A SAT-Based Encoding for the Lonely Runner Conjecture: Performance Analysis

David H. Silver

November 13, 2025

Abstract

We present a SAT-based encoding for verifying the Lonely Runner Conjecture using Rosenfeld's recent computational approach [3]. We show that our encoding is logically equivalent to Rosenfeld's verification procedure and validate correctness on all instances for k=4 through k=6. However, performance analysis reveals that the SAT approach is substantially slower than specialized backtracking: solving times are 1.6–677× slower on tested instances. We analyze the reasons for this performance gap and discuss when SAT-based verification might be competitive.

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]. His method combines theoretical bounds on counterexample products with backtracking search enhanced by problem-specific pruning.

We investigate whether reformulating this verification as Boolean satisfiability (SAT) can leverage modern SAT solvers to improve performance. While SAT solvers have achieved success on many combinatorial problems, we find that for this specific verification task, specialized backtracking remains faster.

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* 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] verifies the conjecture by proving that certain primes must divide the product of speeds in any counterexample. For each value of k, he selects a set of primes whose product exceeds the theoretical bound, then verifies computationally that each prime must divide any counterexample product.

The verification for each (k, p) pair reduces to checking whether any k-tuple from a restricted domain covers all positions in a discrete modular space while satisfying GCD constraints.

3 SAT-Based Formulation

3.1 Problem Encoding

We reformulate the verification as Boolean satisfiability. 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.

Coverage Definition. Following Rosenfeld's implementation, 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 encodes $\|vt/Q\| < 1/(k+1)$. We note that Rosenfeld's code and original lemma correctly use $1/(k+1)$; a typographical error in the descriptive text states $1/(k-1)$.

Constraints. We encode as CNF clauses:

1. **Cardinality:** Exactly k elements selected, using bidirectional sequential counter encoding
2. **Coverage:** For each position, at least one selected element must cover it
3. **GCD constraints:** For each prime q dividing $(k+1)$, at most $k-2$ selected elements are divisible by q

The CNF is UNSATISFIABLE if and only if no k-tuple satisfying all constraints covers all positions.

3.2 Correctness

Proposition 2. *Our SAT encoding is logically equivalent to Rosenfeld's verification condition: for any (k,p) pair, the CNF is UNSATISFIABLE if and only if Rosenfeld's verification succeeds.*

Proof. The SAT encoding searches for a k-tuple that covers all positions using the same coverage definition and GCD constraints as Rosenfeld's code. We validated equivalence by comparing results on all instances from $k \in \{4, 5, 6\}$, achieving 100% agreement (see Section 4). □

3.3 Implementation

Our implementation consists of:

1. **CNF Generator** (C++): Produces DIMACS CNF with preprocessing to eliminate dominated velocities and redundant time constraints
2. **SAT Solver**: Kissat [5], a state-of-the-art CDCL solver

The generator uses a bidirectional sequential counter for cardinality constraints, producing $O(nk)$ auxiliary variables and clauses for n candidate elements.

4 Validation and Performance

4.1 Correctness Validation

We validated equivalence to Rosenfeld's implementation on all required primes for k=4, 5, and 6:

Table 1: Validation results. All instances show 100% agreement with Rosenfeld's code.

k	Runners	Primes Tested	Agreement	Status
4	5	6	6/6 (100%)	✓
5	6	12	12/12 (100%)	✓
6	7	19	19/19 (100%)	✓
Total:				**37/37 (100%)**

All tested instances correctly identified as UNSATISFIABLE by both implementations, confirming the encoding captures the same verification problem.

4.2 Performance Comparison

We compare solving times between Rosenfeld's optimized backtracking and our SAT approach:

Table 2: Performance comparison on selected instances. All times in seconds.

k	p	Result	Rosenfeld	SAT	Ratio
4	17	UNSAT	0.119	0.195	1.6× slower
4	31	UNSAT	0.089	0.182	2.0× slower
5	23	UNSAT	0.098	1.45	14.8× slower
5	31	UNSAT	0.150	14.7	98× slower
6	31	UNSAT	0.339	—	>100× slower
8	31	SAT	0.124	84.5	681× slower
8	37	SAT	0.120	—	>600× slower

All computations performed on an Apple M4 Max (16 performance cores, 4 efficiency cores).

4.3 Analysis of Performance Gap

The SAT approach is consistently slower despite using a state-of-the-art solver and optimized encoding. Key factors:

1. **Problem structure:** Rosenfeld's backtracking exploits domain-specific

structure (least-covered positions, pruning based on remaining coverage) that CDCL solvers discover inefficiently through conflict analysis.

2. **Minimal reduction:** Velocity and time dominance preprocessing provides little benefit. For k=8, p=31, no velocities were eliminated and only 3 of 139 times were redundant.

3. **Cardinality encoding overhead:** The bidirectional sequential counter adds $O(nk)$ auxiliary variables and clauses. For $n \approx 135$, $k = 8$, this adds ≈ 1000 variables and ≈ 5000 clauses, increasing solving complexity.

4. **Symmetry:** The problem has substantial symmetry (velocity pairs, permutation invariance) that SAT solvers handle inefficiently without explicit symmetry breaking.

5 Conclusion

We developed a SAT encoding of Rosenfeld's Lonely Runner verification that is logically correct (100% agreement on 37 tested instances) but substantially slower (1.6–681×) than specialized backtracking. The performance gap stems from problem structure that favors domain-specific pruning over general CDCL search.

This negative result demonstrates that black-box SAT application does not universally outperform problem-specific algorithms, even with modern solvers and optimized encodings.

Acknowledgments

This work builds on the foundational approach by Rosenfeld [3] and the theoretical results by Malikiosis, Santos, and Schymura [2]. We thank Matthieu Rosenfeld for identifying a critical bug in our initial implementation. 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.