Solving Combinatorial Problems Using Satisfiability

Aaron Barnoff

MSc Thesis Defense

Department of Computer Science
University of Windsor

Co-Supervisor: Dr. Curtis Bright
Co-Supervisor: Dr. Ahmad Biniaz
Internal Reader: Dr. Yung H. Tsin
External Reader: Dr. Adrian Zahariuc
Defense Chair: Dr. Shaon Shuvo

July 31, 2026

Boolean Satisfiability

Boolean Satisfiability Problem (SAT)

Given a Boolean formula, does there exist an assignment of truth values that makes the formula true?

Example: $\alpha \wedge \neg\beta$

- Assign $\alpha \equiv \mathbf{T}$ and $\beta \equiv \mathbf{F}$
- Then $\alpha \wedge \neg\beta \equiv \mathbf{T} \wedge \neg\mathbf{F} \equiv \mathbf{T} \wedge \mathbf{T} \equiv \mathbf{T}$

Notation:

- AND($\wedge$): $p \wedge q$
- OR($\vee$): $p \vee q$
- NOT($\neg$): $\neg p$
- Implication($\rightarrow$): $p \rightarrow q \equiv \neg p \vee q$

Boolean Satisfiability

SAT was the first problem to be proven NP-complete.

SAT turns a combinatorial search problem into a logical formula.

- The solutions can be searched for automatically by a SAT solver.

SAT solvers can search spaces that are too large for naive enumeration.

- They have been used to solve many difficult problems, e.g. Lam's problem and Boolean Pythagorean triples.

Boolean Satisfiability

In this thesis, SAT is used to solve two problems:

- Topic 1: Searching for lattice walks avoiding collinear points.
- Topic 2: Finding mutually orthogonal Latin squares.

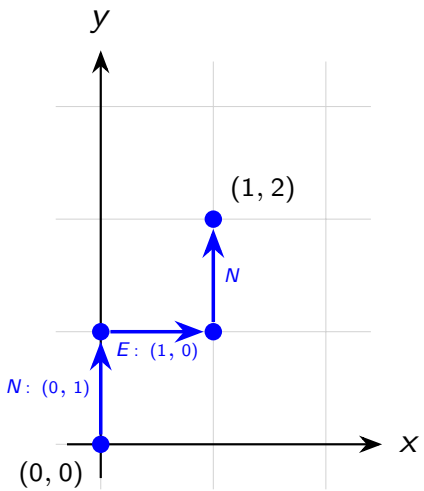
Topic 1: Gerver-Ramsey Collinearity Problem

North–East Walk

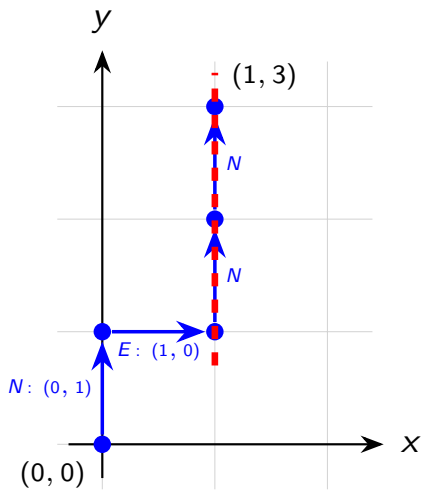
A north–east walk is a lattice path in which each step is one unit north, $N\colon (0,1)$, or one unit east, $E\colon (1,0)$.

Gerver-Ramsey Collinearity Problem

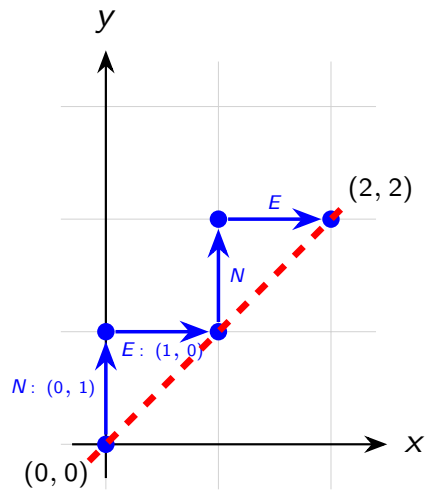
How long can a north–east walk be before it must contain k collinear points?



Longest path avoiding 3 collinear points



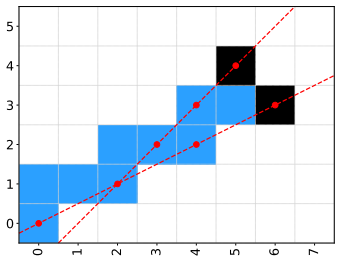
3 collinear points (north)



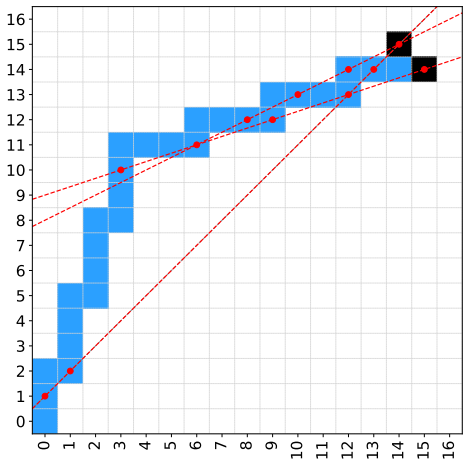
3 collinear points (east)

Topic 1: Gerver-Ramsey Collinearity Problem

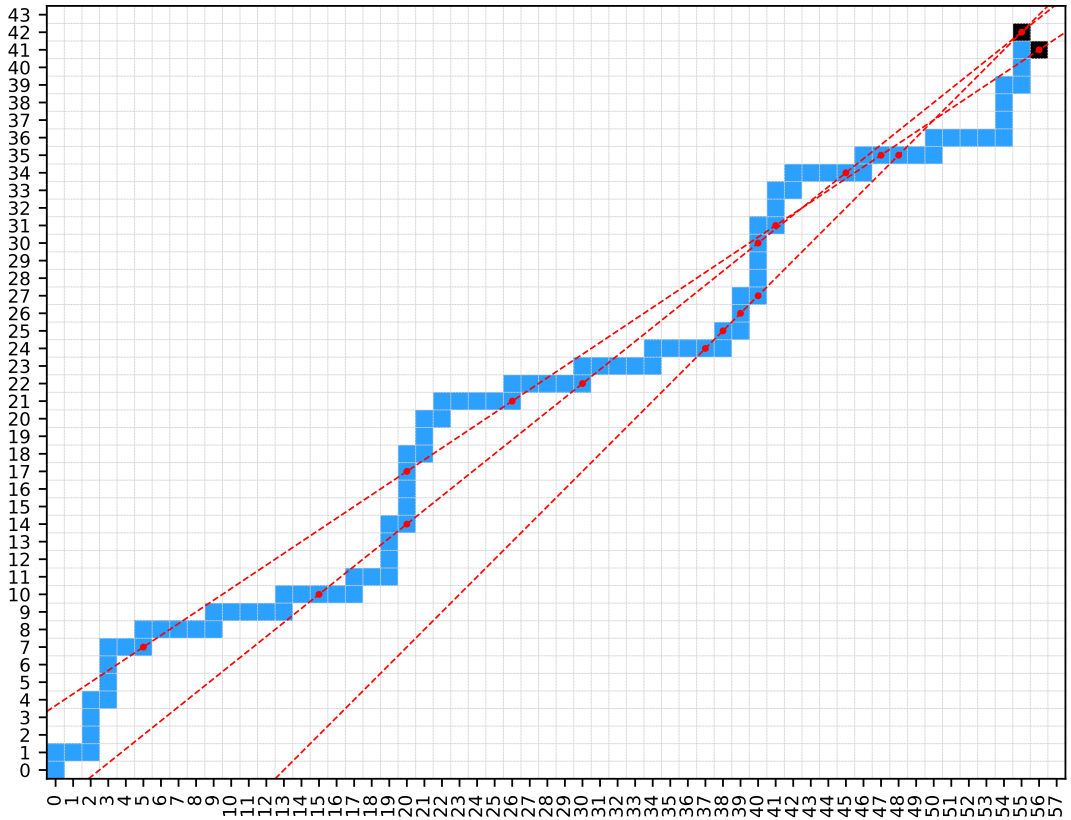
Let $a(k)$ denote the minimum path length needed for **all** north–east walks to contain k collinear points.



$a(4) = 9$ steps.



$a(5) = 29$ steps.



$a(6) = 97$ steps.

The blue paths are the longest possible paths avoiding k collinear points.

Topic 1: Gerver–Ramsey Collinearity Problem

$a(k)$: minimal path length needed for all walks to contain k collinear points.

(1979): Gerver and Ramsey gave extremely loose bounds for $a(k)$:

$$\underbrace{\left\lfloor \left(32(k-1)^{2\log_2(k-1)-7}\right)^{1/18} \right\rfloor + 2}_{\textbf{Lower Bound}} \leq a(k) \leq \underbrace{(k-1)2^{2^{13}(k-1)^4}}_{\textbf{Upper Bound}}$$

Superpolynomial: $2^{\Theta((\log_2 k)^2)}$

Superexponential: $2^{\Theta(k^4)}$

For $k = 6$, the exact value is $a(6) = 97$, but the bounds give:

$$\underbrace{2}_{\text{Lower bound}} \leq \boxed{a(6) = 97} \leq \underbrace{5 \cdot 2^{5,120,000}}_{\substack{\text{Upper bound} \\ \text{1,541,275 digits}}}$$

Topic 1: Gerver-Ramsey Collinearity Problem

$a(k)$: minimal path length needed for all walks to contain k collinear points.

In 2013, Jeffrey Shallit determined $a(k)$ up to $k = 6$.

- He used a custom backtracking search written in APL.

Shallit also determined that $a(7) \geq 261$ steps.

- He did this by finding a 260-step walk that avoids 7 collinear points.

To push past 260 steps, we encode this problem into **SAT**.

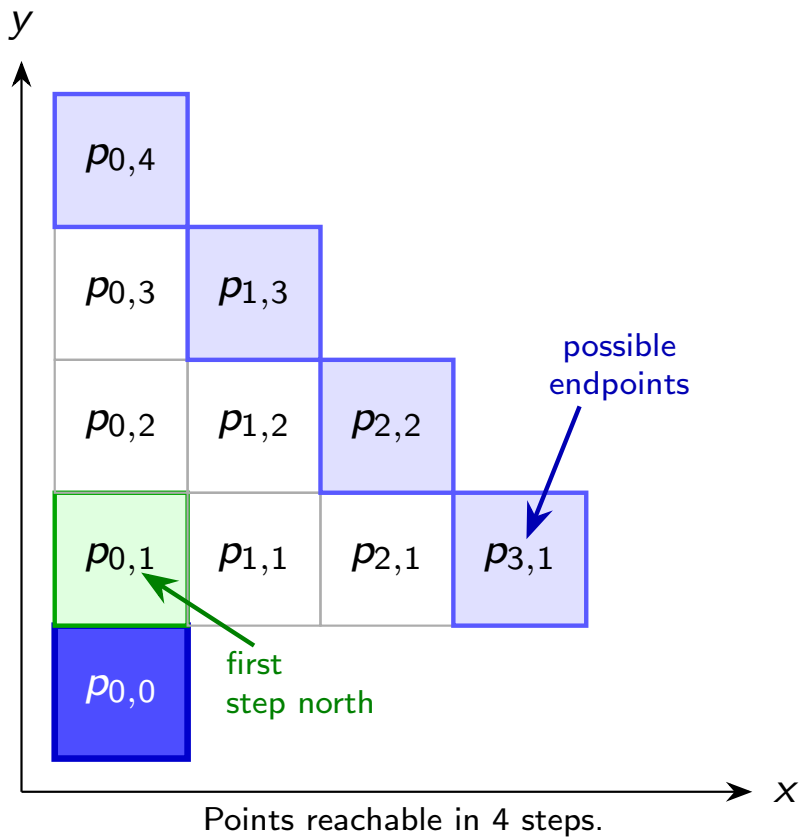
Topic 1: Encoding North–East Walks

Fix in advance:

- The path length m to be searched.
- The number of collinear points k to avoid.

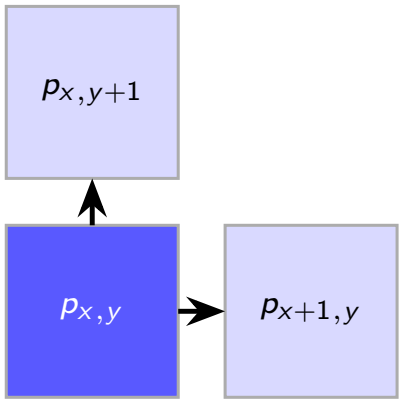
Let the variable $p_{x,y}$ be true *iff* the point (x, y) is on the path:

- The path begins at $p_{0,0}$.
- **Symmetry break**: $p_{0,1} \equiv \mathbf{T}$.



Topic 1: Encoding North–East Walks

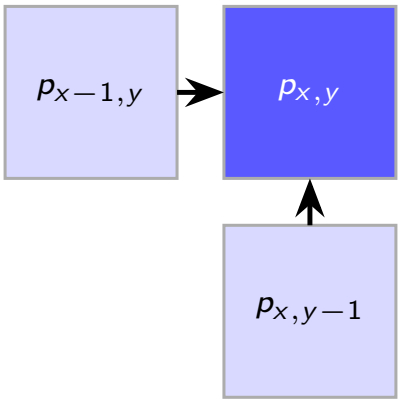
Forward connectivity



$$p_{x,y} \to (p_{x+1,y} \vee p_{x,y+1})$$

Every non-final point must continue north or east.

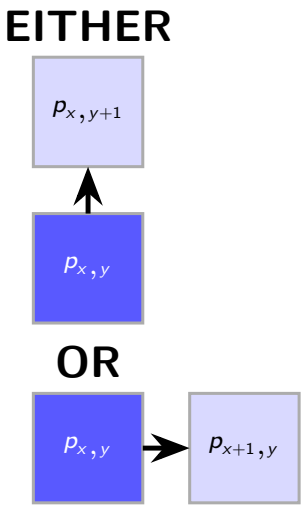
Reverse connectivity



$$p_{x,y} \to (p_{x-1,y} \vee p_{x,y-1})$$

Every non-origin point must have a predecessor.

No branching



$$\neg (p_{x+1,y} \wedge p_{x,y+1})$$

A non-final point cannot continue in both directions.

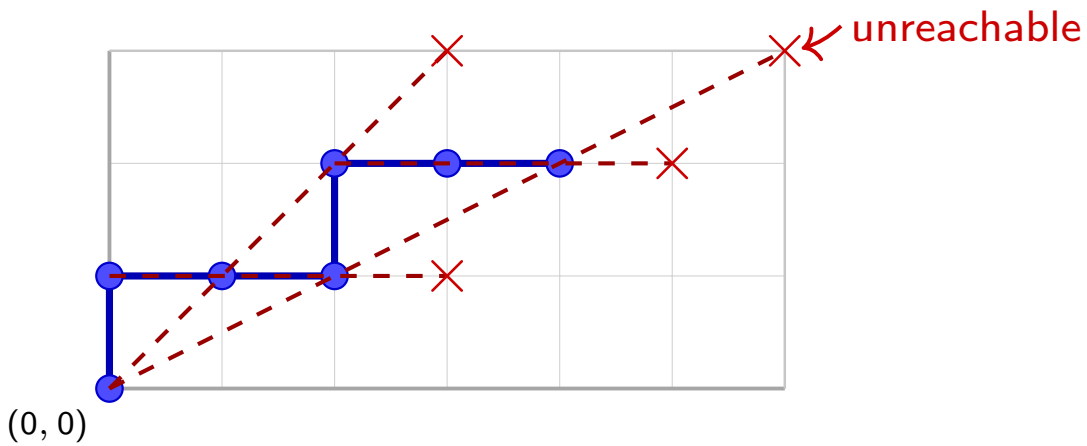
Topic 1: Encoding Non-Collinearity

Non-Collinearity Cardinality Constraint

For every lattice line L, at most $k-1$ of its point variables may be true:

$$\sum_{(x,y)\in L} p_{x,y} \leq k-1.$$

Example: avoiding $k = 4$ collinear points



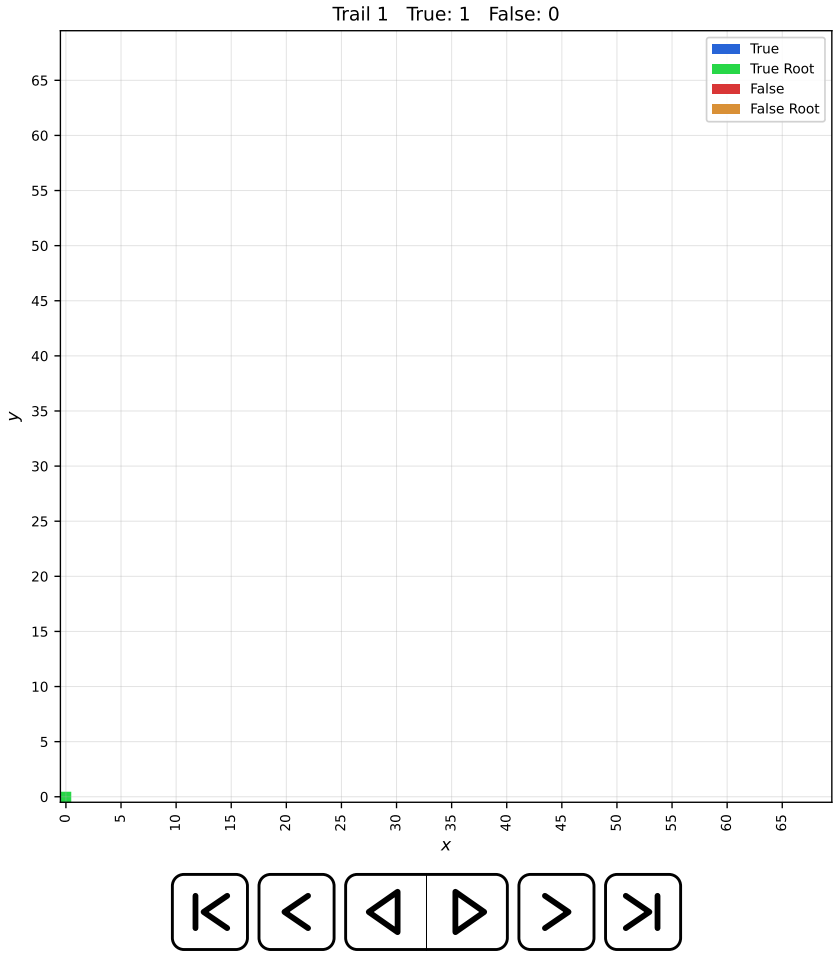
When $k = 4$:

$$\sum_{(x,y)\in L} p_{x,y} \leq 3.$$

Each line with at least k reachable points is translated into a logical formula using a cardinality encoding.

Topic 1: Solver Example

Solving a 69-step path avoiding 6 collinear points:

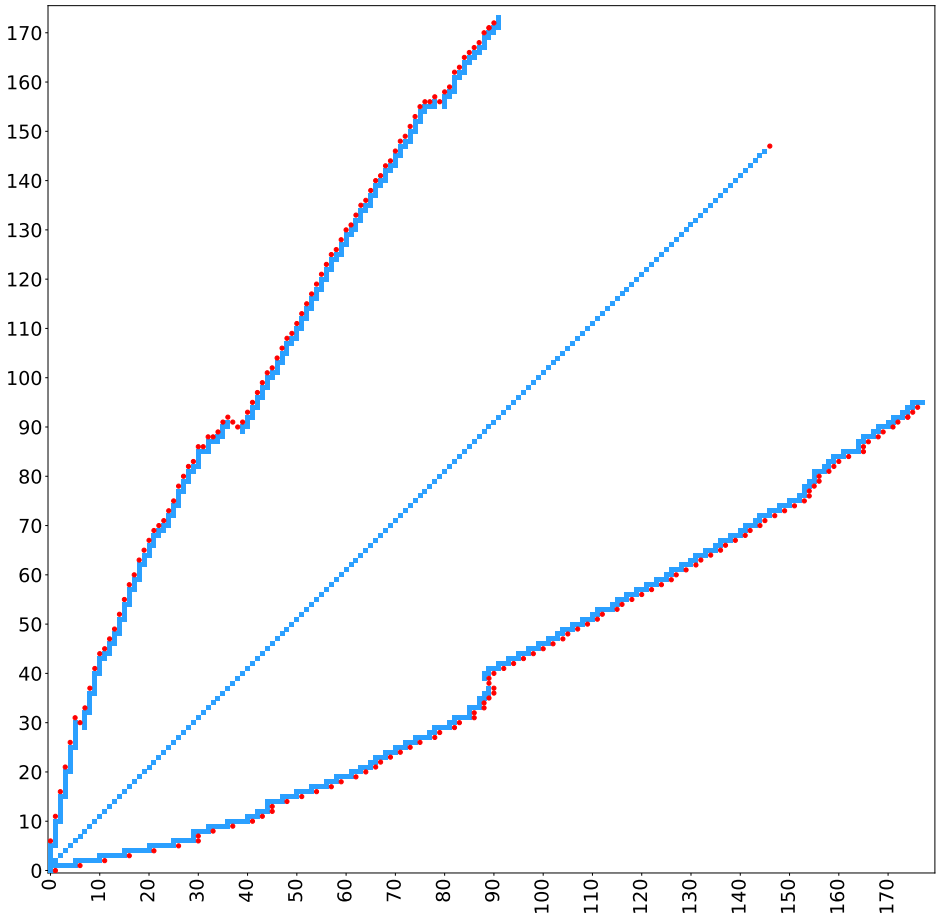


- Path: 2485 variables, 7316 clauses
- Cardinality: 3095 lines $\rightarrow$ 43,280 variables, 93,853 clauses

Topic 1: Computing the reachability bounds for $k = 7$

We computed reachability bounds for walks up to 267 steps:

- The upper and lower bounds took over 1 CPU year.
- The final unreachable diagonal point took 18 CPU years (6306 cores).

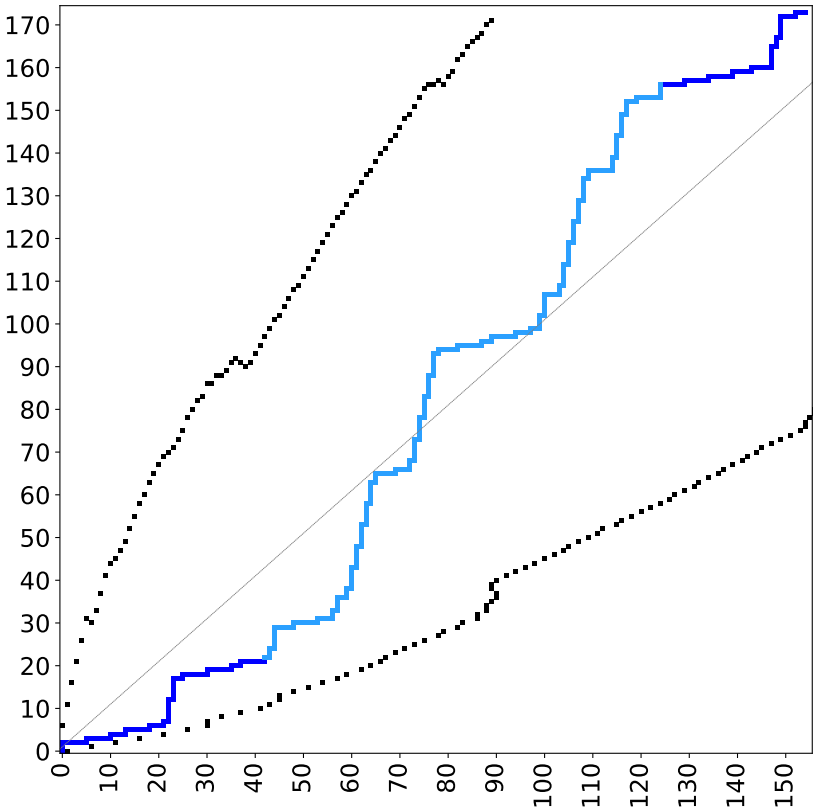


Reachability bounds for walks avoiding 7 collinear points.

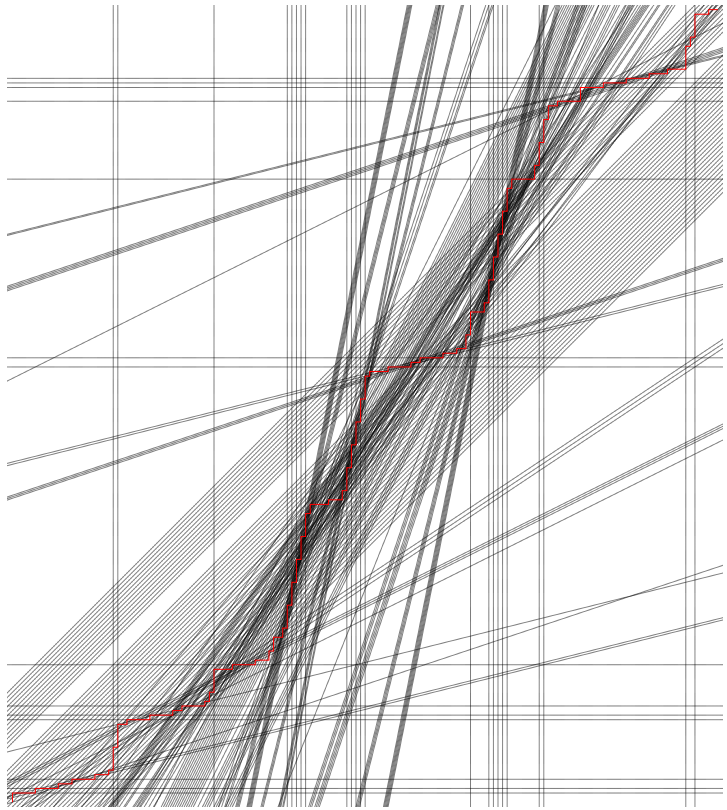
Topic 1: Results for $a(7)$

We used a SAT parallelization technique to search for $a(7)$:

- We found two 322-step paths in 346 CPU days using 133 cores.
- Extended to 327 steps in 3 hours by encoding a shared subpath (light blue).



A north–east walk with 327 steps.



All lines with 6 collinear points.

Topic 2: Finding Mutually Orthogonal Latin Squares

Latin Square

A Latin square of order n is an $n \times n$ array with n symbols, where each symbol occurs exactly once in each row and column.

0	1	2
1	2	0
2	0	1

Square P

Mutually Orthogonal Latin Squares

Mutually Orthogonal Latin Squares (MOLS)

Two Latin squares are orthogonal mates if every ordered pair of symbols occurs exactly once when they are overlaid.

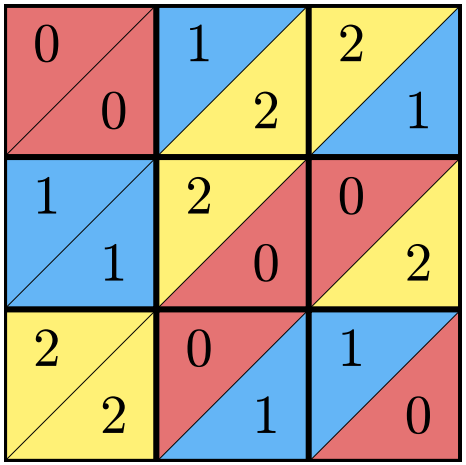
- A set of k mutually orthogonal Latin squares of order n is called a k-MOLS(n).

0	1	2
1	2	0
2	0	1

Square P

0	2	1
1	0	2
2	1	0

Square R (mate)



2-MOLS(3)

Topic 2: Finding Mutually Orthogonal Latin Squares

In 1959, Parker et al. proved 2-MOLS(n) exist for all $n \neq 2, 6$.

Order 10 is the first unresolved case for the maximum number of MOLS.

- We know the bound is between 2 and 6.

Whether a **3-MOLS(10)** exists is a longstanding open problem.

- One way to search for a 3-MOLS(10) is by trying to extend a 2-MOLS(10).

Transversals

Transversal

A transversal of a Latin square of order n is a set of n cells containing one cell from each row and column, and one occurrence of each symbol.

0	1	2
1	2	0
2	0	1

Square P

0	1	2
1	2	0
2	0	1

Transversal T_0

0	1	2
1	2	0
2	0	1

Transversal T_1

0	1	2
1	2	0
2	0	1

Transversal T_2

Euler's Observation

Euler's Observation

Euler's observation: A Latin square has an orthogonal mate if and only if it contains n disjoint transversals.

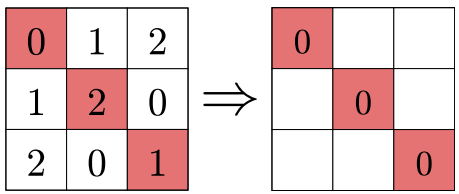
- To build a mate, assign one distinct symbol to each transversal:

0	1	2
1	2	0
2	0	1

Square P

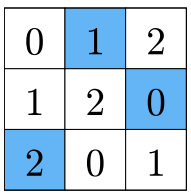
0	1	2
2	0	1
1	2	0

Square R (mate)

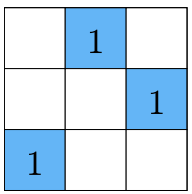


T_0

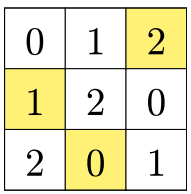
Label 0



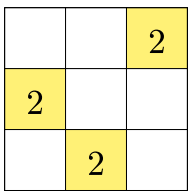
T_1



Label 1



T_2



Label 2

The Euler–Parker Algorithm

(1959): Parker used Euler's observation to create an algorithm:

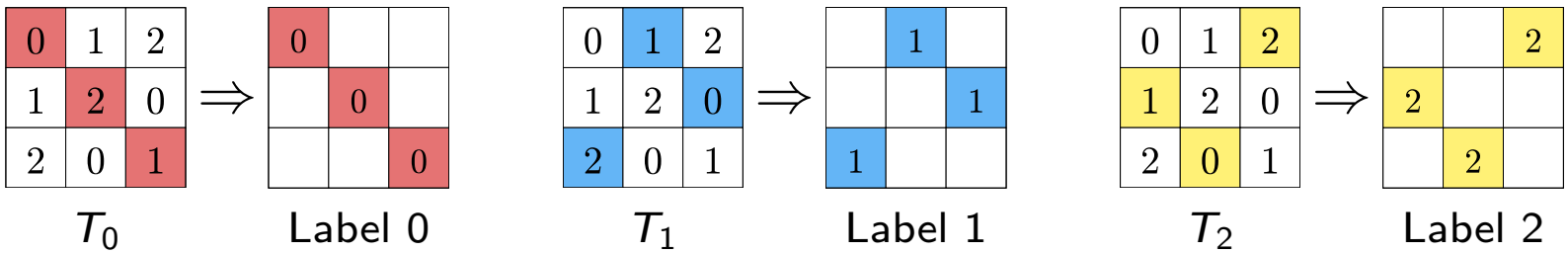
- Take as input a Latin square.
- Step 1: Find **all** transversals of the square.
- Step 2: Select n **disjoint** transversals from that set.
- Step 3: Relabel each transversal with a distinct symbol.

0	1	2
1	2	0
2	0	1

Square P

0	1	2
2	0	1
1	2	0

Square R (mate)



Solving 2-MOLS(n) with SAT

SAT solvers have previously proved useful at finding 2-MOLS(n).

The SAT encoding is a declarative version of Euler–Parker:

- 1. Continually search for P squares.
- 2. Try find n disjoint transversals of P.

Problem: The solver can only see n transversals at a time.

- It can't look at the set of **all** transversals and pick out n disjoint ones.

Result: The solver will likely move to a new P square before finding the right set of transversals.

Hybrid SAT + Euler–Parker Idea

Hybrid Solver

Augment the SAT solver with the Euler–Parker algorithm.

Hybrid method:

- 1. SAT solver still searches for P squares and n disjoint transversals.
- 2. When a full P square is found, run Euler–Parker:
  - Step 1: Find **all** transversals of the square.
  - Step 2: Select n **disjoint** transversals from that set.
- 3. If no mate exists, block the square and continue the SAT search.

Hybrid SAT Results

We tested the method on a very difficult 2-MOLS(10) problem.

(1999): Myrvold considered a special case of 3-MOLS(10):

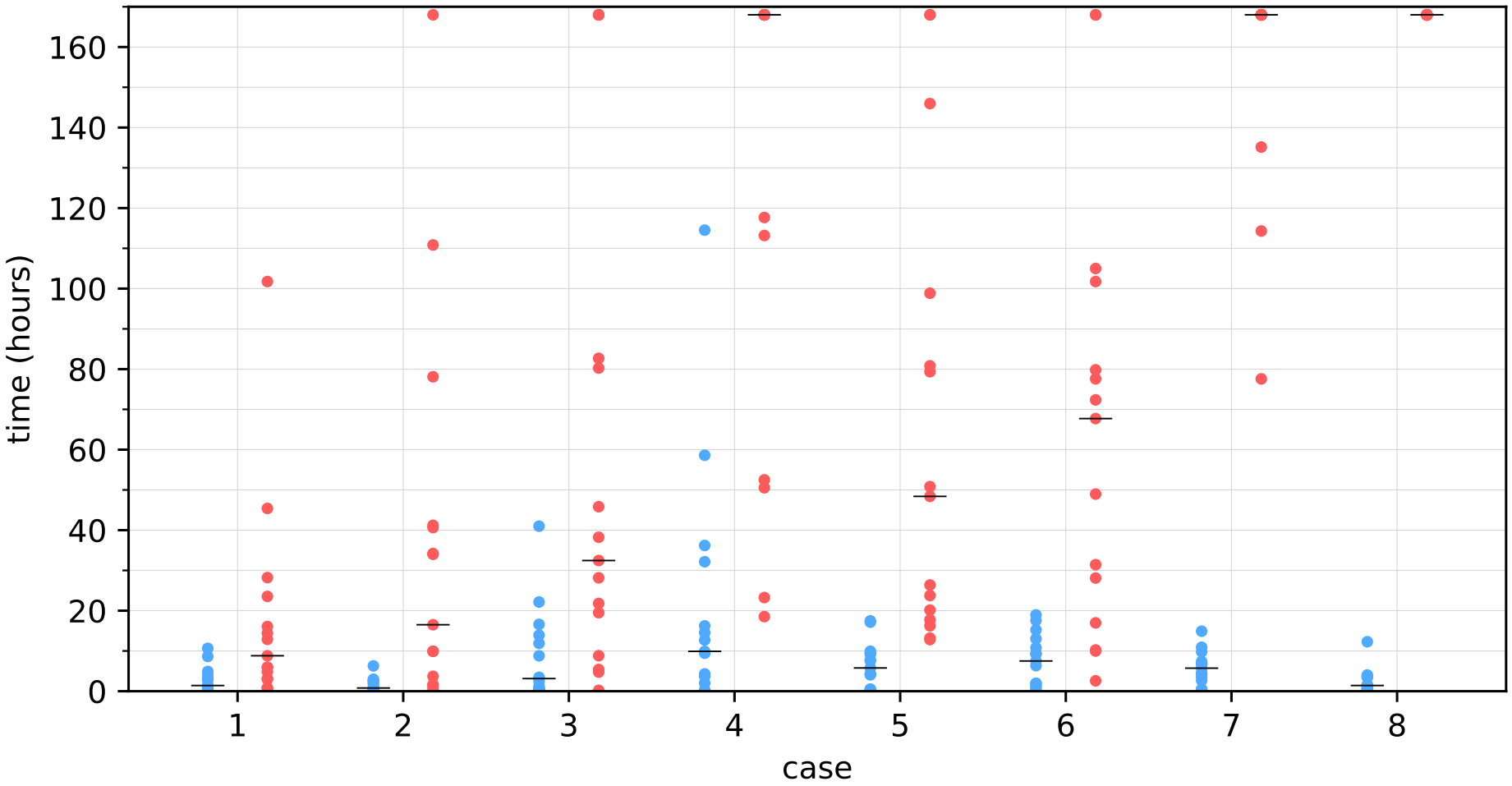
- A 2-MOLS(10) extended by a third mate containing a 4×4 Latin subsquare.
- The subsquare induces symbol constraints in the 2-MOLS(10).

Myrvold could not disprove the existence of eight 2-MOLS(10) cases.

- (2025): Bright et al. used SAT to find 2-MOLS(10) for each case.

Hybrid SAT Results

We reproduced Bright et al.'s results using the hybrid solver:



Hybrid (blue) vs pure SAT (red); 15 random seeds, 7-day timeout.

Conclusion

Our work on the Gerver-Ramsey problem has been published in *Advances in Applied Mathematics* (AAM).

Our work on the MOLS problem will appear in the *Proceedings of the 2026 International Workshop on Satisfiability Checking and Symbolic Computation*, Oldenburg, Germany.

Thank you for listening!

Solving 2-MOLS(n)

Solving unconstrained 2-MOLS(n) is not challenging.

- Pick a Latin square at random, use Euler–Parker on it.

But searching for 2-MOLS(n) of a special form is more difficult.

- e.g. mutually orthogonal diagonal latin squares:

0	1	2	3	4
2	3	4	0	1
4	0	1	2	3
1	2	3	4	0
3	4	0	1	2

0	1	2	3	4
3	4	0	1	2
1	2	3	4	0
4	0	1	2	3
2	3	4	0	1

Extra constraint: Symbols on the main line and antidiagonal line are distinct.

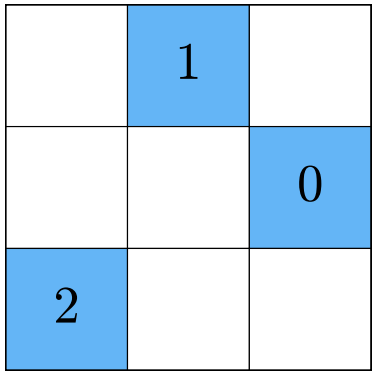
SAT solvers have proved useful in finding constrained 2-MOLS(n).

Solving 2-MOLS(n) with SAT

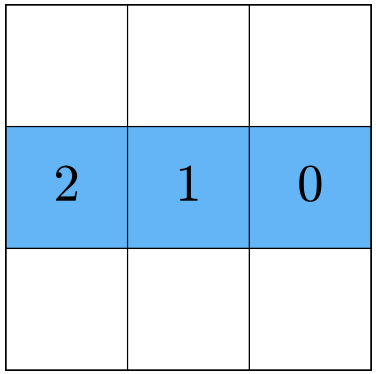
The SAT encoding asserts the existence of n disjoint transversals in P.

The encoding asserts three Latin squares: P, Q, R.

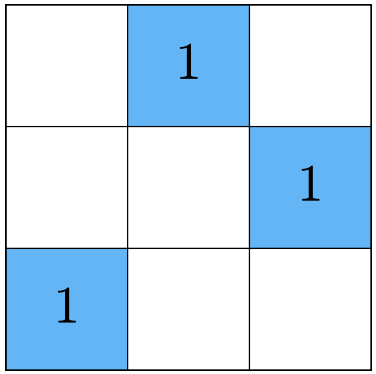
- P and R are the orthogonal mates.
- Q stores the transversals of P as row vectors.
- The row index of Q becomes the transversal's symbol in R.



Square P
transversal T_1



Witness Q
stored in row 1



Square R
relabelled to
symbol 1

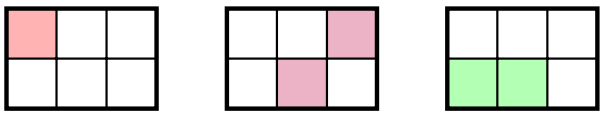
Exact Cover

Exact Cover Problem

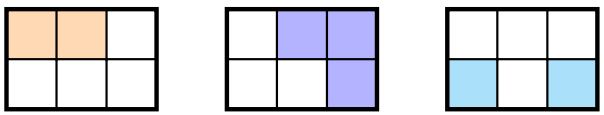
Given a set X and a collection of subsets of X, can we select subsets so that each element of X appears exactly once?

Tiling a 2×3 grid with no gaps or overlaps:

Collection of Subsets

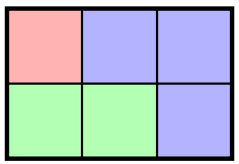


A B C



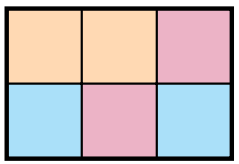
D E F

Exact cover solutions



Solution 1:

A, E, C



Solution 2:

D, B, F

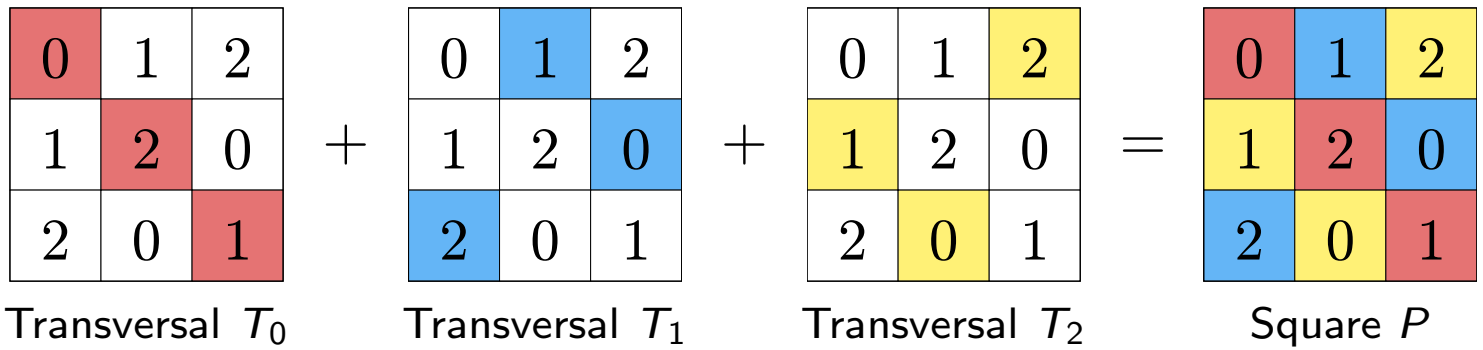
Euler–Parker as Exact Cover

Euler–Parker can be viewed as two exact cover problems:

- Problem 1: Finding transversals
- Problem 2: Combining transversals

Problem 2: Combining transversals

Select n transversals, such that **each cell** in P is covered exactly once.

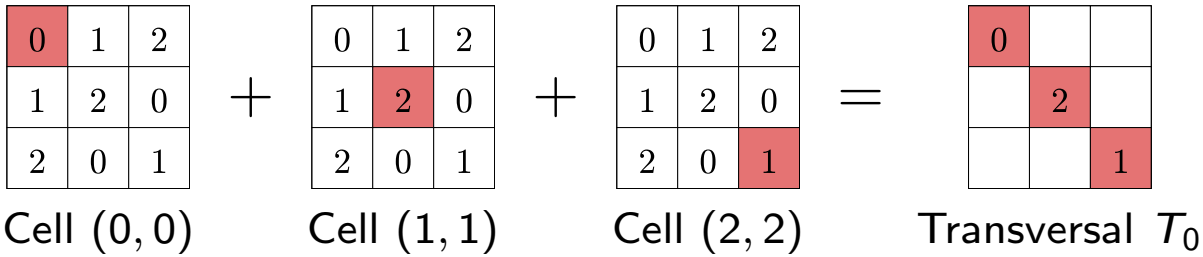


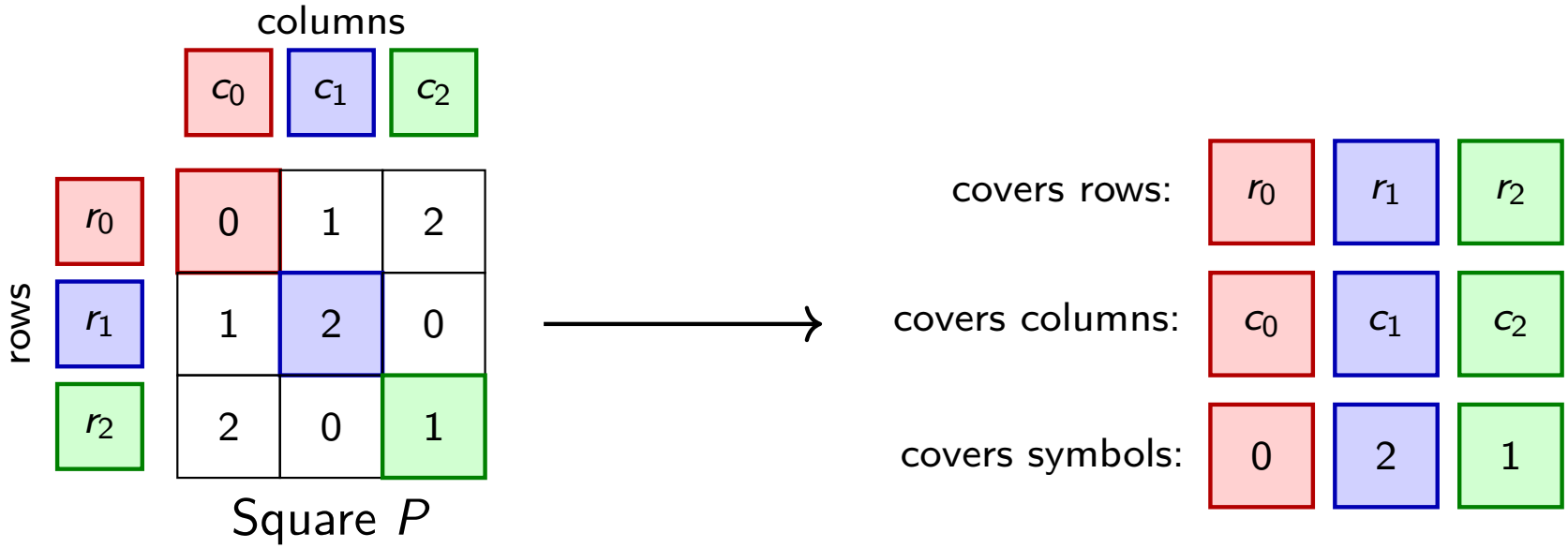
Transversal T_0 Transversal T_1 Transversal T_2 Square P

Euler–Parker as Exact Cover

Problem 1: Finding transversals

Select n cells, such that **each row**, **each column**, and **each symbol** in P is covered exactly once.





Hybrid SAT Results

We first tested the hybrid method on unconstrained 2-MOLS(n):

- Easy problem, simply looking for any pair of mates.
- Results show pure-SAT does not exploit Euler–Parker.

Order n	8	9	10	11	**12**	13	14	15
Hybrid	0.26	0.42	0.78	1.20	**2.23**	107.26	85.42	316.96
Pure SAT	0.09	168.19	3486.03	39 383.54	**timeout**	timeout	timeout	timeout

Table: Median times of 15 random seeds, with a 4-day timeout.