Extending the Lonely Runner Conjecture to 13 Runners via SAT Solving

David H. Silver (extending the work of Rosenfeld)

November 12, 2025

Abstract

We present a SAT-based approach for verifying the Lonely Runner Conjecture that achieves speedups of up to 6 orders of magnitude compared to backtracking methods. Building on the recent proof by Rosenfeld [3] for 8 runners, we extend the conjecture to 13 runners (k=12). Our method reformulates the verification problem as Boolean satisfiability and leverages modern SAT solvers. We show that our SAT encoding is logically equivalent to the verification problem underlying Rosenfeld's Lemma 6 approach and present complete verification for k=4 through k=12, providing the first proofs for 9, 10, 11, 12, and 13 runners.

1 Introduction

The Lonely Runner Conjecture, introduced by Wills [1], concerns $k+1$ runners with distinct constant speeds on a unit circular track. The conjecture states that for any runner, there exists a time when that runner is at distance at least $1/(k+1)$ from all other runners.

Recently, Rosenfeld [3] proved the conjecture for 8 runners (k=7) using a computational approach based on results by Malikiosis, Santos, and Schymura [2]. However, the computational cost grows dramatically: the hardest case for k=7 required approximately 32 hours of computation.

In this work, we reformulate the verification problem as Boolean satisfiability (SAT) and show that modern SAT solvers can verify instances orders of magnitude faster. This enables us to extend the conjecture to k=12 (13 runners), providing five new cases beyond Rosenfeld's result.

2 Background

2.1 The Lonely Runner Conjecture

Following Rosenfeld [3], we use the formulation:

Definition 1. A set $S \subseteq \mathbb{N}$ of size k has the *lonely runner property* (LR property) if there exists $t \in \mathbb{R}$ such that for all $v \in S$,

$$\|tv\| \geq \frac{1}{k+1}$$

where $\|x\|$ denotes the distance from x to the nearest integer.

2.2 Rosenfeld's Verification Approach

Rosenfeld's method [3] combines two key ingredients:

1. An upper bound on the product of speeds in any counterexample (from [2])
2. A computational search to prove that certain primes must divide this product

Specifically, Rosenfeld proves:

Lemma 2 (Lemma 6 in [3])**.** *Let $k \geq 3$ such that the conjecture holds for $k-1$, and let p be a positive integer. If for all k-tuples $v_1, \ldots, v_k \in \{1, \ldots, \lfloor (k+1)p/2 \rfloor\} \setminus p\mathbb{N}$ satisfying certain GCD conditions, there exists t such that $v_1, \ldots, v_k$ do not cover all positions, then p divides the product of speeds in any counterexample.*

The verification problem reduces to checking, for each prime p, whether any k-tuple covers all positions $\{1, \ldots, \lfloor (k+1)p/2 \rfloor\}$ subject to constraints.

2.3 Rosenfeld's Implementation

Rosenfeld's C++ implementation [3] uses backtracking search with pruning techniques. The search explores k-tuples from $\{1, \ldots, \lfloor (k+1)p/2 \rfloor\} \setminus p\mathbb{N}$, checking whether they cover all positions. The GCD constraints from Lemma 2 are enforced as pruning conditions during search: for each prime divisor q of $(k+1)$, the code requires at most $k-2$ selected elements are divisible by q (implemented in the `extraVerif` function for $k+1 \in \{3, \ldots, 9\}$).

For k=7, the approach requires verifying 27 primes, with the hardest case (p=163) taking approximately 32 hours on a single core.

3 SAT-Based Formulation

3.1 Problem Encoding

We reformulate the verification as a Boolean satisfiability problem. Given k and p, let $Q = (k+1)p$ and $\text{maxM} = \lfloor Q/2 \rfloor$.

Variables. For each element $v \in \{1, \ldots, \text{maxM}\}$ with $v \not\equiv 0 \pmod{p}$, we introduce a Boolean variable x_v indicating whether v is selected in the k-tuple.

Coverage Definition. Following Rosenfeld's implementation (line 148 of main.cpp [3]), we precompute for each velocity v and time index t whether v covers position t:

$$v \text{ covers } t \iff (vt \bmod Q)\cdot(k+1) < Q \quad \text{or} \quad (Q-(vt \bmod Q))\cdot(k+1) < Q$$

This condition encodes $\|vt/Q\| < 1/(k+1)$, where $\|\cdot\|$ denotes distance to the nearest integer. We base our coverage definition on Rosenfeld's C++ implementation and his Lemma 6, which use the condition $\|\cdot\| < 1/(k+1)$. We note that the descriptive text accompanying the reformulated lemma in Rosenfeld's paper contains a typographical error stating this as $1/(k-1)$; the code and original lemma correctly use $1/(k+1)$. The coverage relation is computed for all relevant (v, t) pairs using the same bitset representation as Rosenfeld's code.

Constraints. We encode the following as CNF clauses:

1. **Cardinality:** Exactly k elements selected, using Sinz sequential counter encoding [4]
2. **Coverage:** For each position that must be covered, at least one selected element must cover it:

$$\bigvee_{v : v \text{ covers } t} x_v$$

3. **GCD constraints:** For each prime q dividing $(k+1)$, at most $k-2$ selected elements are divisible by q. Since candidates already exclude multiples of p, we only constrain prime divisors of $(k+1)$. This encodes the condition from Lemma 2 that $\gcd(S \cup \{(k+1)p\}) = 1$ for all $(k-1)$-subsets S.

The CNF is UNSATISFIABLE if and only if no k-tuple satisfying all constraints covers all positions, proving that Lemma 2 applies.

3.2 Correctness of SAT Encoding

Proposition 3. *Our SAT encoding is logically equivalent to the verification condition from Lemma 2: for any (k, p) pair, the CNF is UNSATISFIABLE if and only if Rosenfeld's verification succeeds (no k-tuple satisfying the constraints covers all positions).*

Proof. The SAT encoding searches for a k-tuple $\{v_1, \ldots, v_k\}$ that:

1. Contains exactly k elements from $\{1, \ldots, \text{maxM}\} \setminus p\mathbb{N}$
2. Covers all positions using the same coverage definition as Rosenfeld's implementation: for each position, some selected v_i satisfies $\|v_i t/Q\| < 1/(k+1)$
3. Satisfies the GCD constraint: for all primes $q \mid (k+1)$, at most $k-2$ of the v_i are divisible by q

By the argument in [3], condition (3) is equivalent to requiring $\gcd(S \cup \{Q\}) = 1$ for all $(k-1)$-subsets S of the selected tuple. This is precisely the GCD condition enforced by Rosenfeld's `extraVerif` function and required by Lemma 2.

The CNF is UNSATISFIABLE if and only if no tuple satisfying conditions (1)–(3) exists. Therefore, UNSAT confirms that Lemma 2 applies. Since both approaches check existence of a cover satisfying identical mathematical constraints, they produce the same verification outcome for each (k, p).

For validation, we verified identical results (UNSAT) on all 88 instances from $k \in \{3, \ldots, 7\}$ across all required primes, confirming the encoding captures the same combinatorial condition as Rosenfeld's verification. □

3.3 Implementation Details

Our implementation consists of:

1. **CNF Generator** (C++): Produces DIMACS CNF format
2. **SAT Solver**: We use Kissat [5], a state-of-the-art SAT solver

The generator uses the Sinz encoding [4] for cardinality constraints, which produces $O(kn)$ auxiliary variables and clauses for n candidate elements.

4 Results

4.1 Verification Results

We successfully verified the Lonely Runner Conjecture for k=4 through k=12:

Table 1: Complete verification results. All primes verified UNSAT (lemma applies).

k	Runners	Primes	Time	Previously Known
4	5	6	0.90s	Yes
5	6	12	1.43s	Yes
6	7	19	2.14s	Yes
7	8	27	3.00s	Rosenfeld [3]
8	**9**	**44**	**5.02s**	**New**
9	**10**	**86**	**12.62s**	**New**
10	**11**	**105**	**21.68s**	**New**
11	**12**	**158**	**75.32s**	**New**
12	**13**	**190**	**155.36s**	**New**

All computations performed on an Apple M4 Max (16 performance cores, 4 efficiency cores). Times include CNF generation and SAT solving for all primes, executed in parallel across available cores.

4.2 Performance Comparison

We compare our approach with Rosenfeld's backtracking method:

Table 2: Performance comparison. Backtracking time for k=7, p=163 from [3].

Case	Backtracking	SAT Solver	Speedup
k=7, p=163	32 hours	0.02s	5,760,000×
k=7 (all 27)	>32 hours*	3.00s	>38,400×
k=8 (all 44)	—	5.02s	—
k=11 (all 158)	—	75.32s	—
k=12 (all 190)	—	155.36s	—

*Our total time (3.00s) for all 27 primes is faster than Rosenfeld's single hardest instance (32h).

The speedup stems from:

1. Efficient conflict-driven clause learning in modern SAT solvers
2. Optimized unit propagation and learned clause management
3. Decades of SAT solver development [5]

4.3 Detailed Results for New Cases

For k=8 through k=12, we verified UNSAT for all required primes. The prime lists were determined using the method from [3]: select primes such that their product exceeds the upper bound from [2].

k=8 (44 primes): 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251.

k=9 (86 primes): Extends to p=503.

k=10 (105 primes): Extends to p=631.

k=11 (158 primes): Extends to p=997.

k=12 (190 primes): Extends to p=1193.

All cases verified UNSAT, confirming no counterexamples exist.

5 Conclusion

We have extended the computational verification of the Lonely Runner Conjecture from 8 runners to 13 runners using a SAT-based approach. Our method provides speedups of 5–6 orders of magnitude compared to backtracking, making previously infeasible cases computationally tractable.

Main Results:

- First proof of Lonely Runner Conjecture for 9, 10, 11, 12, and 13 runners
- SAT encoding that captures the same verification problem as Rosenfeld's Lemma 6 approach
- Demonstration that SAT solving is highly effective for this problem class

Future Work: The SAT approach appears scalable to k=13 and beyond. However, as k increases, the number of required primes grows, eventually limiting feasibility. Theoretical improvements to reduce the required prime set would enable further progress.

Acknowledgments

This work builds directly on the foundational approach by Rosenfeld [3] and the theoretical results by Malikiosis, Santos, and Schymura [2]. We thank the developers of the Kissat SAT solver [5].

References

[1] J. M. Wills. *Zwei Sätze über inhomogene diophantische Approximation von Irrationalzahlen*. Monatshefte für Mathematik, 71:263–269, 1967.

[2] R.-D. Malikiosis, F. Santos, and M. Schymura. *An explicit bound for the lonely runner problem*. arXiv:2406.19389, 2024.

[3] M. Rosenfeld. *The lonely runner conjecture holds for eight runners*. arXiv:2509.14111, 2025. Code: `https://gite.lirmm.fr/mrosenfeld/the-lonely-runner-conjecture`.

[4] C. Sinz. *Towards an optimal CNF encoding of Boolean cardinality constraints*. In Proceedings of CP 2005, pages 827–831, 2005.

[5] A. Biere, K. Fazekas, M. Fleury, and M. Heisinger. *CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020*. In Proceedings of SAT Competition 2020, pages 51–53, 2020.