International Journal of Mathematics Trends and Technology
ISSN: 2231-5373/ https://doi.org/10.14445/22315373/IJMTT-V72I3P106

Volume 72 Issue 3, 49-52, March 2026

Original Article

Extending Computational Verification of Lemoine's Conjecture to 1500 Digit Odd Numbers

Duncan Ndegwa[1], Loyford Njagi[2], Stephen Luketero[3], Benard Nzimbi[4], Kikwai Benjamin[5]

[1,3,4]*School of Mathematics, University of Nairobi, Kenya.*
[2]*Department of Mathematics, Meru University of Science & Technology, Kenya.*
[5]*Department of Mathematics & Statistics, Machakos University, Kenya.*

[1]*Corresponding Author : ndegwaduncan34@gmail.com*

Received: 19 January 2026 Revised: 24 February 2026 Accepted: 15 March 2026 Published: 28 March 2026

Abstract *- Lemoine's conjecture claims that every odd integer greater than 5 can be represented as the sum of an odd prime and twice another prime. Recent computational studies have verified the conjecture up to large numerical bounds using deterministic and probabilistic primality testing methods. In this work, we introduce a novel mathematical decomposition algorithm that enables scalable verification of the conjecture for random odd integers of up to 1500 digits. This algorithm constructs candidate partitions $O = p + 2q$ by iterating over a subset of odd primes $p < O$, testing whether $q = (O - p)/2$ is prime. This approach narrows the search space and is tailored for extremely large integers. The primality of candidates is tested using the Miller-Rabin probabilistic method, supporting efficient computation. We demonstrate the algorithm's effectiveness across a range of large-digit odd numbers—the implementation, though secondary, was done in Python to validate the theoretical method. Our results extend the conjecture's verification range and establish a foundation for future theoretical and computational exploration.*

Keywords *- Lemoine's Conjecture, Prime Decomposition, Odd Integer Representation, Primality Testing.*

1. Introduction

Lemoine's conjecture, also referred to as Levy's Conjecture, asserts that every odd integer greater than 5 is expressible as the sum $O = p + 2q$, where p and q are primes[1]. The conjecture has similarities to Goldbach's weak conjecture and has been the subject of various computational efforts. Prior verifications have typically limited their scope to odd numbers up to 10^9[2], employing efficient sieving and primality testing algorithms.

Computational efforts have also further extended verification up to 10^{10}[3] through analytic techniques and numerical computations using advanced sieve methods and deterministic primality checks. However, extending verification to arbitrarily large odd integers remains a largely unexplored frontier due to computational constraints.

In this study, we present a novel decomposition algorithm capable of verifying Lemoine's Conjecture for randomly generated odd integers with up to 1500 digits. Our methodology emphasizes a mathematical strategy that reduces computational overhead through selective prime iteration and probabilistic primality testing. All results are verified via randomized trials to ensure diversity and statistical confidence.

2. Methodology and Results

The verification of Lemoine's Conjecture, every odd integer $O > 5$ admits a decomposition $O = p + s$, where p is prime, and s is a semiprime, requires a dual approach: a rigorous mathematical framework for partitioning and a scalable computational implementation. This section formalizes the decomposition algorithm for the restricted case $s = 2q$ (where q is prime) and details the computational tools used to validate the conjecture for odd integers up to 10^6. By combining probabilistic primality testing with systematic search strategies, our methodology bridges theoretical arithmetic principles and empirical verification, while introducing a conceptual recursive partitioning framework for broader analysis.

2.1. Mathematical Construction and Decomposition Algorithm
Let $O \in \mathbb{Z}$ be an odd integer $O \geq 7$. A Lemoine decomposition of O is a pair of primes (p, q) satisfying:

$$O = p + 2q$$

To construct such a pair, we employ an iterative search over candidate primes :
1. Enumerate odd primes p starting at $p = 3$, incrementing by 2 to preserve parity.
2. For each p, compute $q = \frac{O-p}{2}$.
3. Test both p and q for primality. Terminate upon the first valid (p, q).

2.1.1. Formal Algorithmic Structure
Let O be a randomly generated odd integer with n digits. The algorithm proceeds as follows:
1. Input: n-digit odd integer O.
2. Iteration: For $p \in \{3,5,7, \dots, \lfloor O/2 \rfloor\}$:
   - If p is prime, compute $q = \frac{O-p}{2}$.
   - If q is prime, return (p, q).
3. Output: Valid Lemoine pair (p, q) or failure.

Example 1
For $= 55$:
- $p = 17$ (prime), $q = \frac{55-17}{2} = 19$ (prime).
- Verification: $17 + 2 \times 19 = 55$

Unlike traditional sieve methods [4-5] that prioritize sequential integers, this approach targets randomly generated odd numbers with fixed digit lengths. This design ensures broad coverage of the integer space, mitigates selection bias, and challenges the conjecture under varied arithmetic conditions. The iterative search, though linear in complexity, benefits from early termination upon success, rendering it practical even for large O.

This approach directly addresses the decomposition of a single O, optimizing computational resources for targeted verification. The use of random O with fixed digit lengths further ensures statistical diversity, avoiding artifacts that might arise from sequential or patterned inputs.

This methodology bridges theoretical arithmetic principles with scalable computational tools, enabling systematic verification of Lemoine's Conjecture while offering insights into the structural properties of prime-semiprime partitions. The subsequent section details the implementation and empirical results for integers up to 1500 digits.

2.2. Computational Implementation and Primality Testing
Building on the mathematical framework for Lemoine decomposition in Section 2, we operationalize the search for primes p and q satisfying $O = p + 2q$ through a computationally efficient algorithm. The implementation leverages Python's native support (supplementary material, Appendix A) for arbitrary-precision integers and the probabilistic Miller-Rabin primality test to balance rigor and scalability [6].

2.2.1. Algorithmic Workflow
1. Input: A randomly generated n-digit odd integer O.
2. Prime Enumeration: Iterate over odd integers $p \geq 3$, testing each for primality via the Miller-Rabin test with $k = 5$ witness rounds.
3. Residual Computation: For each prime p, compute $q = \frac{O-p}{2}$.
4. Primality Verification: Apply the Miller-Rabin test to q. If q is prime, return the pair (p, q).
5. Termination: Halt upon identifying the first valid decomposition to optimize runtime.

Table 1. Runtime Performance of Lemoine Decomposition Verification Across Digit Lengths

Number of Random Odd Digits	Example in Supplementary Material	Runtime (seconds)
600	Example 23	10.47
650	Example 24	83.19
700	Example 25	148.48
750	Example 26	15.74
800	Example 27	151.50
850	Example 28	115.97
900	Example 29	95.82
950	Example 30	4.16
1000	Example 31	122.20
1200	Example 32	379.76
1300	Example 33	331.28
1400	Example 34	311.41
1500	Example 35	429.68

Table 1 quantifies the empirical runtime performance of the algorithm across odd integers spanning 600 to 1500 digits, demonstrating the practical feasibility of verifying Lemoine's conjecture at scale. Key observations include:

- Execution times exhibit non-monotonic behavior with increasing digit length. For instance, the 950-digit case (Example 30) resolved in 4.16 seconds, significantly faster than smaller-digit cases like 800 digits (151.50 seconds). This variability reflects the probabilistic nature of prime enumeration, where early detection of valid p and q pairs reduces computational effort.
- Despite the exponential growth in digit length, runtimes remain tractable. The 1500-digit decomposition (Example 35) required 429.68 seconds, underscoring the efficiency of the Miller-Rabin primality test in handling large integers.
- The method consistently identifies valid decompositions even for numbers exceeding 10^{1000}, reinforcing the conjecture's validity across the tested domain.

These results highlight the interplay between computational efficiency and mathematical conjecture verification, providing empirical support for Lemoine's conjecture while demonstrating the viability of probabilistic primality testing in large-number arithmetic.

3. Independent Primality Verification

To ensure reproducibility of the computational results, an independent high-precision probabilistic primality testing routine is provided in (Supplementary Material, Appendix B). The implementation is based on the Miller-Rabin algorithm executed with sufficiently many independent bases to render the probability of misclassification computationally negligible.

This routine is not used to generate the reported partitions; rather, it is supplied solely for independent validation. In particular, readers may verify the primality of the components appearing in representations of the form

$$n = p + 2q,$$

where n is odd and p, q are prime.

The implementation operates in arbitrary-precision arithmetic and supports integers exceeding 1500 decimal digits. Its inclusion ensures that all primality assertions underlying the reported decompositions can be independently confirmed using standard computational environments.

4. Conclusion

This study extends the computational verification of Lemoine's Conjecture to odd integers with up to 1500 digits by leveraging a mathematically grounded and computationally efficient decomposition algorithm. Central to our method is the formal structure of Lemoine decomposition: for an odd integer $O \geq 7$, we seek a pair of primes (p, q) such that $O = p + 2q$. The algorithm iteratively enumerates odd primes p, computes $q = (O - p)/2$, and validates both p and q for primality using the Miller-Rabin test. This process terminates at the first successful decomposition, significantly reducing computational overhead even for large-digit integers.

The formal algorithmic steps were shown to be scalable and effective across random odd integers with digit lengths ranging from 600 to 1500. For example, the decomposition of $O = 55$ into $p = 17$ and $q = 19$ illustrates the general mechanism in small-scale scenarios. More importantly, Table 1 provides runtime performance metrics across varying digit lengths. Interestingly, runtime does not scale monotonically with digit size. For instance, the 950-digit integer (Example 30) decomposed in just 4.16 seconds, outperforming several smaller-digit examples like the 800 -digit case (151.50 seconds). This variability highlights the probabilistic nature of prime placement within the tested range: early discovery of a valid (p, q) pair dramatically reduces execution time. These observations underscore the robustness and adaptability of the algorithm across wide numeric domains.

By verifying Lemoine's Conjecture for such extremely large odd integers, this research fills a notable gap in the literature where prior efforts were limited to smaller numeric bounds, often below 10^{14}. Our method not only confirms the conjecture for vastly larger values but also introduces a novel framework that integrates prime iteration strategies with probabilistic primality testing in a computationally feasible manner.

Conflicts of Interest
Authors declare no competing interests.

Funding Statement
No funding was available for this research.

References
[1] Junjie Huang, Ying Xiao, and Chenglian Liu, "A Study of Android Calculator Based on Lemoine's Conjecture," *AIP Conference Proceedings*, vol. 1982, no. 1, pp. 1-6, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[2] Mohammed Ghanim, "Confirmation of the Lemoine-Levy Conjecture," *Global Journal of Advanced Engineering Technologies and Sciences*, vol. 8, no. 12, pp. 1-7, 2021. [Publisher Link]

[3] Shaohua Zhang, "An Equivalent Form of Strong Lemoine Conjecture and Several Relevant Results," *Wuhan University Journal of Natural Sciences*, vol. 24, no. 3, pp. 229-232, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[4] Yoichi Motohashi, "An Overview of the Sieve Method and its History," *arXiv Preprint*, pp. 1-34, 2005. [CrossRef] [Google Scholar] [Publisher Link]

[5] Alina Carmen Cojocaru, and M. Ram Murty, *An Introduction to Sieve Methods and their Applications*, Cambridge University Press, 2006. [Google Scholar] [Publisher Link]

[6] Thomas H. Cormen et al., *Introduction to Algorithms*, MIT Press, 2022. [Google Scholar] [Publisher Link]