Satisfiability Solving and Lattice Reduction for Integer Factorization

Curtis Bright
`curtisbright.com`

University of Windsor

April 4, 2025

University of Windsor School of Computer Science Colloquium

SATisfiability

Formulae in Boolean logic consist of expressions formed with true/false variables connected with logical operators such as

$$\wedge \text{ (and)}, \vee \text{ (or)}, \neg \text{ (not)}, \oplus \text{ (xor)}, \leftrightarrow \text{ (iff)}.$$

For example:

$$(x \vee y) \wedge (\neg x \leftrightarrow z)$$

SAT: Given a Boolean logic expression, can it can be made true?

SATisfiability

Formulae in Boolean logic consist of expressions formed with true/false variables connected with logical operators such as

$$\wedge \text{ (and)}, \vee \text{ (or)}, \neg \text{ (not)}, \oplus \text{ (xor)}, \leftrightarrow \text{ (iff)}.$$

For example:

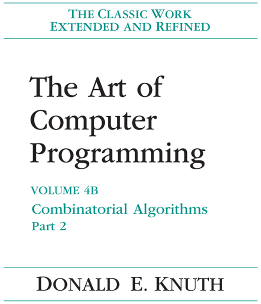
$$(x \vee y) \wedge (\neg x \leftrightarrow z)$$

SAT: Given a Boolean logic expression, can it can be made true?

The above example is satisfiable (take $x = y = \text{true}$, $z = \text{false}$).

SAT Solving

Donald Knuth's *The Art of Computer Programming Vol. 4B* (2022) is over 700 pages and half of it is devoted to the art of solving the SAT problem.

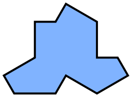


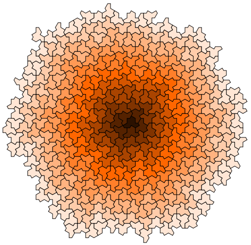
Despite having no provably fast algorithms, "SAT solvers" can be surprisingly effective and can be used solve a variety of problems seemingly unrelated to Boolean logic, like Sudoku or graph colouring.[1]

[1] Bright, Gerhard, Kotsireas, Ganesh. Effective Problem Solving Using SAT Solvers. *Maple Conference 2019.*

A SAT Success Story 1/4

A longstanding problem was to find a single shape that tiles the plane but only *aperiodically*. This was accomplished in 2022:[2]


the "hat"

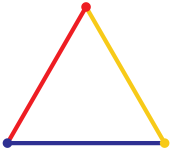


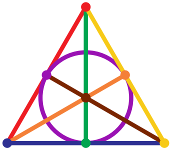
A SAT solver was used to find hat tilings containing at least 16 "layers" around a central hat. The tilings generated by the SAT solver helped produce a proof the hat tiles the plane.

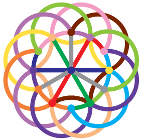
[2] Smith, Myers, Kaplan, Goodman-Strauss. An aperiodic monotile. Combinatorial theory 2024.

A SAT Success Story 2/4

In a *projective plane*, every pair of lines meet at a unique point and every pair of points define a unique line. In the *finite* case, when every line has n points, the plane is said to have *order* $n - 1$.


order 1


order 2

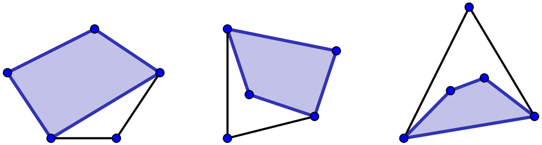

order 3

Projective planes exist for every prime power order, and do not exist in order 6. I used a SAT solver and 15,000 CPU hours to verify "Lam's problem" that there is no projective plane of order 10.[3]

[3] Bright et al. A SAT-based Resolution of Lam's Problem. *AAAI 2021*.

A SAT Success Story 3/4

Erdős asked whether every sufficiently large set of points (no three collinear) in $\mathbb{R}^2$ contains a k-hole: a k-sided convex polygon without a point inside.


there is a 4-hole in every set of five points

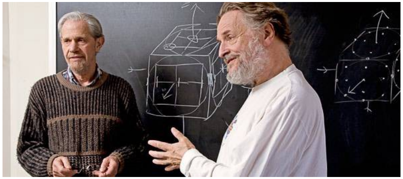
Moreover, every set of ten points contain a 5-hole, but there are arbitrarily large sets that do not contain a 7-hole.

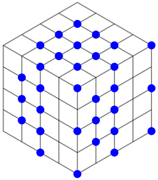
A SAT solver and 17,000 CPU hours was used to show that every set of 30 points has a 6-hole.[4]

[4] Heule, Scheucher. Happy Ending: An Empty Hexagon in Every Set of 30 Points. *TACAS 2024.*

A SAT Success Story 4/4

Conway and Kochen proved the *Free Will Theorem*—if humans have have "free will" then so do quantum particles. Their proof uses a set of 31 vectors called a Kochen–Specker (KS) system.





In 2021, I started mentoring Brian Zhengyu Li (Waterloo grad). He used a SAT solver to show a KS system must have ≥ 23 vectors and the work has been extended to ≥ 24 vectors.[5,6]

[5] Kirchweger, Peitl, Szeider. Co-Certifcate Learning with SAT Modulo Symmetries. *IJCAI 2023*.

[6] Li, Bright, Ganesh. A SAT Solver + Computer Algebra Attack on the Minimum Kochen–Specker Problem. *IJCAI 2024*.

. . . And *Many* More!

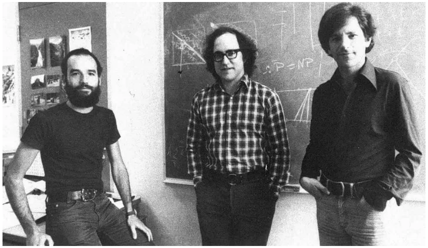
- Minimizing finite automata (Barnoff, Bright, Shallit).
- Dependency resolution when managing packages (Sakib, Asaduzzaman, Bright).
- Finding collisions in step-reduced SHA-256 (Alamgir, Negati, Bright).
- Computing new Schur numbers (Zaman, Ahmed, Bright).
- Searching for Latin squares of certain forms (Bright, Keita, Stevens).
- Finding new algorithms for 3×3 matrix multiplication (Heule, Kauers, Seidl).
- Solving variants of the Collatz conjecture (Yocu, Aaronson, Heule).

For this talk, I will focus on applying SAT to integer factorization—work with Yameen Ajani (MSc graduate, 2024).[7]

[7] Ajani, Bright. SAT and Lattice Reduction for Integer Factorization. *ISSAC 2024*.

Rivest–Shamir–Adleman Cryptosystem

The cryptosystem RSA relies on the difficulty of factoring large integers into primes.



RSA encryption involves a semiprime $N = p \cdot q$ for two randomly chosen primes p and q of the same bitlength (known only to the recipient).

The best known general attack on RSA involves factoring N, but no efficient integer factoring algorithms are known, unless you have a quantum computer.

Reduction of Factoring to SAT

Multiplication circuits can be converted to SAT by operating directly on the bit-representation of the integers.

Say $(N_3 N_2 N_1 N_0)_2$ is the binary representation of N. Use variables p_1, p_0 and q_1, q_0 to denote the bits of the prime factors of N:

$$\begin{array}{cccc} & & q_1 & q_0 \\ & \times & p_1 & p_0 \\ \hline & & a_1 & a_0 \\ & b_1 & b_0 & \\ \hline c_1 & c_0 & & \\ \hline N_3 & N_2 & N_1 & N_0 \end{array} \qquad \begin{aligned} a_0 &= p_0 q_0 \\ a_1 &= p_0 q_1 \\ b_0 &= p_1 q_0 \\ b_1 &= p_1 q_1 \end{aligned} \qquad \begin{aligned} a_0 &= N_0 \\ a_1 + b_0 &= N_1 + 2c_0 \\ b_1 + c_0 &= N_2 + 2c_1 \\ c_1 &= N_3 \end{aligned}$$

These equations can be broken into logical expressions, e.g., $a_0 \leftrightarrow (q_0 \wedge p_0)$, $N_1 \leftrightarrow (a_1 \oplus b_0)$, and $c_0 \leftrightarrow (a_1 \wedge b_0)$, etc.

SAT vs. Algebraic Methods

It's somewhat mind-boggling to realize that numbers can be factored without using any number theory! No greatest common divisors, no applications of Fermat's theorems, etc., are anywhere in sight. [...] Of course we can't expect this method to compete with the sophisticated factorization algorithms...

Donald Knuth, TAOCP 4B

As might be expected, computer algebraic methods *dramatically* outperform SAT.

The *number field sieve* can factor an n-bit integer heuristically in time $\exp(O^{\sim}(n^{1/3}))$ (super-polynomial, but sub-exponential in n).

Side-channel Attacks

Cryptographic implementations have an Achilles heel—they are implemented in the real world, *not* a platonic universe.

Side-channel attacks exploit the fact that cryptographic implementations may leak information about the private key in practice.

For example, timing how long a decryption takes might leak information about p and q.

Motivating Example 1/2

Suppose you are using disk encryption with RSA. In order to read from the disk, your private key, including the prime factors of N, is kept in memory.

What if an attacker steals your screen-locked machine? Is there any way they can extract your private key?

Motivating Example 1/2

Suppose you are using disk encryption with RSA. In order to read from the disk, your private key, including the prime factors of N, is kept in memory.

What if an attacker steals your screen-locked machine? Is there any way they can extract your private key?

Experiments have shown that after an hour without power, 99.9% of bits in DRAM modules remain readable—assuming the DRAM was kept in liquid nitrogen.[8]

[8] Halderman et al. Lest We Remember: Cold-Boot Attacks on Encryption Keys. *Communications of the ACM*, 2009.

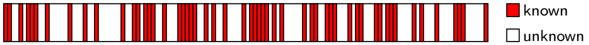
Motivating Example 2/2

When power is removed, bits in DRAM modules decay to a predictable ground state (say 0).

Any bits that are 1 after the power is removed **must originally have been 1**, while 0 bits may have been 0 or 1.

The result is that the attacker learns bits of the private key **at bit positions they don't control** (in practice, at essentially random positions).

bits of p:



Exploiting Leaked Bits

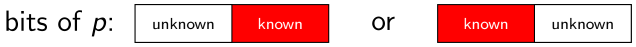
Algebraic methods like the number field sieve cannot seem to exploit leaked bits.

With SAT, it is easy assign any leaked bits of the prime factors to their correct value. This speeds up the solver—but SAT solvers are slow for this problem, as they don't exploit algebraic properties.

Question we address: *Can we use algebraic methods to improve SAT solvers on random leaked-bit factorization problems?*

Coppersmith's Method

Don Coppersmith showed that if the lowest or highest 50% of the bits of a prime factor of N are leaked...

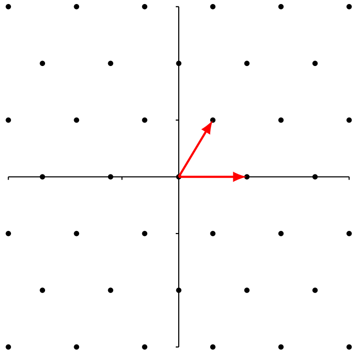


then N can be factored in polynomial time via the techniques of *polynomial root finding* and *lattice basis reduction*.[9]

[9] Coppersmith. Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known. *EUROCRYPT*, 1996.

Lattices

A *lattice* is a discrete subgroup of $\mathbb{R}^n$. For example, the *lattice* in $\mathbb{R}^2$ spanned by vectors $[3, 5]$ and $[6, 0]$ looks like:



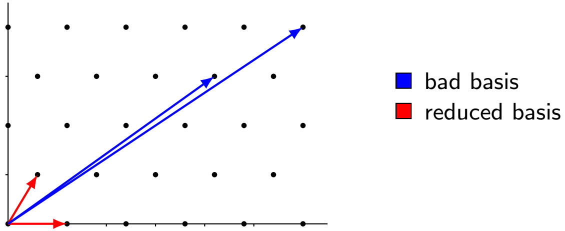
Coppersmith uses a lattice generated by polynomials using a correspondence like

$$a_0 + a_1 x + a_2 x^2 + a_3 x^3 \ \leftrightarrow\ [a_0, 10a_1, 10^2 a_2, 10^3 a_3] \in \mathbb{R}^4.$$

Lattice Basis Reduction

Lattices have many applications in mathematics. *Lattice basis reduction* is a method of finding a "reduced" basis—a basis having short and relatively orthogonal vectors.

The *LLL algorithm* can reduce a lattice basis and find an approximation of the shortest nonzero lattice vector.



Intuition Behind Coppersmith's Method

Coppersmith's method can find mod-p roots of a polynomial f, where p is an *unknown* divisor of a known integer N.

If p were known, finding roots of f mod p would be easy—but we don't know p.

Coppersmith finds another polynomial having the same roots as f mod p, except with the roots **over the reals**, and root finding over the reals can be done efficiently.

Factoring with Leaked Bits

Write $p = \hat{p} + \check{p}$, where $\hat{p}$ encodes the leaked high bits of p, and $\check{p}$ encodes the unknown low bits.

For example, $p = \hat{p} + \check{p} = 7580 + 3 = (1110110011100)_2 + (11)_2$.

With enough leaked bits, $\check{p}$ is a small mod-p root of

$$f(x) := \hat{p} + x,$$

as $f(\check{p}) = p \equiv 0 \pmod{p}$.

Coppersmith Example 1/2

Say $N = 58{,}563{,}509$ and $\hat{p} = 7580$, so $f(x) \coloneqq 7580 + x$.

Every vector in the lattice generated by

$$\begin{array}{rrrrr} f(x) & [\quad 7580, & 10, & 0, & 0] \\ xf(x) & [\quad 0, & 75800, & 100, & 0] \\ x^2 f(x) & [\quad 0, & 0, & 758000, & 1000] \\ N & [58563509, & 0, & 0, & 0] \end{array}$$

corresponds to a polynomial with $\check{p}$ as a mod-p root.

If the vector is also short, $\check{p}$ will in addition be a **real** root.

Coppersmith Example 2/2

Apply LLL to the lattice basis:

$$\begin{bmatrix} 7580 & 10 & & \\ & 75800 & 100 & \\ & & 758000 & 1000 \\ 58563509 & & & \end{bmatrix} \xrightarrow{\text{LLL}} \begin{bmatrix} 429 & -1460 & 100 & 0 \\ -9 & -540 & 1600 & 1000 \\ 15 & -530 & 2200 & -2000 \\ 7151 & 1470 & -100 & 0 \end{bmatrix}$$

The first vector of the reduced basis corresponds to $429 - 146x + x^2$ which has the integer roots 3 and 143.

The root $\check{p} = 3$ gives $f(\check{p}) = 7583$, the unknown factor of N.

Limitations of Coppersmith

Coppersmith can be generalized to work when the leaked bits of p are in multiple "chunks". However, the method is exponential in the number of chunks.[10]

Key point: *Coppersmith is not effective if the leaked bits are randomly distributed*.

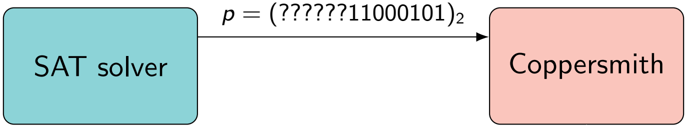
However, our SAT-based approach will use Coppersmith's method as a subroutine.

[10] Herrmann and May. Solving Linear Equations Modulo Divisors: On Factoring Given Any Bits. *ASIACRYPT 2008.*

SAT + Coppersmith

As SAT solvers search for solutions, they find “partial” solutions (where some variables will be unassigned).

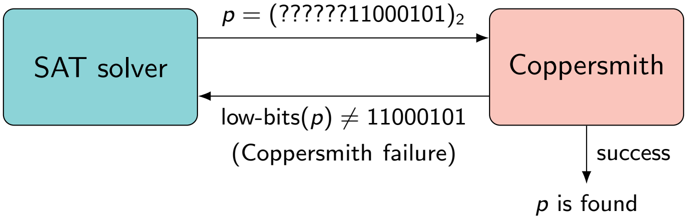
Say that a partial solution has assigned values to all of the bottom-half of the bits of p:



SAT + Coppersmith

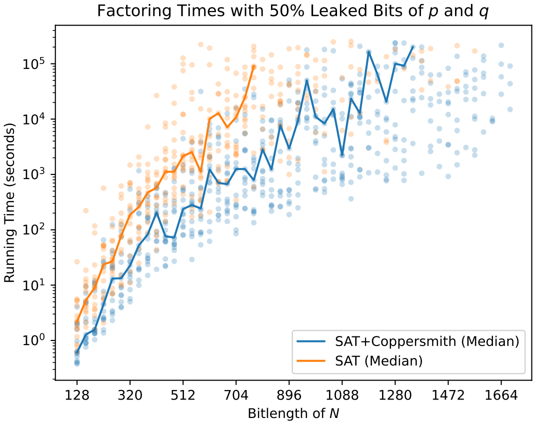
As SAT solvers search for solutions, they find “partial” solutions (where some variables will be unassigned).

Say that a partial solution has assigned values to all of the bottom-half of the bits of p:



If Coppersmith’s method succeeds, N is factored. If not, tell the solver that at least one of the low bits of p must change.

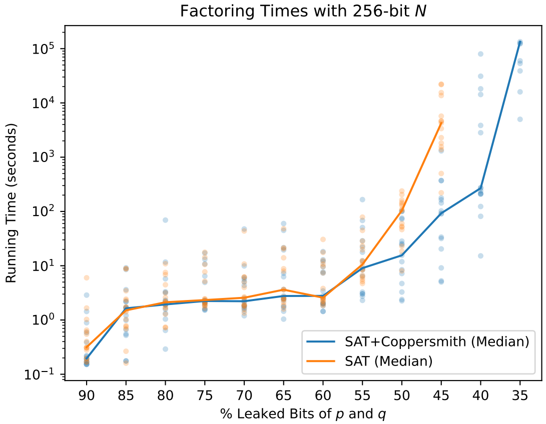
Results 1/2



Fifteen random instances were run for 3 days for varying bitlengths. For comparison, the number field sieve on 512-bit N uses around 2770 CPU hours.[11]

[11] Valenta et al. Factoring as a Service. *Financial Cryptography and Data Security*, 2016.

Results 2/2



Fifteen random instances were run for 3 days for varying leaked percentages. For comparison, an algebraic "branch and prune" technique[12] with 40% leaked bits used around 2000 seconds and 90 GiB; our approach used at most 0.5 GiB.

[12] Heninger and Shacham. Reconstructing RSA Private Keys from Random Key Bits. *CRYPTO* 2009.

Final Thoughts

I've been working on combining SAT with computer algebra systems (CAS) for 10 years. In many problems SAT+CAS solvers provide exponential speedups over other approaches.

The approach works well for problems requiring *both* search and advanced mathematics.

review articles

DOI:10.1145/3500921

The science of less-than-brute force.

BY CURTIS BRIGHT, ILIAS KOTSIREAS, AND VIJAY GANESH

When Satisfiability Solving Meets Symbolic Computation

MATHEMATICIANS HAVE LONG been fascinated by objects that exhibit exceptionally nice combinatorial properties. However, it is often difficult to determine whether objects satisfying a given combinatorial property exist. Sometimes, the only feasible method of definitively answering the question of existence is simply to perform a systematic search. A famous example of this is the proof of the *four-color theorem*—the notion that four colors suffice to color the regions of a planar map with adjacent regions colored differently.[3] The theorem has been known to be true since 1977, but every known proof relies on computer calculations in an essential way. Mathematical arguments are used to reduce the search for counterexamples to a finite number of cases, and the cases are then exhaustively checked using a custom-written computer program to rule out any counterexamples.

Independently, computer scientists have made significant progress over the last 50 years on developing general-purpose programs that can automatically solve many kinds of mathematical problems. *Satisfiability solving* and *symbolic computation* are two important branches of computer science that each specialize in solving mathematical problems. Both fields have long histories and have produced impressive tools—satisfiability (SAT) solvers in the former and computer algebra systems (CASs) in the latter. Originally, SAT solvers were designed to solve problems in logic, and CASs were tools to manipulate and simplify algebraic expressions. As we will see, these tools have since found an abundance of new applications outside of these original domains.

Despite their common specialization in solving mathematical problems, the SAT and CAS communities have developed independently of each other.[1] Broadly speaking, the SAT community has focused on effective search methods, while the CAS community has focused on effective mathematical algorithms. Recently, these two communities have started to collaborate in crossover initiatives like the SC-square project.[a,2] Since the insights of these communities are largely complementary, bringing them together has resulted in new solutions to problems that were out-of-reach of either community separately and has produced advances in problems involving nonlinear real arithmetic,[10] linear integer arithmetic,[13] and Boolean polynomials,[20] to name a few. In this overview, we focus on our own contribution to this ongoing project—a hybrid SAT and CAS system called MathCheck[b] that we have applied to mathematical problems in graph theory,[16] finite geometry,[5] combinatorics,[9] and number theory.[c,11]

Satisfiability solving. A SAT solver is a program that solves the satisfiability problem from Boolean logic—given a formula in conjunctive normal form, is there an assignment to its variables that makes the expression true? At first glance, SAT solvers seem disconnected from the kinds of problems that most mathematicians and engineers care about. However, stunning progress in applied SAT solving over the last several decades[11] has led to a surprising diversity of applications for SAT solvers—

a www.sc-square.org

b www.uwaterloo.ca/mathcheck

c For more applications of the SAT+CAS paradigm see the overview article[6] appearing in a special issue of the *Journal of Symbolic Computation* devoted to SAT and CAS synergies.

key insights

- Satisfiability (SAT) solving and symbolic computation are fields of computer science with distinguished histories that have developed mostly independently.
- Advances in SAT solving and computer algebra systems (CASs) have led to the development of tools that can solve mathematical search problems significantly larger than ever before—and in a faster, more verifiable way.
- Hybrid "SAT+CAS" systems combine the efficient search and learning routines of SAT solvers with the efficient mathematical algorithms and expressiveness of CASs in order to achieve the best of both worlds.

84 COMMUNICATIONS OF THE ACM | JULY 2022 | VOL. 65 | NO. 7

JULY 2022 | VOL. 65 | NO. 7 | COMMUNICATIONS OF THE ACM 85

Communications of the ACM, 2022