Chinese Remaindering with Errors

Oded Goldreich
Weizmann Institute

Dana Ron
Tel Aviv U.

Madhu Sudan
MIT

Chinese Remainder Theorem

Notation:

– $p_1, \ldots, p_k$: relatively prime.
– $K \stackrel{\text{def}}{=} p_1 \times p_2 \times \cdots \times p_k$
– $(\mathbb{Z}_N \stackrel{\text{def}}{=} \{0, \ldots, N-1\})$

Thm: Given $m_1, \ldots, m_k; \quad m_i \in \mathbb{Z}_{p_i};$
$\exists\,!\; m \in \mathbb{Z}_K$ s.t. $m_i = m \bmod p_i$

(i.e., $\mathbb{Z}_K \sim \mathbb{Z}_{p_1} \times \cdots \mathbb{Z}_{p_k}$)

- Algorithmic conversion.
- Simplifies $+, -, \times$ over $\mathbb{Z}_K$.
- Applications in complexity.

Redundancy from CRT

CRT Code:

- $p_1 < p_2 < \cdots p_k < p_{k+1} < \cdots p_n$
- $m \in \mathbb{Z}_K$ represented as

$$\langle m_1, m_2, \ldots, m_n \rangle$$

$$m_i = m (\bmod\ p_i)$$

- Any k out of n residues specify m.
  - Used in probabilistic comm. complexity, Karp-Rabin pattern matching.
- Tolerates up to $\frac{n-k}{2}$ errors.
  - Algorithmic! [Mandelbaum '78]

Comparison with polynomial codes

- Two representations of $p(x) = \sum_{i=0}^{k-1} c_i x^i$

$$
\begin{array}{ccc}
\langle c_0, c_1, \ldots, c_{k-1} \rangle & \sim & \langle p(x_1), \ldots, p(x_k) \rangle \\
\downarrow & & \downarrow \\
p(x) \bmod \prod_{i=1}^{k}(x - x_i) & \sim & \langle p(x) \bmod x - x_1, \\
 & & p(x) \bmod x - x_2, \\
 & & \vdots \\
 & & p(x) \bmod x - x_k \rangle
\end{array}
$$

- CRT over $F[x] \sim$ Interpolation

Comparison with Polynomial Codes

- Reed-Solomon codes:
  Message: $p(x) = \sum_{i=1}^{k} c_i x^{i-1}$
  Encoding: Evaluate at $n > k$ places:
  $$p(x_1), \ldots, p(x_k), p(x_{k+1}), \ldots, p(x_n)$$

- RS codes: $F = \{0,1\}^{\log n}$
  - Encode k el'ts of $\{0,1\}^{\log n}$.
  - Using n el'ts of $\{0,1\}^{\log n}$.
  - Any k el'ts out of n specify message.

- CRT: $p_1, \ldots, p_n \in \{n \log n, \ldots, 2n \log n\}$
  - Encode k el'ts of $\{0,1\}^{\log n + \log\log n}$.
  - Using n el'ts of $\{0,1\}^{\log n + \log\log n + 1}$.
  - Any k el'ts out of n specify message.

Comparison with polynomials (contd.)

Algorithms	RS	CRT
Encoding	$\tilde{O}(n)$	$\tilde{O}(n)$
Decoding (no errors)	$\tilde{O}(n)$	$\tilde{O}(n)$
Decoding ($(n-k)/2$ errors)	$\tilde{O}(n)$	$\tilde{O}(n)$
Decoding ($(n-o(n))$ errors)	$\mathrm{poly}(n)$	?

CRT decoding problem

Given:

- $p_1, \ldots, p_n$
- $r_1, \ldots, r_n$; $r_i \in \mathbb{Z}_{p_i}$
- Integers K, t

Task: Find all integers

$m \in \mathbb{Z}_K$ s.t.

$m \bmod p_i = r_i$

for at least t values of $i \in \{1, \ldots, n\}$

Results

Algorithms	RS	CRT
Encoding	$\tilde{O}(n)$	$\tilde{O}(n)$
Decoding (no errors)	$\tilde{O}(n)$	$\tilde{O}(n)$
Decoding ($(n-k)/2$ errors)	$\tilde{O}(n)$	$\tilde{O}(n)$
Decoding ($(n-o(n))$ errors)	$\mathrm{poly}(n)$	$\mathrm{poly}(n)$

Review of Polynomial Decoding

Given:

- $x_1, \ldots, x_n \in F$ distinct.
- $r_1, \ldots, r_n \in F$.
- Integers k, t

Task: Find all poly p of deg. $k-1$ s.t.

$p(x_i) = r_i$

for at least t values of $i \in \{1, \ldots, n\}$.

Review of Polynomial Decoding (contd.)

[Peterson60, Berlekamp66, Massey66, Welch-Bkmp86]

Key concept: Error locator polynomial

$$Y(x) \text{ s.t. } Y(x_i) = 0 \text{ if } p(x_i) \neq r_i$$

1. Y has low-degree ($\leq n - t$)

2. $Z = Y.p$ has low-degree ($\leq n - t + k - 1$)

3. $\forall i, \quad Z(x_i) = Y(x_i).p(x_i) = Y(x_i).r_i$

(Can ignore all references to p above and look for Y, Z.)

Review of Polynomial Decoding (contd.)

I. Find (Y, Z) s.t.

- $Y \not\equiv 0$
- $\deg Y \leq n - t$
- $\deg Z \leq n - t + k - 1$
- $\forall i, \quad Z(x_i) = Y(x_i).r_i$

(May use linear algebra)

II. Output $\frac{Z(x)}{Y(x)}$.

RS to CRT dictionary

- p polynomial $\longrightarrow$ m integer
- $p(x_i)$ $\longrightarrow$ $m \bmod p_i$
- p low degree $\longrightarrow$ m small

CRT algorithm

I. Find integers y, z s.t.

- $1 \leq y \leq Y_{\max}$
- $0 \leq z \leq Z_{\max}$
- $\forall i, \quad z = y.r_i \pmod{p_i}$

II. Output $\frac{z}{y}$.

($Y_{\max}, Z_{\max}$ to be set later.)

Usual questions and answers

1. Why does (y, z) exist?

2. How do we find them?

3. (y, z) may not be unique.
   Is z/y same for all of them?

Usual Answer (to 1.)

Let $m \in \mathbb{Z}_K$ s.t.

$$\langle m \bmod p_1, \ldots, m \bmod p_n \rangle$$

close to $\langle r_1, \ldots, r_n \rangle$

Let $I \stackrel{\text{def}}{=} \{i | r_i \neq m \bmod p_i\}$ and

$$y \ \stackrel{\text{def}}{=} \ \prod_{i \in I} p_i \ ; \ z \ \stackrel{\text{def}}{=} \ y.m$$

Then y small, z small and

$$\forall i, \quad z = y.r_i (\bmod p_i)$$

Any (y, z) will do

Lemma:

- Let (y, z) be any output of I.
- Let $m \in \mathbb{Z}_K$ s.t.

$$\langle m \bmod p_1, \ldots, m \bmod p_n \rangle$$

  close to $\langle r_1, \ldots, r_n \rangle$
- Then $\frac{z}{y} = m$ (or $z - m.y = 0$).

Proof: $m \bmod p_i = r_i \;\Rightarrow\; z = m.y (\bmod p_i)$

$$\Rightarrow z = m.y (\bmod \prod_{i \notin I} p_i)$$

But $z,\; m.y < \prod_{i \notin I} p_i$, so $z = m.y$.

How to find (y, z)?

- Recall: Want y, z small and

$$\forall i, \quad z = y.r_i \pmod{p_i} \tag{1}$$

  Not the usual linear algebra!

- Let $N \stackrel{\text{def}}{=} \prod_{i=1}^{n} p_i$ and
  $r \in \mathbb{Z}_N$ s.t. $r \bmod p_i = r_i$.

- Then Eq. (1) wants $z = y.r \bmod N$.

Integer Program for I.

- Task:

$$\begin{aligned}&\text{Find } y, z, j \text{ s.t.}\\&1 \leq y \leq Y_{\max}\\&0 \leq z \leq Z_{\max}\\&z = yr + jN\end{aligned}$$

  I.P. in 3 variables.

  Solvable in polytime [Lenstra].

Finally ...

(after appropriate setting of $Y_{\max}, Z_{\max}$)

Thm: CRT algorithm corrects

$$(n-k)\frac{\log p_1}{\log p_1 p_n}$$

errors in polynomial time.

Remarks:

1. $p_1, \ldots, p_n \in \{n \log n, \ldots, 2n \log n\}$ implies errors corrected $\sim (n-k)\frac{1}{2+o(1)}$.

2. Running time can be made nearly linear.

Decoding more errors

- Answer m may not be unique!
- Should find a list of all m s.t.

$$m \pmod{p_i} = r_i$$

for t values of $i \in \{1, \ldots, n\}$.

Thm: Can correct approximately $n - 2\sqrt{kn}$ errors in poly time (for $p_1 \geq \frac{1}{2} p_n$).

CRT algorithm revisited

I. Find c_0, c_1 s.t.

$$c_0 + c_1.r = 0 (\mathrm{mod}\ N)$$

(c_0, c_1 small.)

II. Output integer root of

$$c_0 + c_1 x \in \mathbb{Z}[x]$$

- Linear poly $\Rightarrow$ at most one root.
- Higher degree poly $\Rightarrow$ recover more errors?

List-decode CRT

I. Find $c_0, \ldots, c_d$ (small) s.t.

$$\sum_{j=0}^{d} c_j . r^j = 0 (\mathrm{mod}\ N)$$

II. Output all integer roots of

$$C(x) \stackrel{\text{def}}{=} \sum_j c_j x^j \in \mathbb{Z}[x]$$

- Usual questions:
  - Why do $c_0, \ldots, c_d$ exist?
  - How to find them?
  - Why does it help? (in absence of uniqueness)

Existence: **Pigeonhole principle**

- Say $c_i \in \{-L, \ldots, 0, \ldots, L\}$
- $\#$ of tuples $\langle c_0, \ldots, c_d \rangle \sim (2L)^{d+1}$
- Further suppose $(2L)^{d+1} > N (\Leftarrow L > N^{1/d})$
- Then (by PHP) $\exists \langle a_0, \ldots, a_d \rangle, \langle b_0, \ldots, b_d \rangle$ s.t.

$$\sum_j a_j r^j = \sum_j b_j r^j (\mathrm{mod}\ N)$$

$$\Leftrightarrow \sum_j (a_j - b_j) r^j = 0 (\mathrm{mod}\ N)$$

Implying $c_j = a_j - b_j$ gives the desired polynomial with $c_j \in \{-2L, \ldots, 0, \ldots, 2L\}$.

How to find them?

- For constant d can still use IP in fixed dimensions. How about larger d?
- Larger d reduces to Short vector problem in the lattice formed by the columns of:

$$\begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & \ddots & & \\ & & & & 1 & \\ M.1 & M.r & M.r^2 & \cdots & M.r^d & M.N \end{bmatrix}$$

- Can find approx. short vectors in P [LLL]. This suffices.

Why does it help?

1. c_j's small.

2. m small

3. Agreement between m and $\langle r_1, \ldots, r_n \rangle$ large.

$$A \stackrel{\text{def}}{=} \prod_{\{i \mid m (\bmod\ p_i) = r_i\}} p_i \text{ large.}$$

4. Thus we have a small integer $\sum_j c_j m^j$ that is zero modulo a large integer A and is hence identically zero.

Finally ...

(after appropriate setting of unspecified parameters)

Thm: List-decode CRT corrects approximately $n - 2\sqrt{kn}$ errors (for $p_1 \geq \frac{1}{2}p_n$).

Applications?

1. Secret-sharing?

2. Cryptanalysis? (CRT analog of [Jakobsen '98]'s use of RS-decoding?)

3. Tools for dealing with 2-parameter problems?

   - Discrete log (g, p, x)
     - Hard for random x.
     - Hard for random p?
   - Permanent (M, p)
     - Hard for random M.
     - Hard for random p also! (Using CRT decoding).

Applications (contd).

4. Hard core predicates [Håstad+Näslund]
   - Every bit of $ax+b \mod p$ is secure given $a, b, p, f(x)$, for owf f

5. Finding smooth numbers [Boneh].

Recent Improvements

- [Boneh] corrects from $\sqrt{kn(\log p_n / \log p_1)}$ errors. (Bound in this talk was possibly twice as large.)

- [Guruswami, Sahai, S.] correct from $\sqrt{kn} + O(1)$ errors. Also some results for weighted Chinese remaindering with errors.