Factorization: state of the art

1. Batch NFS
2. Factoring into coprimes
3. ECM

D. J. Bernstein
University of Illinois at Chicago

Tanja Lange
Technische Universiteit Eindhoven

Sieving small integers $i > 0$
using primes $2, 3, 5, 7$:

1								
2	2							
3					3			
4	2	2						
5							5	
6	2				3			
7								7
8	2	2	2					
9					3	3		
10	2						5	
11								
12	2	2			3			
13								
14	2							7
15					3		5	
16	2	2	2	2				
17								
18	2				3	3		
19								
20	2	2					5	

etc.

Sieving i and $611 + i$ for small i using primes $2, 3, 5, 7$:

1								
2	2							
3					3			
4	2	2						
5							5	
6	2				3			
7								7
8	2	2	2					
9					3	3		
10	2						5	
11								
12	2	2			3			
13								
14	2							7
15					3		5	
16	2	2	2	2				
17								
18	2				3	3		
19								
20	2	2					5	

612	2	2			3	3						
613												
614	2											
615					3			5				
616	2	2	2									7
617												
618	2				3							
619												
620	2	2						5				
621					3	3	3					
622	2											
623												7
624	2	2	2	2	3							
625								5	5	5	5	
626	2											
627					3							
628	2	2										
629												
630	2				3	3		5				7
631												

etc.

Have complete factorization of the "congruences" $i(611+i)$ for some i's.

$14 \cdot 625 = 2^1 3^0 5^4 7^1$.
$64 \cdot 675 = 2^6 3^3 5^2 7^0$.
$75 \cdot 686 = 2^1 3^1 5^2 7^3$.

$14 \cdot 64 \cdot 75 \cdot 625 \cdot 675 \cdot 686$
$= 2^8 3^4 5^8 7^4 = (2^4 3^2 5^4 7^2)^2$.

$\gcd\{611, 14 \cdot 64 \cdot 75 - 2^4 3^2 5^4 7^2\}$
$= 47$.

$611 = 47 \cdot 13$.

Why did this find a factor of 611?
Was it just blind luck:
$\gcd\{611, \text{random}\} = 47$?

No.

By construction 611 divides $s^2 - t^2$
where $s = 14 \cdot 64 \cdot 75$
and $t = 2^4 3^2 5^4 7^2$.
So each prime > 7 dividing 611
divides either $s - t$ or $s + t$.

Not terribly surprising
(but not guaranteed in advance!)
that one prime divided $s - t$
and the other divided $s + t$.

Why did the first three completely factored congruences have square product? Was it just blind luck?

Yes. The exponent vectors $(1, 0, 4, 1), (6, 3, 2, 0), (1, 1, 2, 3)$ happened to have sum 0 mod 2.

But we didn't need this luck! Given long sequence of vectors, easily find nonempty subsequence with sum 0 mod 2.

This is linear algebra over $\mathbf{F}_2$.
Guaranteed to find subsequence
if number of vectors
exceeds length of each vector.

e.g. for $n = 671$:

$$1(n + \ \ 1) = 2^5 3^1 5^0 7^1;$$
$$4(n + \ \ 4) = 2^2 3^3 5^2 7^0;$$
$$15(n + 15) = 2^1 3^1 5^1 7^3;$$
$$49(n + 49) = 2^4 3^2 5^1 7^2;$$
$$64(n + 64) = 2^6 3^1 5^1 7^2.$$

$\mathbf{F}_2$-kernel of exponent matrix is
gen by (0 1 0 1 1) and (1 0 1 1 0);
e.g., $1(n+1)15(n+15)49(n+49)$
is a square.

Plausible conjecture: $\mathbf{Q}$ sieve can separate the odd prime divisors of any n, not just 611.

Given n and parameter y:

Try to completely factor $i(n+i)$ for $i \in \{1, 2, 3, \ldots, y^2\}$ into products of primes $\leq y$.

Look for nonempty set of i's with $i(n+i)$ completely factored and with $\prod_i i(n+i)$ square.

Compute $\gcd\{n, s-t\}$ where $s = \prod_i i$ and $t = \sqrt{\prod_i i(n+i)}$.

How large does y have to be for this to find a square?

Uniform random integer in $[1, n]$ has $n^{1/u}$-smoothness chance roughly u^{-u}.

Plausible conjecture:
$\mathbf{Q}$ sieve succeeds with $y = \lfloor n^{1/u} \rfloor$ for all $n \geq u^{(1+o(1))u^2}$; here $o(1)$ is as $u \to \infty$.

More generally, if $y \in \exp\sqrt{\left(\frac{1}{2c} + o(1)\right)\log n \log\log n}$, conjectured y-smoothness chance is $1/y^{c+o(1)}$.

Find enough smooth congruences by changing the range of i's: replace y^2 with $y^{c+1+o(1)} = \exp\sqrt{\left(\frac{(c+1)^2+o(1)}{2c}\right)\log n \log\log n}$.

Increasing c past 1
increases number of i's but
reduces linear-algebra cost.
So linear algebra never dominates
when y is chosen properly.

Improving smoothness chances

Smoothness chance of $i(n+i)$
degrades as i grows.
Smaller for $i \approx y^2$ than for $i \approx y$.

Crude analysis: $i(n+i)$ grows.
$\approx yn$ if $i \approx y$;
$\approx y^2 n$ if $i \approx y^2$.

More careful analysis:
$n+i$ doesn't degrade, but
i is always smooth for $i \le y$,
only 30% chance for $i \approx y^2$.

Can we select congruences
to avoid this degradation?

Choose q, square of large prime.
Choose a "q-sublattice" of i's:
arithmetic progression of i's
where q divides each $i(n+i)$.
e.g. progression $q - (n \bmod q)$,
$2q - (n \bmod q)$, $3q - (n \bmod q)$,
etc.

Check smoothness of
generalized congruence $i(n+i)/q$
for i's in this sublattice.
e.g. check whether $i, (n+i)/q$ are
smooth for $i = q - (n \bmod q)$ etc.

Try many large q's.
Rare for i's to overlap.

e.g. $n = 314159265358979323$:

Original $\mathbf{Q}$ sieve:

i	$n+i$
1	314159265358979324
2	314159265358979325
3	314159265358979326

Use 997^2-sublattice,
$i \in 802458 + 994009\mathbf{Z}$:

i	$(n+i)/997^2$
802458	316052737309
1796467	316052737310
2790476	316052737311

Crude analysis: Sublattices eliminate the growth problem. Have practically unlimited supply of generalized congruences

$$(q-(n \bmod q))\frac{n+q-(n \bmod q)}{q}$$

between 0 and n.

More careful analysis: Sublattices are even better than that! For $q \approx n^{1/2}$ have $i \approx (n+i)/q \approx n^{1/2} \approx y^{u/2}$ so smoothness chance is roughly $(u/2)^{-u/2}(u/2)^{-u/2} = 2^u/u^u$, 2^u times larger than before.

Even larger improvements from changing polynomial $i(n+i)$.

"Quadratic sieve" (QS) uses $i^2 - n$ with $i \approx \sqrt{n}$; have $i^2 - n \approx n^{1/2+o(1)}$, much smaller than n.

"MPQS" improves $o(1)$ using sublattices: $(i^2 - n)/q$. But still $\approx n^{1/2}$.

"Number-field sieve" (NFS) achieves $n^{o(1)}$.

Generalizing beyond $\mathbf{Q}$

The $\mathbf{Q}$ sieve is a special case of the number-field sieve.

Recall how the $\mathbf{Q}$ sieve factors 611:

Form a square as product of $i(i + 611j)$ for several pairs (i, j):
$14(625) \cdot 64(675) \cdot 75(686)$
$= 4410000^2$.

$\gcd\{611, 14 \cdot 64 \cdot 75 - 4410000\}$
$= 47$.

The $\mathbf{Q}(\sqrt{14})$ sieve factors 611 as follows:

Form a square as product of $(i + 25j)(i + \sqrt{14}j)$ for several pairs (i, j):

$$(-11 + 3 \cdot 25)(-11 + 3\sqrt{14})$$
$$\cdot (3 + 25)(3 + \sqrt{14})$$
$$= (112 - 16\sqrt{14})^2.$$

Compute

$s = (-11 + 3 \cdot 25) \cdot (3 + 25)$,

$t = 112 - 16 \cdot 25$,

$\gcd\{611, s - t\} = 13$.

Why does this work?

Answer: Have ring morphism $\mathbf{Z}[\sqrt{14}] \to \mathbf{Z}/611$, $\sqrt{14} \mapsto 25$, since $25^2 = 14$ in $\mathbf{Z}/611$.

Apply ring morphism to square:

$$(-11 + 3 \cdot 25)(-11 + 3 \cdot 25) \cdot (3 + 25)(3 + 25)$$
$$= (112 - 16 \cdot 25)^2 \text{ in } \mathbf{Z}/611.$$

i.e. $s^2 = t^2$ in $\mathbf{Z}/611$.

Unsurprising to find factor.

Generalize from $(x^2 - 14, 25)$
to (f, m) with irred $f \in \mathbf{Z}[x]$,
$m \in \mathbf{Z}$, $f(m) \in n\mathbf{Z}$.

Write $d = \deg f$,
$f = f_d x^d + \cdots + f_1 x^1 + f_0 x^0$.

Can take $f_d = 1$ for simplicity,
but larger f_d allows
better parameter selection.

Pick $\alpha \in \mathbf{C}$, root of f.
Then $f_d\alpha$ is a root of
monic $g = f_d^{d-1} f(x/f_d) \in \mathbf{Z}[x]$.

$$\mathbf{Q}(\alpha) \leftarrow \mathcal{O} \leftarrow \mathbf{Z}[f_d\alpha] \xrightarrow{f_d\alpha \mapsto f_d m} \mathbf{Z}/n$$

Build square in $\mathbf{Q}(\alpha)$ from congruences $(i - jm)(i - j\alpha)$ with $i\mathbf{Z} + j\mathbf{Z} = \mathbf{Z}$ and $j > 0$.

Could replace $i - jx$ by higher-deg irred in $\mathbf{Z}[x]$; quadratics seem fairly small for some number fields. But let's not bother.

Say we have a square $\prod_{(i,j)\in S}(i - jm)(i - j\alpha)$ in $\mathbf{Q}(\alpha)$; now what?

$\prod(i - jm)(i - j\alpha)f_d^2$
is a square in $\mathcal{O}$,
ring of integers of $\mathbf{Q}(\alpha)$.

Multiply by $g'(f_d\alpha)^2$,
putting square root into $\mathbf{Z}[f_d\alpha]$:
compute r with $r^2 = g'(f_d\alpha)^2 \cdot$
$\prod(i - jm)(i - j\alpha)f_d^2$.

Then apply the ring morphism
$\varphi : \mathbf{Z}[f_d\alpha] \to \mathbf{Z}/n$ taking
$f_d\alpha$ to $f_d m$. Compute gcd$\{n,$
$\varphi(r) - g'(f_d m) \prod(i - jm) f_d\}$.
In $\mathbf{Z}/n$ have $\varphi(r)^2 =$
$g'(f_d m)^2 \prod(i - jm)^2 f_d^2$.

How to find square product of congruences $(i - jm)(i - j\alpha)$?

Start with congruences for, e.g., y^2 pairs (i, j).

Look for y-smooth congruences: y-smooth $i - jm$ and y-smooth $f_d \operatorname{norm}(i - j\alpha) = f_d i^d + \cdots + f_0 j^d = j^d f(i/j)$. Here "$y$-smooth" means "has no prime divisor $> y$."

Find enough smooth congruences. Perform linear algebra on exponent vectors mod 2.

Sublattices

Consider a sublattice
of pairs (i, j) where
q divides $j^d f(i/j)$.

Assume squarish lattice.
$(i - jm)j^d f(i/j)$
expands by factor $q^{(d+1)/2}$
before division by q.

Number of sublattice elements
within any particular bound
on $(i - jm)j^d f(i/j)$
is proportional to $q^{-(d-1)/(d+1)}$.

Compared to just using $q = 1$,
conjecturally obtain $y^{4/(d+1)+o(1)}$
times as many congruences
by using sublattices for
all y-smooth integers $q \le y^2$.

Separately consider
$i - jm$ and $j^d f(i/j)/q$
for more precise analysis.

Limit congruences accordingly,
increasing smoothness chances.

Multiple number fields

Assume that $f + x - m \in \mathbf{Z}[x]$ is also irred.

Pick $\beta \in \mathbf{C}$, root of $f + x - m$.
Two congruences for (i, j):
$(i - jm)(i - j\alpha)$; $(i - jm)(i - j\beta)$.
Expand exponent vectors to handle both $\mathbf{Q}(\alpha)$ and $\mathbf{Q}(\beta)$.

Merge smoothness tests by testing $i - jm$ first, aborting if $i - jm$ not smooth.

Can use many number fields: $f + 2(x - m)$ etc.

Optimizing NFS

Finding smooth congruences is *always* a bottleneck.

"What if it's much faster than linear algebra?"
Answer: If it is, trivially save time by decreasing y.

Optimizing NFS

Finding smooth congruences
is *always* a bottleneck.

"What if it's much faster
than linear algebra?"
Answer: If it is, trivially
save time by decreasing y.

Main job of NFS implementor:
speed up smoothness detection.

Optimizing NFS

Finding smooth congruences is *always* a bottleneck.

“What if it’s much faster than linear algebra?”
Answer: If it is, trivially save time by decreasing y.

Main job of NFS implementor: speed up smoothness detection.

Other ways to speed up NFS: optimize set of pairs (i, j), choice of f, etc. Fun: e.g., compute $\int_{-\infty}^{\infty} \frac{dx}{((x-m)f)^{2/(d+1)}}$.

1977 Schroeppel "linear sieve," forerunner of QS and NFS:
Factor $n \approx s^2$ using congruences $(s+i)(s+j)((s+i)(s+j)-n)$.
Sieve these congruences.

1996 Pomerance:
"The time for doing this is unbelievably fast compared with trial dividing each candidate number to see if it is Y-smooth.
If the length of the interval is N, the number of steps is only about $N \log \log Y$, or about $\log \log Y$ steps on average per candidate."

Asymptotic cost exponents

Number of bit operations
in number-field sieve,
with theorists' parameters,
is $L^{1.90\ldots+o(1)}$ where $L =$
$\exp((\log n)^{1/3}(\log\log n)^{2/3})$.

What are theorists' parameters?

Choose degree d with
$d/(\log n)^{1/3}(\log\log n)^{-1/3}$
$\in 1.40\ldots + o(1)$.

Choose integer $m \approx n^{1/d}$.
Write n as
$m^d + f_{d-1}m^{d-1} + \cdots + f_1 m + f_0$
with each f_k below $n^{(1+o(1))/d}$.
Choose f with some randomness
in case there are bad f's.

Test smoothness of $i - jm$
for all coprime pairs (i, j)
with $1 \le i, j \le L^{0.95\ldots+o(1)}$,
using primes $\le L^{0.95\ldots+o(1)}$.

$L^{1.90\ldots+o(1)}$ pairs.
Conjecturally $L^{1.65\ldots+o(1)}$
smooth values of $i - jm$.

Use $L^{0.12\ldots+o(1)}$ number fields.

For each (i,j)
with smooth $i - jm$,
test smoothness of $i - j\alpha$
and $i - j\beta$ and so on,
using primes $\le L^{0.82\ldots+o(1)}$.

$L^{1.77\ldots+o(1)}$ tests.
Each $|j^d f(i/j)| \le m^{2.86\ldots+o(1)}$.
Conjecturally $L^{0.95\ldots+o(1)}$
smooth congruences.

$L^{0.95\ldots+o(1)}$ components
in the exponent vectors.

Three sizes of numbers here:

$(\log n)^{1/3}(\log\log n)^{2/3}$ bits:
y, i, j.

$(\log n)^{2/3}(\log\log n)^{1/3}$ bits:
m, $i - jm$, $j^d f(i/j)$.

$\log n$ bits: n.

Unavoidably 1/3 in exponent:
usual smoothness optimization
forces $(\log y)^2 \approx \log m$;
balancing norms with m
forces $d \log y \approx \log m$;
and $d \log m \approx \log n$.

Batch NFS

The number-field sieve used $L^{1.90\ldots+o(1)}$ bit operations finding smooth $i - jm$; only $L^{1.77\ldots+o(1)}$ bit operations finding smooth $j^d f(i/j)$.

Many n's can share one m; $L^{1.90\ldots+o(1)}$ bit operations to find squares for *all* n's.

Oops, linear algebra hurts; fix by reducing y.
But still end up factoring batch in much less time than factoring each n separately.

Asymptotic batch-NFS parameters:

$d/(\log n)^{1/3}(\log\log n)^{-1/3}$
$\in 1.10\ldots + o(1)$.

Primes $\le L^{0.82\ldots+o(1)}$.

$1 \le i, j \le L^{1.00\ldots+o(1)}$.

Computation independent of n finds $L^{1.64\ldots+o(1)}$ smooth values $i - jm$.

$L^{1.64\ldots+o(1)}$ operations for each target n.

Batch NFS for RSA-3072

Expand n in base $m = 2^{384}$:
$n = n_7 m^7 + n_6 m^6 + \cdots + n_0$
with $0 \le n_0, n_1, \ldots, n_7 < m$.

Assume irreducibility of
$n_7 x^7 + n_6 x^6 + \cdots + n_0$.

Choose height $H = 2^{62} + 2^{61} + 2^{57}$:
consider pairs $(a, b) \in \mathbf{Z} \times \mathbf{Z}$ such
that $-H \le a \le H$, $0 < b \le H$,
and $\gcd\{a, b\} = 1$.

Choose smoothness bound
$y = 2^{66} + 2^{55}$.

There are about
$12H^2/\pi^2 \approx 2^{125.51}$
pairs (a, b).

Find all pairs (a, b) with
y-smooth $(a - bm)c$ where
$c = n_7 a^7 + n_6 a^6 b + \cdots + n_0 b^7$.

Combine these congruences
into a factorization of n,
if there are enough congruences.

Number of congruences needed
$\approx 2y/\log y \approx 2^{62.06}$.

Heuristic approximation:
$a - bm$ has same y-smoothness chance as a uniform random integer in $[1, Hm]$,
and this chance is u^{-u}
where $u = (\log(Hm))/\log y$.

Have $u \approx 6.707$
and $u^{-u} \approx 2^{-18.42}$,
so there are about
$2^{107.09}$ pairs (a, b)
such that $a - bm$ is smooth.

Heuristic approximation:
c has same y-smoothness chance
as a uniform random integer in
$[1, 8H^7m]$,
and this chance is v^{-v}
where $v = (\log(8H^7m))/\log y$.

Have $v \approx 12.395$
and $v^{-v} \approx 2^{-45.01}$,
so there are about
$2^{62.08}$ pairs (a, b) such that
$a - bm$ and c are both smooth.

Safely above $2^{62.06}$.

Biggest step in computation:
Check $2^{125.51}$ pairs (a, b)
to find the $2^{107.09}$ pairs
where $a - bm$ is smooth.

This step is independent of N,
reused by many integers N.

Biggest step in computation:
Check $2^{125.51}$ pairs (a, b)
to find the $2^{107.09}$ pairs
where $a - bm$ is smooth.

This step is independent of N,
reused by many integers N.

Biggest step depending on N:
Check $2^{107.09}$ pairs (a, b)
to see whether c is smooth.

This is much less
computation! . . . or is it?

The $2^{107.09}$ pairs (a, b)
do not form a lattice,
so no easy way to sieve
for prime divisors of c.

The $2^{107.09}$ pairs (a, b)
do not form a lattice,
so no easy way to sieve
for prime divisors of c.

Fix:
"Factoring into coprimes";
next topic today.

The $2^{107.09}$ pairs (a, b)
do not form a lattice,
so no easy way to sieve
for prime divisors of c.

Fix:
"Factoring into coprimes";
next topic today.

A different fix:
ECM; this afternoon.

Better smoothness estimates

Consider a uniform random integer in $[1, 2^{400}]$.

What is the chance that the integer is 1000000-**smooth**, i.e., factors into primes ≤ 1000000?

"Objection: The integers in NFS are not uniform random integers!" True; will generalize later.

Traditional answer:

Dickman's ρ function is fast.

A uniform random integer in $[1, y^u]$ has chance $\approx \rho(u)$ of being y-smooth.

If u is small then chance/$\rho(u)$ is $1 + O(\log\log y/\log y)$ for $y \to \infty$.

Flaw #1 in traditional answer:
Not a very good approximation.

Flaw #2 in traditional answer:
Not easy to generalize.

Another traditional answer,
trivial to generalize:

Check smoothness of many independent uniform random integers.

Can accurately estimate smoothness probability p after inspecting $10000/p$ integers; typical error $\approx 1\%$.

But this answer is very slow.

Here's a better answer.

(starting point: 1998 Bernstein)

Define S as the set of 1000000-smooth integers $n \geq 1$.

The Dirichlet series for S is $\sum [n \in S] x^{\lg n} =$

$$(1 + x^{\lg 2} + x^{2\lg 2} + x^{3\lg 2} + \cdots)$$
$$(1 + x^{\lg 3} + x^{2\lg 3} + x^{3\lg 3} + \cdots)$$
$$(1 + x^{\lg 5} + x^{2\lg 5} + x^{3\lg 5} + \cdots)$$
$$\cdots$$
$$(1 + x^{\lg 999983} + x^{2\lg 999983} + \cdots).$$

Replace primes

$2, 3, 5, 7, \ldots, 999983$

with slightly larger real numbers

$\overline{2} = 1.1^8$, $\overline{3} = 1.1^{12}$, $\overline{5} = 1.1^{17}$,

$\ldots$, $\overline{999983} = 1.1^{145}$.

Replace each $2^a 3^b \cdots$ in S with

$\overline{2}^a \overline{3}^b \cdots$, obtaining multiset $\bar{S}$.

The Dirichlet series for $\bar{S}$

is $\sum [n \in \bar{S}] x^{\lg n} =$

$$(1 + x^{\lg \overline{2}} + x^{2 \lg \overline{2}} + x^{3 \lg \overline{2}} + \cdots)$$
$$(1 + x^{\lg \overline{3}} + x^{2 \lg \overline{3}} + x^{3 \lg \overline{3}} + \cdots)$$
$$(1 + x^{\lg \overline{5}} + x^{2 \lg \overline{5}} + x^{3 \lg \overline{5}} + \cdots)$$
$$\ldots$$
$$(1 + x^{\lg \overline{999983}} + x^{2 \lg \overline{999983}} + \cdots).$$

This is simply a power series

$s_0 z^0 + s_1 z^1 + \cdots =$

$(1 + z^8 + z^{2\cdot 8} + z^{3\cdot 8} + \cdots)$

$(1 + z^{12} + z^{2\cdot 12} + z^{3\cdot 12} + \cdots)$

$(1 + z^{17} + z^{2\cdot 17} + z^{3\cdot 17} + \cdots)$

$\cdots (1 + z^{145} + z^{2\cdot 145} + \cdots)$

in the variable $z = x^{\lg 1.1}$.

Compute series mod (e.g.) z^{2910};
i.e., compute $s_0, s_1, \ldots, s_{2909}$.
$\bar{S}$ has $s_0 + \cdots + s_{2909}$ elements
$\le 1.1^{2909} < 2^{400}$, so S has
at least $s_0 + \cdots + s_{2909}$
elements $< 2^{400}$.

So have guaranteed lower bound on number of 1000000-smooth integers in $[1, 2^{400}]$.

Can compute an upper bound to check looseness of lower bound.

If looser than desired,
move 1.1 closer to 1.
Achieve any desired accuracy.

2007 Parsell–Sorenson: Replace big primes with RH bounds, faster to compute.

NFS smoothness is much more complicated than smoothness of uniform random integers.

Most obvious issue: NFS doesn't use *all* integers in $[-H, H]$;
it uses only values $f(c, d)$
of a specified polynomial f.

Traditional reaction
(1979 Schroeppel, et al.):
replace H by "typical" f value,
heuristically adjusted for
roots of f mod small primes.

Can compute smoothness chance much more accurately.
No need for “typical” values.

We’ve already computed series
$s_0 z^0 + s_1 z^1 + \cdots + s_{2909} z^{2909}$
such that there are
$\ge s_0$ smooth$\le 1.1^0$,
$\ge s_0 + s_1$ smooth$\le 1.1^1$,
$\ge s_0 + s_1 + s_2$ smooth$\le 1.1^2$,
$\vdots$,
$\ge s_0 + \cdots + s_{2909}$ smooth$\le 1.1^{2909}$.
Approximations are very close.

Number of $f(c,d)$ values in $[-H,H]$ is $\approx (3/\pi^2)H^{2/\deg f}Q(f)$. Can quickly compute $Q(f)$.

For each $i \le 2909$, number of smooth $|f(c,d)|$ values in $[1.1^{i-1}, 1.1^{i}]$ is approximately

$$\frac{3Q(f)s_i}{\pi^2}\,\frac{1.1^{2i/\deg f} - 1.1^{2(i-1)/\deg f}}{1.1^{i} - 1.1^{i-1}}.$$

Add to see total number of smooth $f(c,d)$ values.

Approximation so far
has ignored roots of f.

Fix: Smoothness chance in $\mathbf{Q}(\alpha)$ for $c - \alpha d$ is, conjecturally, very close to smoothness chance for ideals of the same size as $c - \alpha d$.

Dirichlet series for smooth ideals: simply replace
$1 + x^{\lg p} + x^{2 \lg p} + \cdots$ with
$1 + x^{\lg P} + x^{2 \lg P} + \cdots$
where P is norm of prime ideal.

Same computations as before.
Should also be easy to adapt Parsell–Sorenson to ideals.

Typically $f(c, d)$ is product $(c - md) \cdot$ norm of $(c - \alpha d)$.

Smoothness chance in $\mathbf{Q} \times \mathbf{Q}(\alpha)$ for $(c - md, c - \alpha d)$ is, conjecturally, close to smoothness chance for ideals of the same size.

Can account in various ways for correlations and anti-correlations between $c - md$ and $c - \alpha d$, but these effects seem small.

Dirichlet-series computations easily handle early aborts and other complications in the notion of smoothness.

Example: Which integers are 1000000-smooth integers $< 2^{400}$ times one prime in $[10^6, 10^9]$?
Multiply $s_0 z^0 + \cdots + s_{2909} z^{2909}$ by $x^{\lg \overline{1000003}} + \cdots + x^{\lg \overline{999999937}}$.

Polynomial selection

Many f's possible for n.
How to find f that
minimizes NFS time?

General strategy:
Enumerate many f's.
For each f, estimate time using
information about f arithmetic,
distribution of $d^{\deg f} f(c/d)$,
distribution of smooth numbers.

Let's restrict attention to $f(x) = (x - m)(f_5 x^5 + f_4 x^4 + \cdots + f_0)$.

Take m near $n^{1/6}$.
Expand n in base m:
$n = f_5 m^5 + f_4 m^4 + \cdots + f_0$.
Can use negative coefficients.

Have $f_5 \approx n^{1/6}$.
Typically all the f_i's
are on scale of $n^{1/6}$.

(1993 Buhler Lenstra Pomerance)

To reduce f values by factor B:

Enumerate many possibilities for m near $B^{0.25}n^{1/6}$.

Have $f_5 \approx B^{-1.25}n^{1/6}$.
f_4, f_3, f_2, f_1, f_0 could be as large as $B^{0.25}n^{1/6}$.
Hope that they are smaller, on scale of $B^{-1.25}n^{1/6}$.

Conjecturally this happens within roughly $B^{7.5}$ trials.
Then $(c - dm)(f_5c^5 + \cdots + f_0d^5)$ is on scale of $B^{-1}R^6n^{2/6}$ for c, d on scale of R.

Can force f_4 to be small.

Say $n = f_5 m^5 + f_4 m^4 + \cdots + f_0$.

Choose integer $k \approx f_4/5f_5$.
Write n in base $m + k$:

$$n = f_5(m+k)^5 \\ \qquad + (f_4 - 5kf_5)(m+k)^4 + \cdots.$$

Now degree-4 coefficient
is on same scale as f_5.

Hope for small f_3, f_2, f_1, f_0.
Conjecturally this happens
within roughly B^6 trials.

Improvement:
Skew the coefficients.
(1999 Murphy, without analysis)

Enumerate many possibilities
for m near $Bn^{1/6}$.

Have $f_5 \approx B^{-5} n^{1/6}$.
f_4, f_3, f_2, f_1, f_0 could be
as large as $Bn^{1/6}$.

Force small f_4. Hope for
f_3 on scale of $B^{-2} n^{1/6}$,
f_2 on scale of $B^{-0.5} n^{1/6}$.

Conjecturally this happens within roughly $B^{4.5}$ trials: $(2+1)+(0.5+1)=4.5$.

For c on scale of $B^{0.75}R$ and d on scale of $B^{-0.75}R$, have $c-md$ on scale of $B^{0.25}Rn^{1/6}$ and $f_5c^5+f_4c^4d+\cdots+f_0d^5$ on scale of $B^{-1.25}R^5n^{1/6}$.

Product $B^{-1}R^6n^{2/6}$.

Similar effect of B on $Q(f)$; can afford to compute Q for many attractive f's.

Can we do better? Yes!

The following algorithm:
only about $B^{3.5}$ trials,
conjecturally.

Each trial is fairly expensive,
using four-dimensional
integer-relation finding,
but worthwhile for large B.

This is so fast that
we should start searching
$(m_2x - m_1)(c_5x^5 + c_4x^4 + \cdots + c_0)$.

Say $n = f_5 m^5 + f_4 m^4 + \cdots + f_0$.

Choose integer $k \approx f_4/5f_5$
and integer $\ell \approx m/5f_5$.

Find all short vectors
in lattice generated by
$(m/B^3, 0, 0, 10f_5k^2 - 4f_4k + f_3)$,
$(0, m/B^4, 0, 20f_5k\ell - 4f_4\ell)$,
$(0, 0, m/B^5, 10f_5\ell^2)$,
$(0, 0, 0 \qquad , m)$.

Hope for j below B^1
with $(10f_5k^2 - 4f_4k + f_3)$
$+ (20f_5k\ell - 4f_4\ell)j$
$+ (10f_5\ell^2)j^2$
below m/B^3 modulo m.

Write n in base $m + k + j\ell$.
Obtain degree-5 coefficient
on scale of $B^{-5}n^{1/6}$;
degree-4 coefficient
on scale of $B^{-4}n^{1/6}$;
degree-3 coefficient
on scale of $B^{-2}n^{1/6}$.
Hope for good degree 2.

Bad news, part 1:
All known search methods,
including this one,
become ineffective
as degree increases.

Bad news, part 2:
In batch-NFS context,
searching large m pool
requires scaling up # targets.