

10 Scalable Challenges in Combinatorics: A Public Benchmark Formalised in Lean

Vaibhav Bajpai¹, Nathan Bowler² Johannes Carmesin¹, and Utku Okur¹

¹ Institute of Discrete Mathematics and Algebra, Faculty of Mathematics and
Computer Science,
Technische Universität Bergakademie Freiberg, 09596 Freiberg, Germany
{vaibhav.bajpai,johannes.carmesin,utku.okur}@math.tu-freiberg.de

² Department of Mathematics, Universität Hamburg,
Bundesstraße 55, 20146 Hamburg, Germany
nathan.bowler@uni-hamburg.de

Abstract. AI systems resolve or contribute to resolving a growing number of research problems in mathematics. Currently the mathematical community faces the challenge that keeping track of and verifying proposed solutions absorbs a lot of time and effort. In response, we selected ten significant challenges in combinatorics and developed parameterised versions of each, to yield infinitely many increasingly difficult variants. We made each problem available on our website, together with an automatic solution verification tool. This allows us to keep track of progress on these problems in a simplified way.

Keywords: AI for Mathematics · Lean 4 · formal benchmark · automated theorem proving

Motivation. AI systems now generate mathematical content faster than experts can read: the growing VibeMathed registry already records hundreds of problems to which AI systems have contributed [3]. This calls for review, both to verify new results and to identify which content is most significant.

We propose a way to address this problem: compact sets of parameterised problems with automatic verification. This method can be used by mathematicians in a wide variety of fields to support ongoing automated assessment of AI capability, even as AI abilities approach or partly exceed those of human experts. As an example from our field, we present *Ten Challenges*, a collection of 10 combinatorial problems which can be attempted on our website³. Our proposal adopts the following methods, to maximise usefulness to the community:

1. the problems were selected by a small team of experts, then modified on the basis of feedback from a broader group of experts in the field;
2. we formalised each challenge in Lean 4 [1], using the Mathlib library [2];
3. we made a tool available on our website to check submissions automatically;
4. we thoroughly checked each formalisation to reduce the risk of errors;
5. each challenge is parametrised by a number, building towards the full solution as the number grows.

³ <https://tenchallenges.github.io/ten-challenges/>

The challenges. Each challenge is a well-known open problem in combinatorics, redesigned as a family of statements indexed by a parameter r (a natural number or an ordinal). Very small values of r correspond to simple exercises; the next few cover harder known results approaching the current frontier, while further values represent increasingly difficult open problems. These problems were selected not only for their significance but also for admitting such a scaling; in some cases the design of the scaled version required new ideas.

- | | |
|--|-------------------------------------|
| 1. Hadwiger conjecture | 6. Better-quasi-ordering of finite |
| 2. Excluded minors of | graphs under the minor relation |
| $\text{GF}(p^m)$ -representable matroids | 7. Sphere packing problem |
| 3. Caccetta–Häggkvist conjecture | 8. Ryser’s hypergraph conjecture |
| 4. Sidorenko conjecture for bipartite | 9. Union-closed sets conjecture |
| graphs with bounded sides | 10. Unfriendly partition conjecture |
| 5. Erdős–Hajnal conjecture | |

The submission system. A contestant submits, via our website, a Lean proof of one of the challenge statements for a chosen parameter, or a proof or disproof of the full statement. The submission is checked automatically: it must compile against the unmodified statement and may rely only on Lean’s standard axioms. Accepted submissions are recorded on a leaderboard with public and private tracks, where new advances are recognised.

Quality control of formalisations. A Lean statement may fail to capture the intended mathematical meaning, so every target statement is audited in two ways. First, a *theorem sandwich*: we prove in Lean that the statement follows from a set of lemmata and implies interesting corollaries, anchoring its intended meaning in established mathematics on both sides. Second, a *blind round trip*: a different AI model receives only the Lean statement, without the original informal statement, and translates it back into prose; the two readings are compared and any discrepancy is resolved before the formalisation is finalised.

Conclusion. We have proposed 10 central open problems in combinatorics as challenges and a benchmark for current AI systems in mathematics. A full solution—or even a solution for certain parameter values—would constitute a significant result for the field. As the volume of AI-generated mathematics grows, projects of this kind may help mathematicians efficiently track meaningful advances without inspecting the entire stream of new output.

References

1. de Moura, L., Ullrich, S.: The Lean 4 theorem prover and programming language. In: CADE 28. LNCS, vol. 12699, pp. 625–635. Springer (2021)
2. The mathlib Community: The Lean mathematical library. In: CPP 2020. ACM (2020). <https://doi.org/10.1145/3372885.3373824>
3. VibeMathed contributors: VibeMathed, a record of mathematical problems solved with AI. <https://vibemathed.com> (2026)