SAT and Lattice Reduction for Integer Factorization

Curtis Bright

University of Windsor

joint with my MSc student Yameen Ajani

August 22, 2024

PIMS-CORDS SFU Operations Research Seminar

🏠 curtisbright.com

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:

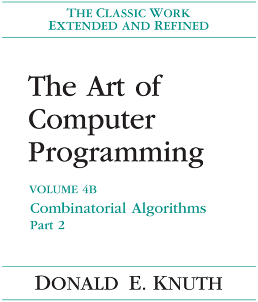
$$(x \vee y) \wedge (\neg x \leftrightarrow z)$$

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 is important!

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



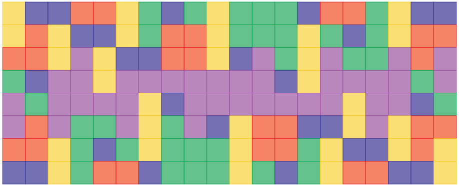
Despite having no provably fast algorithms, "SAT solvers" can be surprisingly effective and can solve many search problems seemingly unrelated to Boolean logic, like Sudoku.[1]

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

A SAT Success Story I

In 1912, Issai Schur showed any colouring of the positive integers using k colours must have a monochromatic triple $(x, y, x+y)$.

Determining how far you can k-colour the postive integers before introducing a monochromatic triple $(x, y, x+y)$ is difficult.



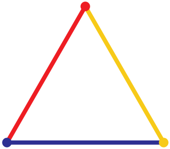
A 5-colouring of the integers from 1 to 160 with no monochromatic triple $(x, y, x+y)$.

Heule used a SAT solver and 123,000 CPU hours to show every way of 5-colouring 1 to 161 has a monochromatic triple $(x, y, x+y)$.[2]

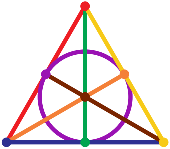
[2] Heule. Schur Number Five. *AAAI 2018*.

A SAT Success Story II

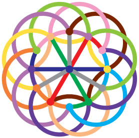
In a *projective plane*, every pair of lines meet at a unique point (and every pair of points define a unique line). It has *order n* when every line has $n + 1$ points.



order 1



order 2



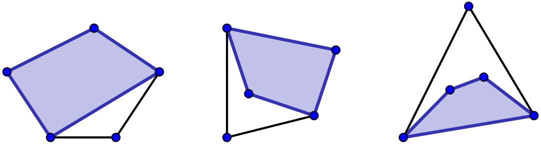
order 3

Projective planes exist for every prime power order, and do not exist in order 6. *Lam's problem* is to determine if they exist in order 10. Bright et al. used a SAT solver and 15,000 CPU hours to verify Lam's result that a plane of order 10 does not exist.[3]

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

A SAT Success Story III

Erdős asked whether every sufficiently large set of points in the plane with no three collinear points contains a k-hole: a k-gon without a point inside.



every set of five points contains a 4-hole

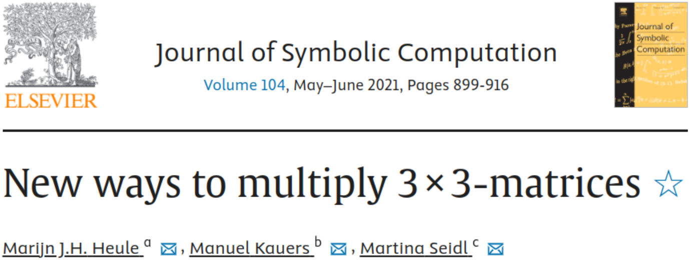
Additionally, every set of ten points contain a 5-hole, but there are arbitrarily large sets that do not contain a 7-hole.

Heule and Scheucher used a SAT solver and 17,000 CPU hours 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.

A SAT Success Story IV

How fast can you multiply 3×3 matrices? Before 2021, four algorithms were known using 23 scalar multiplications. Then. . .


ELSEVIER
Journal of Symbolic Computation
Volume 104, May–June 2021, Pages 899-916

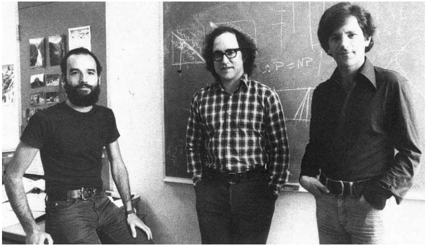
New ways to multiply 3×3-matrices ☆

Marijn J.H. Heule [a], Manuel Kauers [b], Martina Seidl [c]

Heule et al. used a SAT solver and over 300,000 CPU hours to find over 17,000 distinct 3×3 matrix multiplication algorithms with 23 scalar multiplications.

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:

$$\begin{array}{rrrr} & & q_1 & q_0 \\ & \times & p_1 & p_0 \\ \hline & & a_1 & a_0 \\ & b_1 & b_0 & \\ \hline c_1 & c_0 & & \\ \hline N_3 & N_2 & N_1 & N_0 \end{array}$$

$$(a_1 a_0)_2 = (q_1 q_0)_2 \times p_0$$
$$(b_1 b_0)_2 = (q_1 q_0)_2 \times p_1$$

$$N_0 = a_0$$
$$(c_0 N_1)_2 = a_1 + b_0$$
$$(c_1 N_2)_2 = b_1 + c_0$$
$$N_3 = c_1$$

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.

Motivating Example

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

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.[5]

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

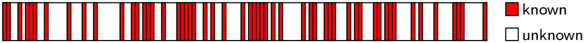
Motivating Example II

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...

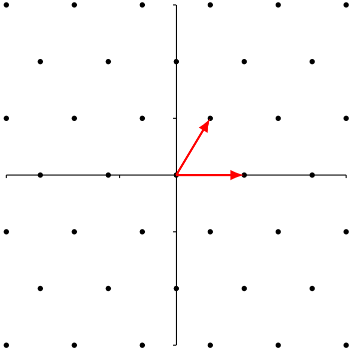
bits of p:

then N can be factored in polynomial time via the techniques of *integer root extraction* and *lattice basis reduction*.[6]

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

Lattices

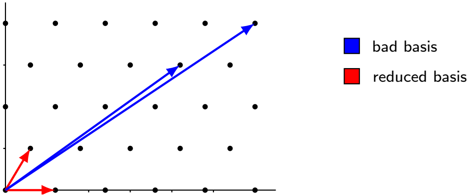
A *lattice* is a discrete subgroup of $\mathbb{R}^n$. For example, the *lattice* in $\mathbb{R}^2$ spanned by vectors $[3, 5]$ and $[6, 0]$ looks like:



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* reduces an n-dimensional lattice in polynomial time in n and finds an approximation of the shortest nonzero lattice vector.



Intuition Behind Coppersmith's Method

Coppersmith's method finds all *small* modular roots x_0 of a polynomial f mod p where p is an *unknown* divisor of N.

It works by setting up a lattice for which a short lattice vector corresponds to a polynomial g with $g(x_0) = 0$ *over the integers*.

Root finding over the integers can be done efficiently, so x_0 can be recovered from g by integer root extraction.

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 $f(x) \coloneqq \hat{p} + x$ that Coppersmith can find.

Example of Coppersmith

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

Associate $a_0 + a_1 x + a_2 x^2 + a_3 x^3$ with the lattice vector

$$[a_0, 10a_1, 10^2 a_2, 10^3 a_3].$$

A short vector in the lattice generated by the polynomials

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

will reveal a polynomial having the root $\check{p}$ *over the integers*.

Example Continued

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.[7]

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

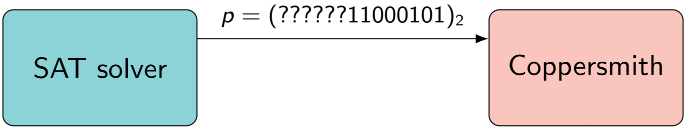
However, our SAT-based approach will use Coppersmith's method as a subroutine.

[7] 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).

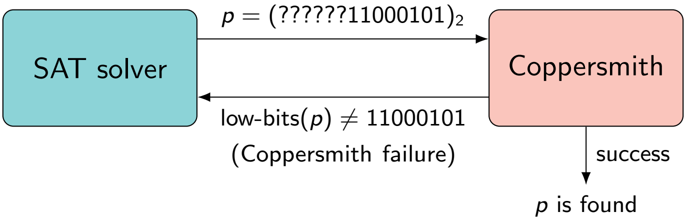
Say that a partial solution has assigned values to all of the bottom-half of the bits of p:



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.

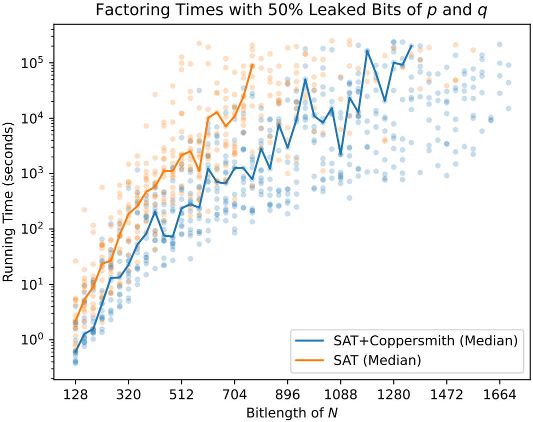
Experimental Setup

For varying bitlengths and percentages of leaked bits, we compared the SAT solver MapleSAT with a version of MapleSAT calling Coppersmith's method on 15 randomly generated instances.

Coppersmith's method (implemented with fplll)[8] is used when at least 60% of the low bits of p are known, as this allows using a lattice of fixed dimension 5 (regardless of the size of N).

[8] fplll, a lattice reduction library, https://github.com/fplll/fplll

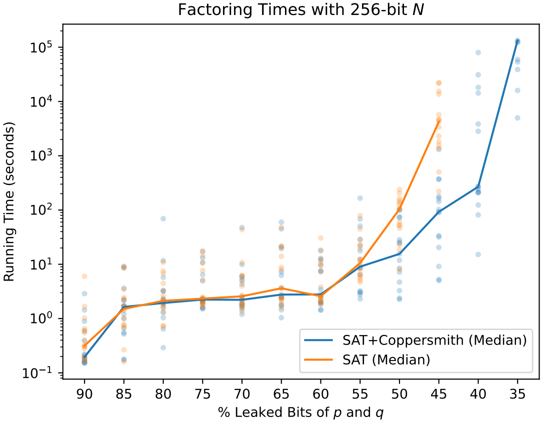
Results



Each instance was run for 3 days. For comparison, the number field sieve on 512-bit N uses around 2770 CPU hours.[9]

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

Results II



Each instance was run for 3 days and used at most 0.5 GiB of RAM. For comparison, an algebraic "branch and prune" technique with 40% leaked bits used around 2000 seconds and 90 GiB.[10]

[10] 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 almost 10 years. SAT+CAS solvers often provide exponential speedups over pure SAT or pure CAS approaches.

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


review articles

The science of less-than-brute force.

BY CURTIS BRIGHT, ILIAS KOTSIREAS, AND VIJAY GANESH

When Satisfiability Solving Meets Symbolic Computation

Communications of the ACM, 2022