Simplified high-speed high-distance list decoding for alternant codes

cr.yp.to/papers.html#simplelist

D. J. Bernstein
University of Illinois at Chicago

Thanks to: Cisco University Research Program

And thanks to: NIST grant 60NANB10D263

Alternant codes

Parameters:

- q: prime power.
- m, n, t: positive integers with $mt \leq n \leq q^m$.
- Distinct $\alpha_1, \ldots, \alpha_n \in \mathbf{F}_{q^m}$.
- Nonzero $\beta_1, \ldots, \beta_n \in \mathbf{F}_{q^m}$.

$C = \mathbf{F}_q^n \cap R$ where $R =$
$\{(\beta_1 f(\alpha_1), \ldots, \beta_n f(\alpha_n)) \in \mathbf{F}_{q^m}^n :$
$f \in \mathbf{F}_{q^m}[x];$ $\deg f < n - t\}$.

C is an $[n, \geq n - mt, \geq t + 1]$ linear code over $\mathbf{F}_q$.

(1974 Helgert, 1975 Chien–Choy, 1975 Delsarte)

Goal: Correct w errors in C.

Assume $q(n/t) \lg q^m \in (\lg n)^{O(1)}$.

Any $w \leq \lfloor t/2 \rfloor$, cost $n^{O(1)}$:

1960 Peterson.

Goal: Correct w errors in C.
Assume $q(n/t) \lg q^m \in (\lg n)^{O(1)}$.

Any $w \leq \lfloor t/2 \rfloor$, cost $n^{O(1)}$:
1960 Peterson. Big speedups—
$n^2(\lg n)^{O(1)}$: 1968 Berlekamp;
$n(\lg n)^{O(1)}$, using FFT etc.:
1976 Justesen, 1977 Sarwate.

Goal: Correct w errors in C.
Assume $q(n/t) \lg q^m \in (\lg n)^{O(1)}$.

Any $w \leq \lfloor t/2 \rfloor$, cost $n^{O(1)}$:
1960 Peterson. Big speedups—
$n^2(\lg n)^{O(1)}$: 1968 Berlekamp;
$n(\lg n)^{O(1)}$, using FFT etc.:
1976 Justesen, 1977 Sarwate.

Any $w < n - \sqrt{n(n-t-1)}$,
cost $n^{O(1)}$: 1998 Guruswami–
Sudan, improving on 1997 Sudan.
Many subsequent speedups.

Goal: Correct w errors in C.
Assume $q(n/t) \lg q^m \in (\lg n)^{O(1)}$.

Any $w \leq \lfloor t/2 \rfloor$, cost $n^{O(1)}$:
1960 Peterson. Big speedups—
$n^2(\lg n)^{O(1)}$: 1968 Berlekamp;
$n(\lg n)^{O(1)}$, using FFT etc.:
1976 Justesen, 1977 Sarwate.

Any $w < n - \sqrt{n(n-t-1)}$,
cost $n^{O(1)}$: 1998 Guruswami–
Sudan, improving on 1997 Sudan.
Many subsequent speedups.

Any $w < n' - \sqrt{n'(n'-t-1)}$,
cost $n^{O(1)}$: 2000 Koetter–Vardy.
Here $n' = n(q-1)/q$.

Example of recent speedups,
JSC 2010 Beelen–Brander:
any $w < n - \sqrt{n(n-t-1)}$,
cost $\ell^5 n (\lg n)^{O(1)}$
for output list size ℓ.

Example of recent speedups,
JSC 2010 Beelen–Brander:
any $w < n - \sqrt{n(n-t-1)}$,
cost $\ell^5 n (\lg n)^{O(1)}$
for output list size ℓ.

2011 Bernstein "Simplified high-speed high-distance list decoding for alternant codes":
any $w < n' - \sqrt{n'(n'-t-1)}$,
cost $\ell^{<3.5} n (\lg n)^{O(1)}$
for output list size ℓ.

Cost $O(n^{<4.5})$ for any
$w < n' - \sqrt{n'(n'-t-1)}$
$+ o((\lg n)/\lg\lg n)$.

CRT codes

Fix distinct primes $p_1, \ldots, p_n$
and a positive integer H.

For $f \in \mathbf{Z}$ define $\mathrm{ev}(f) =$
$(f \bmod p_1, \ldots, f \bmod p_n)$.
The CRT code for $p_1, \ldots, p_n, H$
is $C = \{\mathrm{ev}(f) : f \in \mathbf{Z}, |f| \leq H\}$.

CRT codes

Fix distinct primes $p_1, \ldots, p_n$
and a positive integer H.

For $f \in \mathbf{Z}$ define $\mathrm{ev}(f) =$
$(f \bmod p_1, \ldots, f \bmod p_n)$.
The CRT code for $p_1, \ldots, p_n, H$
is $C = \{\mathrm{ev}(f) : f \in \mathbf{Z}, |f| \leq H\}$.

What you're probably thinking:
"Yeah, I know this pointless
number-theoretic analogue
of Reed–Solomon codes.
Anything you can do with these,
you can do better with RS."

One standard multiplicity-2
CRT list-decoding algorithm:

Receive word $(r_1, \ldots, r_n)$.

Find small nonzero $Q \in \mathbf{Z}[y]$
having multiplicity ≥ 2
at each (p_i, r_i): i.e.,
$Q \in (p_i\mathbf{Z}[y] + (y - r_i)\mathbf{Z}[y])^2 =$
$p_i^2\mathbf{Z}[y]+p_i(y-r_i)\mathbf{Z}[y]+(y-r_i)^2\mathbf{Z}[y]$.

Find all $f \in \mathbf{Z}$, $|f| \leq H$
such that $y - f$ divides Q,
i.e., $Q(f) = 0$.

Fact: This finds all f with
$\mathrm{ev}(f)$ close to $(r_1, \ldots, r_n)$.

List-size-3 definition of "small":

$\deg Q \le 3$: i.e.,
$Q = Q_0 + Q_1 y + Q_2 y^2 + Q_3 y^3$
for some $Q_0, Q_1, Q_2, Q_3 \in \mathbf{Z}$;

and coefficients are small: i.e.,
$|Q_0| \le 2P^{3/4}H^{3/2}$,
$|Q_1| \le 2P^{3/4}H^{1/2}$,
$|Q_2| \le 2P^{3/4}H^{-1/2}$,
$|Q_3| \le 2P^{3/4}H^{-3/2}$,
where $P = p_1 \cdots p_n$.

Then $|Q(f)| \le 8P^{3/4}H^{3/2}$
but $Q(f) \in \prod_{i:f \bmod p_i = r_i} p_i^2 \mathbf{Z}$.
$Q(f)$ must be 0 if
$\prod_{i:f \bmod p_i = r_i} p_i^2 > 8P^{3/4}H^{3/2}$.

Can start with generators
$p_i^2, p_i(y-r_i), (y-r_i)^2, y(y-r_i)^2$
for the lattice
$L_i = \{Q \in \mathbf{Z} + \mathbf{Z}y + \mathbf{Z}y^2 + \mathbf{Z}y^3 :$
$Q \in (p_i\mathbf{Z}[y] + (y-r_i)\mathbf{Z}[y])^2\}.$

Obtain generators for lattice
$L = \{Q \in \mathbf{Z} + \mathbf{Z}y + \mathbf{Z}y^2 + \mathbf{Z}y^3 :$
$\forall i : Q \in (p_i\mathbf{Z}[y] + (y-r_i)\mathbf{Z}[y])^2\}$
$= \bigcap_i L_i$ using standard methods.

Find small nonzero $Q \in L$
by lattice-basis reduction (LLL).

FOCS 2000 Guruswami–Sahai–Sudan: this algorithm (for arbitrary multiplicities, list size).

Simpler, faster, more streamlined construction of same lattice:

Start with 0-error interpolation—
i.e., compute $R \in \mathbf{Z}$ such that
$R \bmod p_i = r_i$ for all i.

Write down generators for $L^\perp$:
L is exactly the set of
$Q_0 + Q_1 y + Q_2 y^2 + Q_3 y^3 \in \mathbf{Z}[y]$
such that
$Q_0 + Q_1 R + Q_2 R^2 + Q_3 R^3 \in P^2\mathbf{Z}$
and $Q_1 + 2Q_2 R + 3Q_3 R^2 \in P\mathbf{Z}$.

From these linear equations
find generators for L
using standard methods.

Even simpler, even faster,

even more streamlined:

Write down generators for L.

$$L = P^2\mathbf{Z} + P(y-R)\mathbf{Z} \\ + (y-R)^2\mathbf{Z} + y(y-R)^2\mathbf{Z}.$$

Find small nonzero $Q \in L$.

Find all $f \in \mathbf{Z}$ with $Q(f) = 0$.

Even simpler, even faster,
even more streamlined:
Write down generators for L.

$$L = P^2\mathbf{Z} + P(y-R)\mathbf{Z} + (y-R)^2\mathbf{Z} + y(y-R)^2\mathbf{Z}.$$

Find small nonzero $Q \in L$.
Find all $f \in \mathbf{Z}$ with $Q(f) = 0$.

1997 Howgrave-Graham:
exactly this algorithm (for
arbitrary multiplicity, list size)
to find all big $f - R$ dividing P.

STOC 2000 Boneh:
This finds all big $\gcd\{P, f - R\}$.
Use for CRT list decoding.

Sensible alternant list decoding

Use fast multiplication: 1866 Gauss, 1963 Karatsuba, etc.

Use fast root-finding for Q: 1969 Zassenhaus.

Use fast lattice-basis reduction: 2003 Giorgi–Jeannerod–Villard.

Increase multiplicity as needed: 1996 Coppersmith.

Write down lattice generators directly: 1997 Howgrave-Graham.

Tweak lattice to correct more errors: 2000 Koetter–Vardy.

Main contribution of this paper: writing down generators directly for the Koetter–Vardy lattice. Brings the Howgrave-Graham simplicity and speed up to $n' - \sqrt{n'(n' - t - 1)}$ errors.

For "wild Goppa codes": correct more errors via 1975 Sugiyama–Kasahara–Hirasawa–Namekawa.

Main contribution of this paper: writing down generators directly for the Koetter–Vardy lattice. Brings the Howgrave-Graham simplicity and speed up to $n' - \sqrt{n'(n'-t-1)}$ errors.

For "wild Goppa codes": correct more errors via 1975 Sugiyama–Kasahara–Hirasawa–Namekawa.

Is this the end? No!
Want "rational" list decoding—reuse partial information from attempted unique decoding; reduces multiplicities; faster!

2003 Bernstein: rational curve-fitting algorithm with multiplicities, but no applications.

Rational list-decoding algorithms:

2007 Wu: $n - \sqrt{n(n-t-1)}$ for Reed–Solomon. BCH: more.

2008 Bernstein "List decoding for binary Goppa codes": $n - \sqrt{n(n-2t-1)}$ for classical degree-t irreducible binary Goppa.

2011 Bernstein "Jet list decoding": $n' - \sqrt{n'(n'-2t-1)}$ for the same Goppa codes.
Jets should also work for AG.