin packing	56.1	49.7	49.8	50.5	48.4	44.2	44.5	44.8	50.2	50.2
3	Cluster editing	50.3	50.2	50.1	49.7	50.2	50.0	49.1	50.2	50.1	49.9
4	Frequency assignment	53.8	45.7	53.5	51.5	52.5	53.8	53.8	45.7	52.6	53.5
5	AtCoder AHC041	85.4	61.3	51.9	36.9	54.1	69.5	23.2	0.1	16.4	40.5
6	Graph coloring	54.9	52.5	47.5	50.2	52.6	49.0	50.2	50.8	51.2	52.5
7	Course timetabling	88.9	41.9	39.7	67.3	28.5	35.8	33.7	82.5	68.7	30.0
8	Vehicle routing	63.5	53.2	56.9	40.5	50.7	55.0	53.1	60.1	61.2	62.2
9	Directed feedback vertex set	52.1	47.6	51.0	51.7	45.2	47.3	15.2	50.5	45.0	49.9
10	Dominating set	45.1	49.0	58.9	42.1	21.1	31.0	25.4	33.3	53.3	43.6
11	Glass cutting	98.5	17.6	51.1	82.0	3.2	1.2	0.0	0.6	0.4	7.1
12	Graph partitioning	58.4	55.7	45.3	48.6	56.0	49.4	41.9	56.8	55.4	45.5
13	Hitting set	56.5	60.4	52.2	37.6	60.3	54.6	42.2	40.8	29.7	64.5
14	Sports timetabling	94.4	44.6	55.2	29.0	45.3	44.0	44.0	66.0	47.9	62.5
15	Job-shop scheduling	60.4	53.8	49.7	46.8	53.8	43.5	34.9	44.7	43.8	31.9
16	Logic synthesis	90.2	54.8	28.3	67.3	66.3	67.2	63.9	81.0	81.2	44.0
17	Maximum clique	53.7	53.2	43.9	53.7	53.5	51.6	49.6	53.0	49.0	51.7
18	Maximum cut	53.7	49.9	49.9	50.2	51.2	52.0	50.5	49.4	41.1	47.1
19	Maximum satisfiability	73.3	50.4	44.7	48.9	61.7	59.7	43.6	60.8	52.7	51.6
20	Machine reassignment	70.9	54.0	43.1	54.2	68.8	67.3	45.9	33.9	65.2	52.8
21	Nurse rostering	65.7	49.4	56.7	44.6	64.5	65.3	54.8	60.1	59.2	43.6
22	Crossing minimization	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0
23	Quadratic assignment	50.8	49.8	50.0	50.2	50.7	49.9	50.4	50.5	50.0	50.1
24	Quantified Boolean formulas	48.6	32.0	32.0	28.1	33.8	36.0	31.2	34.3	35.5	34.3
25	Radiation therapy	60.8	35.5	60.2	56.0	60.8	51.6	58.0	51.6	60.8	51.5
26	Set cover	59.8	55.9	47.5	47.1	55.6	48.8	50.9	53.4	56.7	56.4
27	Syntax-guided synthesis	48.3	48.3	48.3	48.3	48.3	48.3	48.3	48.3	48.3	48.3
28	Traveling salesperson	53.2	49.3	51.8	48.9	51.9	51.7	43.6	51.6	51.6	51.0
29	Treewidth	87.7	43.7	57.1	49.2	53.5	46.3	48.6	53.8	52.9	39.0
30	Vertex-cover certificates	70.5	26.2	22.4	69.7	28.5	49.1	65.4	21.9	17.2	43.6



Figure 11: **Solver results across individual attempts.** All 300 attempts are shown on the Kimi-K3-relative scale; a task uses the same eligible instances for all models. Original unbounded scores appear in Table 17.

F SUBMITTED OUTPUTS AND CASE STUDIES

The selected cases connect benchmark scores to mathematical parameters and proof obligations. Solver cases illustrate improvements on named instances; construction cases identify attained objects and task references; open-conjecture cases examine verified special cases and remaining proof obligations. Each is interpreted under its originating task version.

F.1 LOGIC-SYNTHESIS SOLVERS AND PUBLIC COMPETITION REFERENCES

The solver examples use three logic-synthesis submissions from the designated evaluation. GPT-6-Astra reduces the AND-gate count from 138 to 108 on instance `ex59`; Kimi-K3 reduces it from 19 to 18 on `ex30`; Grok-4.6 reduces it from 51 to 50 on `ex68`. The corresponding reductions are 21.74%, 5.26%, and 1.96%. Each reference is the minimum count across the four teams' publicly released AIG-track circuits in the IWLS 2023 repository (IWLS 2023 Contest Organizers, 2023). We downloaded all twelve reference circuits at commit `7dbfd510e259` and read their AND counts from the AIGER headers. Google's submission attains all three minima, with the other teams tying on `ex30`. The released files and their checksums accompany the case data.

Each example comes from the final evaluation of the submitted program; its objective and validity were checked against the evaluation records. The evaluation instances permute and negate inputs and outputs of the contest functions. These transformations preserve the AND-gate count because inversion is represented on edges. The examples illustrate individual outcomes, while task averages include all scored instances.

The comparison concerns the released competition circuits. The contest presentation also reports aggregate results incorporating unreleased organizer circuits, and these examples do not establish worldwide best-known values or a matched-resource advantage over the original competitors. Locally measured references are excluded from these three examples; the internally measured treewidth reference in Appendix E.2 serves only to explain score sensitivity.

F.2 GRAPH AND QUANTUM-CODE CONSTRUCTIONS

The graph task seeks dense 124-vertex graphs without triangles or 4-cycles. DeepSeek-Flash's 630 edges consist of a published 123-vertex, 621-edge seed and nine new edges to an added vertex; the selected neighbors have pairwise seed distance at least three. The seed size and 629-edge reference at 124 vertices appear in Table 1 of Goedgebeur et al. (2025), version 1. The quantum-code record gives $[[25, 7, 6]]$ binary stabilizer codes versus distance 5 in the designated table entry (Grassl, n.d.). These comparisons use the designated task references, with the source versions stated explicitly.

Verification of the displayed constructions. Figure 4 uses Kimi-K3's graph and quantum code, extracted directly from its submitted formal certificates. Figure 12 shows Astra's corresponding submissions, which attain the same parameters. Independent finite checks confirm that each graph has 124 vertices, 630 unique edges, no triangles, and at most one common neighbor for every vertex pair, excluding 4-cycles. The highlighted cycles are $0, 21, 32, 11, 121, 0$ for Kimi-K3 and $0, 2, 21, 46, 18, 0$ for Astra, establishing girth exactly 5. Each drawing shows every vertex and edge, with positions chosen for readability.

For each quantum code, the 18 submitted pairs of binary masks are converted directly to Pauli letters, with qubits 1–25 corresponding to bits 0–24. Each matrix retains the submitted row order. Independent checks confirm binary rank 18 and pairwise commutation, yielding $25 - 18 = 7$ logical qubits. The archived verifiers certify distance at least 6; independent distance checks corroborate this value. The edge lists, generator masks, source checksums, and drawing checks accompany the two figure data files.

A binary-code upper bound. Astra's submission proves $A(22, 10) \leq 82$. Here $A(22, 10)$ is the largest possible number of binary words of length 22 with pairwise Hamming distance at least 10. The published semidefinite-programming bound is 84 (Gijswijt et al., 2012), also listed in Brouwer's table when checked on September 26, 2026 (Brouwer, n.d.). The verified upper-bound score is 1.016042. The same submission supplies a 64-word code, which gives a separate lower bound. The construction verification protocol is described in Appendix D.3.

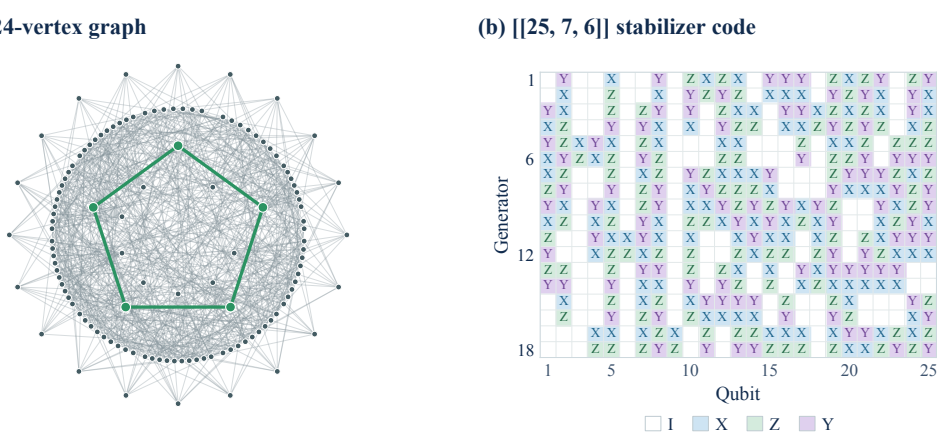


Figure 12: **Two constructions submitted by GPT-6-Astra.** (a) A 124-vertex, 630-edge graph without 3- or 4-cycles, with a shortest cycle highlighted. (b) The 18 submitted stabilizer generators of a $[[25, 7, 6]]$ quantum code. Both panels use Astra's actual submissions; they attain the same parameters as Kimi-K3's constructions in Figure 4.

F.3 A VERIFIED NINE-DIMENSIONAL ALON–JAEGER–TARSI RESULT

Statement and relation to prior work. The Alon–Jaeger–Tarsi conjecture asks whether every invertible matrix over a field with at least four elements admits a vector whose coordinates and image coordinates are all nonzero. Astra's Lean proof establishes

$$\forall p \geq 5 \text{ prime}, \quad \forall n \leq 9, \quad \forall A \in \mathrm{GL}_n(\mathbb{F}_p), \quad \exists x \in (\mathbb{F}_p \setminus \{0\})^n : \; Ax \in (\mathbb{F}_p \setminus \{0\})^n. \tag{8}$$

The contribution relative to the literature reviewed here is the case $p = 5$, $n = 9$. Yu's dimension guarantee $n < 2^{p-2}$ gives $n \leq 7$ over $\mathbb{F}_5$ (Yu, 1999); this guarantee is also stated in the introduction of Nagy & Pach (2025). Version 1 of Zeng et al. (2026), posted August 18, 2026, reports $n \leq 2^{p-2}$, including $n = 8$ over $\mathbb{F}_5$. The large-field result of Nagy & Pach (2025) applies when $61 < |\mathbb{F}| \neq 79$ and does not cover $\mathbb{F}_5$. The nine-dimensional result goes beyond these ranges. The comparison uses the publicly available versions checked on September 26, 2026. For $p \geq 7$, the dimensions in Equation 8 already fall within Yu's guarantee.

From a minimal counterexample to counting constraints. The proof first establishes smaller dimensions and then assumes a minimal counterexample A of dimension nine over $\mathbb{F}_5$. Let $\mathrm{wt}(x)$ denote the number of nonzero coordinates of x. Define

$$W_{ij} = \big|\{x : \mathrm{wt}(x) = i, \; \mathrm{wt}(Ax) = j\}\big|, \qquad V_{ij} = \big|\{x : \mathrm{wt}(x) = j, \; \mathrm{wt}(A^{\mathsf{T}}x) = i\}\big|.$$

These two 10×10 arrays have nonnegative integer entries. Invertibility fixes their row and column sums; a finite Fourier identity relates the two arrays. Minimality supplies further vanishing conditions, including $W_{22} = W_{23} = W_{32} = 0$.

Additional inequalities come from pairs of columns u, v. Each coordinate pair (u_i, v_i) belongs to one of seven classes: both zero, only the first nonzero, only the second nonzero, or one of the four nonzero cancellation ratios $u_i + tv_i = 0$. Figure 5(a) displays all 25 coordinate pairs. Their seven class counts $h_0, \ldots, h_6$ sum to nine, leaving $\binom{15}{6} = 5{,}005$ possible count vectors. The submitted proof checks the relevant inequalities over these finite count vectors and sums them across column pairs. Applying the argument to A, A^{-1}, A^{T}, and $(A^{\mathsf{T}})^{-1}$ yields four nonnegative linear quantities $L_1, \ldots, L_4$ in W and V. These finite checks concern coordinate counts, not enumeration of all invertible nine-dimensional matrices.

The submitted contradiction certificate. Collect the 200 entries of W, V and the four quantities L_o in a vector $z \geq 0$. The identities above give an integer linear system $Ez = b$. The submitted certificate contains integer multipliers λ satisfying

$$\lambda^{\mathsf{T}} E \geq 0, \qquad \lambda^{\mathsf{T}} b < 0. \tag{9}$$

Consequently, $0 \le \lambda^{\mathsf{T}} E z = \lambda^{\mathsf{T}} b < 0$, a contradiction. Of the 197 equations in the source certificate, 143 have nonzero multipliers; the Lean declaration retains precisely these 143 equations. Independent integer recomputation matches each retained equation, multiplier, and right-hand side to the Lean source and verifies Equation 9. All 204 resulting coefficients are nonnegative: 93 are positive and 111 are zero. Figure 5(b) shows their exact support, grouping the coefficients on W, V, and the four pair terms. For its displayed identity, we divide the certificate by $|\lambda^{\mathsf{T}} b|$ and write $c^{\mathsf{T}} = \lambda^{\mathsf{T}} E / |\lambda^{\mathsf{T}} b|$, giving $0 \le c^{\mathsf{T}} z = -1$.

Independent formal verification. The 71-module source project was rebuilt in a clean sandbox using Lean 4.33.0-rc1 and the task's pinned mathlib revision `b1c4a69a`, without reusing model-generated compiled files. The clean build used a 30 GB memory limit. The three declarations establishing the result over $\mathbb{F}_5$, its prime-field extension, and the minimum counterexample dimension depend only on `propext`, `Classical.choice`, and `Quot.sound`. Their dependency audits contain no `sorryAx` or additional axioms and do not depend on the unresolved conjecture slots. The definition of the required nowhere-zero vector agrees with the original task. Separate integer checks reproduce the certificate and the inequalities over the 5,005 class-count vectors.

Scope of the result. The verified result concerns dimensions at most nine over prime fields of order at least five. It leaves larger dimensions and general finite fields unresolved, so it does not constitute a complete resolution of the conjecture. As with the other special cases below, it receives no credit toward the binary open-conjecture score.

F.4 Additional verified results from open-conjecture attempts

We organize and independently verify four further results from Astra outputs in restricted parameter regimes. Independent source builds and axiom audits accept the statements below using only Lean's standard axioms, without dependencies on the unresolved conjecture slots. These checks establish the validity of the formal statements under their stated definitions and assumptions; they do not establish novelty. Our literature checks are not exhaustive, and some results may already be known or follow from existing work. We therefore report the verified statements without claiming that each constitutes a new advance.

A six-variable Fourier entropy–influence bound. For every Boolean function $f : \{-1,1\}^6 \to \{-1,1\}$, the submitted proof establishes

$$H(\widehat{f}^{\,2}) \le (7 \ln 2)\, I(f).$$

Here H is the Shannon entropy of the squared Fourier coefficients, using natural logarithms, and $I(f) = \sum_i \Pr[f(X) \ne f(X^{\oplus i})]$ is the total influence under uniform X. Equivalently, the constant is 7 when entropy is measured in bits. The proof reduces the bound to finite integer inequalities using the discrete values of Walsh coefficients, congruence constraints, and Parseval's identity. It covers all six-variable Boolean functions without enumerating their 2^{64} truth tables. The result gives an explicit constant in a fixed dimension; it does not establish a dimension-independent bound for the general conjecture.

Exact Hilbert-matrix rigidity at high target ranks. Let H_n be the rational Hilbert matrix with entries $(H_n)_{ij} = 1/(i+j+1)$ for $0 \le i, j < n$, and let $R_{H_n}(r)$ be the minimum number of entries that must change to obtain rank at most r. The submitted proof establishes the exact formula

$$R_{H_n}(r) = 2(n-r) - 1, \qquad \lfloor n/2 \rfloor \le r < n.$$

The upper bound constructs a rational matrix of rank exactly r differing in exactly $2(n-r)-1$ entries. The lower bound combines a row-and-column covering argument with nonsingularity of Hilbert submatrices. The conjectured rigidity bound concerns a lower target-rank regime, which remains unresolved by this formula.

An explicit influence bound for quadratic polynomials. For any real polynomial p of total degree at most two satisfying $0 \le p(x) \le 1$ on $\{-1,1\}^n$ and $\mathrm{Var}(p) > 0$, the submitted proof finds a coordinate i with

$$\mathrm{Inf}_i(p) \ge \frac{\mathrm{Var}(p)^2}{16}, \qquad \mathrm{Inf}_i(p) = \mathbb{E}\big[(p(X) - p(X^{\oplus i}))^2\big].$$

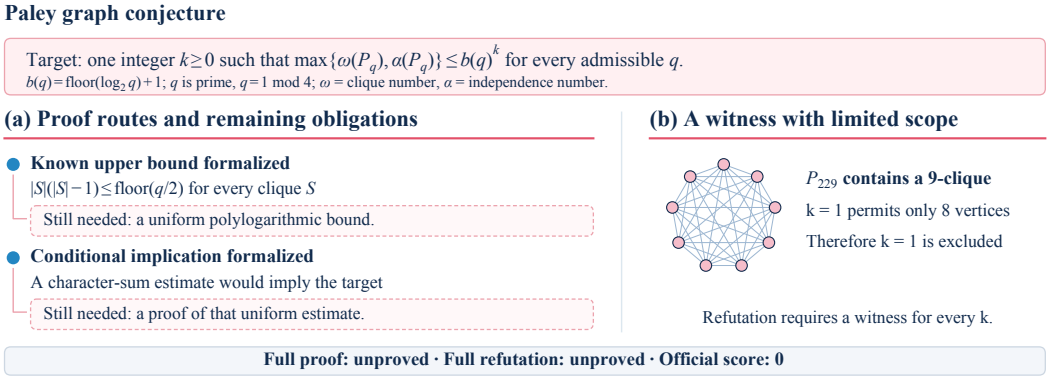


Figure 13: **An unresolved Paley graph attempt.** (a) Auxiliary statements and remaining obligations. (b) A witness excluding $k = 1$ only. All 36 edges of the nine-vertex clique are independently verified.

It also derives $\mathrm{Inf}_i(p) \geq (\varepsilon/2)^3$ whenever $\mathrm{Var}(p) \geq \varepsilon > 0$. The bound is independent of the number of variables. We report the explicit verified inequality without claiming that the qualitative degree-two case is new.

Finite-instance bounds for random-order online matching. The formal model reveals each online vertex's neighbors upon its arrival in a uniformly random permutation. Algorithms may adapt to the observed history and use an arbitrary finite number of independent uniform random bits. For graphs with maximum matching size at most two, an explicit algorithm guarantees a ratio of $7/8$, and a two-by-two instance family gives a matching upper bound. On graphs with three vertices on each side, another algorithm attains $5/6$, again with a matching upper bound. The latter lower bound is restricted to three-by-three graphs; it does not cover arbitrary graphs whose maximum matching size is three.

For every algorithm in this formal model, a five-by-five graph G with $\mathrm{OPT}(G) = 5$ satisfies

$$\frac{\mathbb{E}[\mathrm{ALG}(G)]}{\mathrm{OPT}(G)} \leq \frac{6788069}{8329500} \approx 0.814943154.$$

The proof uses a weighted family of 570 graphs and their online-vertex relabelings. A certificate bounds the weighted expected matching size by 27,152,276 at total weight 6,663,600, yielding the displayed ratio after division by five. Padding with isolated vertices extends the obstruction to larger vertex counts while preserving $\mathrm{OPT} = 5$. We report the finite-instance result under a strict multiplicative guarantee. This padding argument does not establish an upper bound when an additive error is allowed or when OPT must tend to infinity; a broader comparison with literature bounds requires matching these definitions and the algorithm class.

F.5 An unresolved Paley graph conjecture attempt

Figure 13 examines one GPT-6-Astra attempt on the Paley graph conjecture from the aggregate evaluation. In the Paley graph P_q, vertices are residues modulo a prime $q \equiv 1 \pmod 4$; distinct vertices are adjacent when their difference is a square modulo q. Write $b(q) = \lfloor \log_2 q \rfloor + 1$. The target asks whether some fixed nonnegative integer k bounds both the clique and independence numbers by $b(q)^k$ for every admissible prime.

A concrete witness. The submitted declaration `paley229_nine_clique` gives the vertex set

$$\{0, 1, 57, 71, 135, 154, 203, 215, 218\} \subseteq \mathbb{Z}/229\mathbb{Z}.$$

Its 36 unordered pairs all have square differences. A direct Python check reproduced this finite property when preparing the figure; the archived Lean axiom log also records this declaration and `exponent_one_fails` with only the permitted standard axioms. Since $b(229) = 8$, the witness excludes $k = 1$. It does not refute the existence of a larger uniform exponent. No claim about a new clique record or a new mathematical result is made.

Proof routes and missing obligations. The submitted polynomial-method development contains `hanson_petridis_bound`, stating $|S|(|S|-1) \le \lfloor q/2 \rfloor$ for any clique S. This reproduces the square-root-scale bound of Hanson & Petridis (2021). For any fixed exponent, that numerical upper bound eventually exceeds $b(q)^k$, so it alone cannot establish the required polylogarithmic bound. This is a limitation of the estimate, not a disproof of the conjecture.

The character-sum development proves a conditional implication. Let χ_q be the quadratic character, equal to 0 at zero, 1 at nonzero squares, and -1 otherwise. For a fixed k, the required hypothesis is

$$2 \sum_{x,y \in S} \chi_q(x-y) \le |S|^2 \quad \text{whenever } |S| \ge b(q)^k + 3,$$

uniformly over admissible primes q and sets S. The submitted implication derives the target with exponent $k+2$ from this hypothesis; it does not prove the hypothesis. The accompanying research notes record this missing obligation alongside unsuccessful attempts to apply the polynomial bound and an interval-clique construction.

Verification and scope. The clique witness is checked both in Lean and by independent finite computation. The auxiliary statements have only the permitted standard axioms as dependencies. Neither a proof nor a refutation of the full conjecture is established, so the attempt receives a score of zero. The figure distinguishes the verified finite witness and conditional results from the unresolved uniform bound.

F.6 Evidence for reference improvements and novelty claims

Artifact validity and novelty are recorded separately. The archive retains the original output, its checksum, task version, verifier and toolchain, verification log, recomputed value or accepted statement, and evaluation resources. A record comparison additionally includes the original source, date and location, prior value, parameter restrictions, and applicable resource conditions. A floating-point difference within the declared tolerance is not counted as an improvement.

Literature novelty depends on the sources checked and the scope of the claim. A result exceeding a task's internal comparison value can be reported as such even when the best published result differs. Claims about a new literature record require reconciliation with the relevant published or competition reference. Review of a proof of an open conjecture includes renewed statement-fidelity checking before the paper describes a resolution.

G Extension and Maintenance

G.1 Adding a task

An extension specifies the research question and source, submission type, accepted evidence, scoring rule, and execution resources. It then follows the development checks in Appendix C: source correspondence, executable validation, negative tests, trial evaluation, and expert review. The task manifest records its field and parameter coverage so that extensions can increase both breadth and depth without changing the meaning of existing scores.

New problems use the corresponding output interface: a formal proof, a reusable solver, an explicit object with verified properties, or a proof or refutation of a conjecture. A task combining several methods still specifies exactly what must be submitted and what the checker establishes. This permits research workflows to vary while the evaluated output remains explicit.

G.2 Versioning and corrections

Statements, environments, scorers, instance sets, and references remain fixed within an evaluation version. Corrections and additional tasks receive new versions with dated explanations. Reports identify which results are affected and whether re-evaluation is necessary. Source corrections can update a reference value in a later release with provenance; a model improvement is retained as an outcome rather than silently changing the comparison used for that evaluation.

Retired tasks remain available with their historical version and retirement reason. Comparisons across releases report the common task set or explicitly identify the changed collection. A maintained benchmark can thereby grow without rewriting earlier results.

G.3 Recording research outputs

The project maintains an artifact index linking results to the statements or objects checked, their originating task versions, and verification records. Research status distinguishes reproduction, validated improvement relative to a specified reference, and a novelty claim supported by source review. Verified special cases and reductions can be indexed alongside the complete-resolution score, preserving their mathematical content without changing that score retrospectively.