Reductions in Number Theory: Problems and Prospects

Eric Bach

University of Wisconsin
Computer Sciences Dept.
Madison, WI 53706

Some References:

Woll, Reductions Among Number Theoretic Problems, *Inform. Comput.* 1987

Bach & Shallit, Factoring With Cyclotomic Polynomials, *Math. Comp.* 1989

Galbraith & McKee, Pairings on Elliptic Curves over Finite Commmutative Rings, *Proc. Cryptography and Coding* 2005

Coron & May, Deterministic Polynomial Time Equivalence of Computing the RSA Secret Key and Factoring, *J. Crypt.* 2007

Bach & Charles, The Hardness of Computing an Eigenform, *Contemp. Math.* 2008

Zralek, A Deterministic Version of Pollard's $p-1$ Algorithm, *Math. Comp.* 2010

We Were Students Once ... and Young

What Happened in the 1970s?

- P vs NP question
  - Can clever computation substitute for lucky guessing?
  - Still open (almost 40 years!)
- Academic cryptography research
  - Public key systems
  - Use of multiplicative number theory

Lessons From P vs NP

The master combinatorial problem is Boolean satisfiability:

$$\text{Can } x \vee (\bar{y} \wedge z) \vee (\bar{x} \wedge \bar{z}) = 0?$$

Thousands of problems equivalent to it

Is there a dichotomy (everything either in P or NP-complete)?

> No. If P $\neq$ NP, there are NP-intermediate problems (Ladner)

Some early NP-intermediate candidates (primality, linear programming) are now known to be in P.

What's left?

> Integer factorization
>
> Graph isomorphism
>
> No known connection between them

Problem Reductions

The key technical “glue” for NP-completeness and related theories

A is “no harder than” B if we can use a method for solving B to solve A

In a Reduction, What Counts as a Method?

- Classically, an algorithm is a step by step deterministic procedure for accomplishing some task of a symbolic nature
  - Examples: addition, subtraction, multiplication of decimal numbers
  - Is long division an algorithm? Not as usually taught! (Guessing is required.)
- We'll ignore details of run times, only care if poly time or not
  - Allows methods to be composed
  - Insensitive to machine model details
- Number theorists have also included
  - Randomization
  - Name-brand heuristic assumptions (ERH, ABC, BSD, ...)
  - Ad hoc assumptions (the obvious attack on my new crypto system is "hard")
- Appeals to physical effects (e.g. quantum computation) are interesting but not considered here

Problem Reductions B.C. (Before Computers)

Usually used to show a problem was "easy"

Completing the square:

$$ax^2 + bx + c = 0 \qquad \to \qquad \sqrt{b^2 - 4ac}$$

Trig function integration:

$$\int f(\sin x, \cos x)dx \qquad \to \qquad \int g(z)dz,$$

magic substitution is $z = \tan(x/2)$.

Many more examples!

Arguments for "hardness" were not as common

Gauss studied construction of regular n-gons with compass and straightedge. We can do $n = 3$ (equilateral triangle) but not $n = 9$. Therefore, angle trisection is not possible.

1950s: Undecidable problems from non-logical mathematics, such as testing whether two 4-manifolds have the same shape.

Lessons From Cryptography

Complexity is Useful!

- We can design systems with “hard” problems embedded
  - Cryptographic methods based on algebra
  - Pseudo-random generators for specific purposes (e.g. conditional proof that BPP=P).
  - What about error-correcting codes, computer algebra?

Reductions Point to System Weaknesses

- Example: strong primes for RSA
  - The ideal RSA prime has none of $p \pm 1$, $p^2 \pm p + 1$, ... smooth
  - In practice, a random key may be close enough to ideal

Reductions Focus Attention on a Few Standard Attacks

- We lessen the number of unsolved problems
- We concentrate our force at one place

The Master Problem of Elementary Number Theory

Any $n \geq 1$ is uniquely a product of primes:

$$n = \prod_{i=1}^{r} p_i^{e_i}$$

Want to do this quickly, given n in binary.

Bit length of n is about $\log_2 n$, so poly time is $O(\log n)^k$ bops, for some k.

The best deterministic algorithms are still exponential: $n^{1/4+o(1)}$ (Pollard/Strassen).

Evidence that Factoring is NP-Intermediate

With randomization, one can achieve

$$\exp\left((\log n)^{1/2+o(1)}\right)$$

Practical algorithms (number field sieve) apparently have exponent 1/3.

The natural decision problem for factoring lies in

$$\text{NP} \cap \text{co-NP}.$$

Historically, Most Problems were “Solved” via Factoring

- Multiplicative orders of elements mod n
- Finitely generated Abelian group structure
- Number of divisors $d(n)$
- Sum of divisors $\sigma(n)$ and other moments
- Many others!

Is factoring the “obstruction” to number theory, as Boolean satisfiability is for combinatorial problems?

Research Program: Develop a mini-version of NP-completeness, centered on factoring and related problems.

- Early successes with classical problems and reductions using randomization and/or ERH
- Efforts toward derandomization and de-hypothesization have progressed slowly
- Recent work suggests a tie to modularity

Remainder of the Talk

Bringing Problems into the “Big Tent” of Factoring

I. Embedded Factors

II. Equations in Rings

III. Modularity

IV. Deterministic Reductions

Any Other Tents to Visit?

Yes, but this is only a 40 minute talk

Other problems touched on as appropriate

I. Embedded Factors

Smarandache's function (Parberry)

$S(n) = \min\{m : n \text{ divides } m!\}$.

Erdős: $S(n)$ is the largest prime factor of n, for almost all n.

This suggests it has same complexity as factoring.

Reducing S to factoring

This is "maxiplicative":

$$S(n) = \max_{p^e || n} S(p^e)$$

The local contribution is the smallest m with

$$\nu_p(m!) \geq e$$

Use

$$\nu_p(m!) = \lfloor \frac{m}{p} \rfloor + \lfloor \frac{m}{p^2} \rfloor + \lfloor \frac{m}{p^3} \rfloor + \ldots$$

plus binary search.

Reducing factoring to S

For $n > 4$, we have $1 < S(n) < n$, except for prime n.

When $m = S(n)$, the product $m!$ has a prime power that wasn't in $(m-1)!$.

So $\gcd(n, S(n)) > 1$.

Example: $S(25) = 10$, and $\gcd(10, 25) = 5$.

Features of Our Example Reduction

We actually split n, must repeat to get the whole factorization

- Different values of S are used, so this is a Turing reduction
- How many calls are needed?

Last step is a gcd computation (typical)

- Euclid's algorithm uses $O(\log n)$ division steps (Lamé)
- So this runs in poly time

No randomness or extra hypotheses needed

II. Equations in Rings

Let $n = pq$ for simplicity. Then (CRT)

$$R := \mathbf{Z}/n\mathbf{Z} \cong \mathbf{F}_p \oplus \mathbf{F}_q$$

Finding "non-obvious" solutions to equations in R is often equivalent to factoring.

Example 1: $x^2 - x = 0$

Solutions are the idempotents of R

If $x \neq 0, 1$, then $1 < \gcd(x, n) < n$.

Example 2: $x^2 = 1$

Solutions are order ≤ 2 elements of R^*.

If $x \neq \pm 1$, then $\gcd(x-1, n)$ splits n.

This led to the idea of "pushing into subgroups" (Pollard)

Suppose $p - 1 | E$. If $x \in_R \mathbf{Z}/n\mathbf{Z}^*$,

$$y = x^E = (x_p^E, x_q^E) = (1, z).$$

We hope that $\gcd(y-1, n)$ splits n.

Possibility of $z = 1$ can be handled by using $E/2, E/4, \ldots$.

Application: hardness of Euler totient $\varphi(n)$ (RSA)

1980s: Replacing $\mathbf{F}_p$ by extension fields gave proofs that many classical functions are as hard as factoring:

Divisor sums $\sigma_k(n) = \sum_{d|n} d^k$

Multiples of $\Phi_k(p)$ (cyclotomic polynomial)

Principal technical difficulty: Doing Frobenius $x \mapsto x^p$ without knowing p

But not all of them!

Number of divisors $d(n)$

Möbius function $\mu(n)$

Not enough “information” to get a reduction from factoring?

Best result so far (Shallit-Shamir): $d(n)$ is equally hard as finding the “shape” (list of prime factorization exponents).

The Analog of #P: Counting Problems

Units

$$\varphi(n) = |\mathbf{Z}/n\mathbf{Z}^*| = \prod_{p^e || n} (p-1)p^{e-1}$$

Quadratic Residues

For odd n,

$$\mathrm{QR}(n) := |(\mathbf{Z}/n\mathbf{Z}^*)^2| = \prod_{p^e || n} \frac{p-1}{2} p^{e-1},$$

Presence of

$$\Phi_1(p) = p - 1$$

makes both problems as hard as factoring.

Modular Squares

Let

$$\mathrm{SQ}(n) := |(\mathbf{Z}/n\mathbf{Z})^2| = \#\{y : y = x^2\}$$

For odd p, $2\mathrm{SQ}(p^e)$ is a polynomial in p (Delcourte):

$$\begin{array}{lr} e = 1 & p + 1 \\ e = 2 & p^2 - p + 2 \\ e = 3 & p^3 - p^2 + p + 1 \\ & \ldots \end{array}$$

So $\mathrm{SQ}(n)$ contains the factor $p + 1$, unless n is powerful.

Most numbers aren't powerful (Golomb):

$$[\# \text{ of powerful } n \le x] \ \sim \frac{\zeta(3/2)}{\zeta(3)}\sqrt{x}$$

So SQ is as hard as factoring, on almost all n.

Extend to all n? We'd need to exploit shifted cyclotomic polynomials $\Phi_k(p) + a$.

General Principle: The "mod n" version of a "mod p" problem is often as hard as factoring.

Some examples from the 1980s:

Size of $\mathbf{Z}/n\mathbf{Z}^*$

Solving $a^x = b$ in $\mathbf{Z}/n\mathbf{Z}^*$

Solving $x^2 = a \bmod n$

Elliptic curve versions of these problems were proved hard in the 1990s:

Size of E_n (Kunihiro-Koyama)

Discrete logarithm $xP = Q$ on E_n (same)

Square roots $2X = P$ (Meyer-Müller)

Tate pairings on E_n (Galbraith-McKee)

What about nontrivial point construction?

For elliptic curves over finite fields, a deterministic method was found only recently (Shallue-van de Woestijne)

Corresponding problem for $\mathbf{Z}/n\mathbf{Z}^*$ (find a unit) is easy: try $> \log_2 n$ primes.

III. Modularity

Modular forms: Complex functions satisfying periodicity and growth requirements

Their power series coefficients often encode arithmetic information

Example 1: Eisenstein series (the "interesting" part)

$$G_k(X) = \sum_{n \geq 1} \sigma_{2k-1}(n) X^n$$

Coefficients are divisor sums, so hard to compute.

Example 2: Ramanujan's function $\tau(n)$, defined by

$$\Delta(X) := X \prod_{m \geq 1} (1 - X^m)^{24} = \sum_{n \geq 1} \tau(n) X^n$$

The function Δ is a weight 12, level 1 mod form

Why is Tau as Hard as Factoring?

Mordell: $\tau(n)$ is multiplicative, has recurrence relation at prime powers

To factor $n = pq$ (RSA modulus), compute

$$\tau(n)^2 = n^{11}x^2y^2,$$
$$\tau(n^2) = n^{11}(x^2-1)(y^2-1).$$

Solve quadratic equation to get x^2, y^2. Then

$$x^2 = \frac{\tau(p)^2}{p^{11}} \in \mathbf{Q}^2,$$

so in lowest terms $\nu_p(\text{denom})$ is odd.

This factors all RSA moduli n if Lehmer's conjecture that $\tau(p) \neq 0$ is true. Unconditionally, factors almost all of them.

Reduction extends to a large class of Hecke operator eigenvalues.

Open: extend to all n

IV. Derandomization and De-hypothesization

Many reductions from factoring used randomness and/or ERH. Can these be eliminated?

AKS: Prime testing is (unconditionally) in P. Does this help?

> Apparently not. We usually employ randomness to
>
> > Enhance reliability
> >
> > Search for “useful” elements
>
> AKS is engineered to eliminate all doubt about primality, but does nothing else.

What Progress Has Been Made?

For the two-prime (RSA) case, many reductions can be made deterministic

- Euler totient $\varphi(n)$ (exact)
  - $n = pq, \qquad \varphi(n) = n - (p+q) + 1$
  - Extract sum pq and product $p+q$, solve a quadratic.
- Divisor sums $\sigma_k(n)$
  - Use same idea, e.g. $\sigma(n) = n + (p+q) + 1$
- RSA key analysis (recover d from (n, e))
  - (Coron-May): use LLL to obtain exact $\varphi(n)$
  - Keys should be "textbook RSA", e.g. $de < n$.

Open: Extend Beyond Two Primes

What Deterministic Reductions are Known for General n?

Bachmann (about 100 years ago) computed the "derivative"

$$\theta(n) = \prod_{p^e || n} p^{e-1}$$

by evaluating φ several times.

If we know θ we can extract the maximal squarefree divisor.

We can replace φ by the maximum exponent λ (Landau)

Zralek:

Unconditional deterministic version of Pollard's $p-1$ algorithm

With $O(\log n)$ calls to an oracle for $\varphi(n)$, we can factor n in time less than NFS takes.

Subexponential reductions were used earlier by Maurer and Wolf (for discrete logs).