Adaptive decoding for dense and sparse evaluation/interpolation codes

Clément PERNET
INRIA/LIG-MOAIS, Grenoble Université
joint work with
M. Comer, E. Kaltofen, J-L. Roch, T. Roche

Séminaire Calcul formel et Codes, IRMAR, Rennes
23 Novembre, 2012

Outline

Introduction

- High performance exact computations
- Chinese remaindering
- Motivation

Sparse Interpolation with errors

- Berlekamp/Massey algorithm with errors
- Sparse Polynomial Interpolation with errors
- Relations to Reed-Solomon decoding

Dense interpolation with errors

- Decoding CRT codes: Mandelbaum algorithm
- Amplitude codes
- Adaptive decoding
- Experiments

Outline

Introduction
- High performance exact computations
- Chinese remaindering
- Motivation

Sparse Interpolation with errors
- Berlekamp/Massey algorithm with errors
- Sparse Polynomial Interpolation with errors
- Relations to Reed-Solomon decoding

Dense interpolation with errors
- Decoding CRT codes: Mandelbaum algorithm
- Amplitude codes
- Adaptive decoding
- Experiments

High Performance Algebraic Computations (HPAC)

Domain of Computation

- $\mathbb{Z}, \mathbb{Q}$ $\Rightarrow$variable size
- $\mathbb{Z}_p, \mathsf{GF}(p^k)$ $\Rightarrow$specific arithmetic
- $K[X]$ for $K = \mathbb{Z}_p, \ldots$.

High Performance Algebraic Computations (HPAC)

Domain of Computation

- $\mathbb{Z}, \mathbb{Q}$ $\Rightarrow$variable size
- $\mathbb{Z}_p$, GF(p^k) $\Rightarrow$specific arithmetic
- $K[X]$ for $K = \mathbb{Z}_p, \ldots$.

Application domains:

Computational number theory:

- computing tables of elliptic curves, modular forms,
- testing conjectures

Crypto: Algebraic attacks (Quadratic sieves, Groebner bases, index calculus,...)

Graph theory: testing conjectures (graph isomorphism,...)

Representation theory

...

HPAC: rules of thumb

Deal with size of arithmetic

Evaluation/interpolation schemes:

over $\mathbb{Z}$: Chinese Remainder Algorithm:
$\mathbb{Z} \to \mathbb{Z}/m\mathbb{Z} \to \mathbb{Z}/m_1\mathbb{Z} \times \cdots \times \mathbb{Z}/m_k\mathbb{Z}$

over $K[X]$: Evaluation/interpolation: $K[X] \to K$

- Embarassingly parallel

Lifting schemes $\mathbb{Z} \to \mathbb{Z}/p^k\mathbb{Z} \to \mathbb{Z}/p\mathbb{Z}$

- Best sequential complexities

Deal with complexity/efficiency: reduce to Linear algebra

- Matrix product over $\mathbb{Z}_p, K$
- Eliminations: Gauss, Gram-Schmidt (LLL), ...
- Krylov iteration

HPAC: rules of thumb

Deal with size of arithmetic

Evaluation/interpolation schemes:

over $\mathbb{Z}$: Chinese Remainder Algorithm:
$\mathbb{Z} \to \mathbb{Z}/m\mathbb{Z} \to \mathbb{Z}/m_1\mathbb{Z} \times \cdots \times \mathbb{Z}/m_k\mathbb{Z}$

over $K[X]$: Evaluation/interpolation: $K[X] \to K$

- Embarassingly parallel

Lifting schemes $\mathbb{Z} \to \mathbb{Z}/p^k\mathbb{Z} \to \mathbb{Z}/p\mathbb{Z}$

- Best sequential complexities

Deal with complexity/efficiency: reduce to Linear algebra

- Matrix product over $\mathbb{Z}_p, K$
- Eliminations: Gauss, Gram-Schmidt (LLL), ...
- Krylov iteration

Chinese remainder algorithm

If $m_1, \ldots, m_k$ pariwise relatively prime:

$$\mathbb{Z}/(m_1 \ldots m_k)\mathbb{Z} \equiv \mathbb{Z}/m_1\mathbb{Z} \times \cdots \times \mathbb{Z}/m_k\mathbb{Z}$$

Computation of $y = f(x)$ for $f \in \mathbb{Z}[X], x \in \mathbb{Z}^m$

begin
- Compute an upper bound β on $|f(x)|$;
- Pick $m_1, \ldots m_k$, pairwise prime, s.t. $m_1 \ldots m_k > \beta$;
- **for** $i = 1 \ldots k$ **do**
  - Compute $y_i = f(x \mod m_i) \mod m_i$
- Compute $y = \text{CRT}(y_1, \ldots, y_k)$

$$\begin{array}{rccl} \text{CRT}: & \mathbb{Z}/m_1\mathbb{Z} \times \cdots \times \mathbb{Z}/m_k\mathbb{Z} & \to & \mathbb{Z}/(m_1 \ldots m_k)\mathbb{Z} \\ & (x_1, \ldots, x_k) & \mapsto & \sum_{i=1}^k x_i \Pi_i Y_i \mod \Pi \end{array}$$

where $\begin{cases} \Pi = \prod_{i=1}^k m_i \\ \Pi_i = \Pi/m_i \\ Y_i = \Pi_i^{-1} \mod m_i \end{cases}$

Chinese remainder algorithm

If $m_1, \ldots, m_k$ pariwise relatively prime:

$$\mathbb{Z}/(m_1 \ldots m_k)\mathbb{Z} \equiv \mathbb{Z}/m_1\mathbb{Z} \times \cdots \times \mathbb{Z}/m_k\mathbb{Z}$$

Computation of $y = f(x)$ for $f \in \mathbb{Z}[X], x \in \mathbb{Z}^m$

begin
  Compute an upper bound β on $|f(x)|$;
  Pick $m_1, \ldots m_k$, pairwise prime, s.t. $m_1 \ldots m_k > \beta$;
  for $i = 1 \ldots k$ **do**
    Compute $y_i = f(x \mod m_i) \mod m_i$; `/* Evaluation */`
  Compute $y = \mathrm{CRT}(y_1, \ldots, y_k)$; `/* Interpolation */`

$$\begin{array}{rrcl} \text{CRT}: & \mathbb{Z}/m_1\mathbb{Z} \times \cdots \times \mathbb{Z}/m_k\mathbb{Z} & \to & \mathbb{Z}/(m_1 \ldots m_k)\mathbb{Z} \\ & (x_1, \ldots, x_k) & \mapsto & \sum_{i=1}^k x_i \Pi_i Y_i \mod \Pi \end{array}$$

where $\begin{cases} \Pi = \prod_{i=1}^k m_i \\ \Pi_i = \Pi/m_i \\ Y_i = \Pi_i^{-1} \mod m_i \end{cases}$

Chinese remaindering and evaluation/interpolation

Evaluate P in a $\leftrightarrow$ Reduce P modulo $X - a$

Chinese remaindering and evaluation/interpolation

Evaluate P in a $\leftrightarrow$ Reduce P modulo $X - a$

Polynomials	
Evaluation: $P \mod X - a$ Evaluate P in a	
Interpolation: $P = \sum_{i=1}^{k} y_i \frac{\prod_{j \neq i}(X - a_j)}{\prod_{j \neq i}(a_i - a_j)}$	

Chinese remaindering and evaluation/interpolation

Evaluate P in a $\leftrightarrow$ Reduce P modulo $X - a$

Polynomials	**Integers**
Evaluation: $P \mod X - a$ Evaluate P in a	$N \mod m$ "Evaluate" N in m
Interpolation: $P = \sum_{i=1}^{k} y_i \frac{\prod_{j \neq i}(X - a_j)}{\prod_{j \neq i}(a_i - a_j)}$	$N = \sum_{i=1}^{k} y_i \prod_{j \neq i} m_j (\prod_{j \neq i} m_j)^{-1 [m_i]}$

Early termination

Classic Chinese remaindering **Deterministic**

- bound β on the result
- Choice of the m_i: such that $m_1 \dots m_k > \beta$

Early termination

Classic Chinese remaindering — Deterministic

- bound β on the result
- Choice of the m_i: such that $m_1 \dots m_k > \beta$

Early termination — Probabilistic Monte Carlo

- For each new modulo m_i:
  - reconstruct $y_i = f(x) \mod m_1 \times \cdots \times m_i$
  - If $y_i == y_{i-1} \quad \Rightarrow$terminated

Advantage:

- Adaptive number of moduli depending on the output value
- Interesting when
  - pessimistic bound: sparse/structured matrices, ...
  - no bound available

Motivation

ABFT: Algorithm Based Fault Tolerance

HPC: clusters, grid, P2P, cloud computing

- Parallelization based on Evaluation/Interpolation scheme

Need to tolerate:

- soft errors (cosmic rays,...)
- malicious corruption

Signal processing

- Sparse polynomial interpolation

Distinction between **noise** and **outliers**

- Symbolic-numeric methods

Dense/Sparse interpolation with errors

Problem 1: Dense interpolation with errors over $\mathbb{Z}$

Given (y_i, m_i) for $i = 1 \dots n$,
Find $Y \in \mathbb{Z}$ such that $Y = y_i \mod m_i$ except on $\leq e$ values.

Problem 2: Sparse interpolation with errors over $K[X]$

Given (y_i, x_i) for $i = 1 \dots n$,
Find a t-sparse poly. f such that $f(x_i) = y_i$ except on $\leq e$ values.

State of the art

Dense interpolation

	Interpolation	Interpolation with errors
over $K[X]$	Lagrange	Generalized Reed-Solomon codes
over $\mathbb{Z}$	CRT	CRT codes

Sparse Interpolation

	Interpolation	Interpolation with errors
over $K[X]$	Ben-Or & Tiwari	?
over $\mathbb{Z}$	?	?

State of the art

Dense interpolation

	Interpolation	Interpolation with errors
over $K[X]$	Lagrange	Generalized Reed-Solomon codes
over $\mathbb{Z}$	CRT	CRT codes

Sparse Interpolation

	Interpolation	Interpolation with errors
over $K[X]$	Ben-Or & Tiwari	?
over $\mathbb{Z}$	?	?

Contribution

Sparse interpolation code over $K[X]$

- lower bound on the necessary number of evaluations
- optimal unique decoding algorihtm
- list decoding variant

Dense interpolation code over $\mathbb{Z}$

- finer bounds on the correction capacity
- adaptive decoding using the best effective redundancy

Outline

Introduction
- High performance exact computations
- Chinese remaindering
- Motivation

Sparse Interpolation with errors
- Berlekamp/Massey algorithm with errors
- Sparse Polynomial Interpolation with errors
- Relations to Reed-Solomon decoding

Dense interpolation with errors
- Decoding CRT codes: Mandelbaum algorithm
- Amplitude codes
- Adaptive decoding
- Experiments

Preliminaries

Linear recurring sequences

Sequence $(a_0, a_1, \ldots, a_n, \ldots)$ such that

$$\forall j \geq 0 \; a_{j+t} = \sum_{i=0}^{t-1} \lambda_i a_{i+j}$$

generating polynomial: $\Lambda(z) = z^t - \sum_{i=0}^{t-1} \lambda_i z^i$
minimal generating polynomial: $\Lambda(z)$ of minimal degree
linear complexity of $(a_i)_i$: the minimal degree of Λ

Hamming weight: $\text{weight}(x) = \#\{i | x_i \neq 0\}$
Hamming distance: $d_H(x, y) = \text{weight}(x - y)$

Berlekamp/Massey algorithm

Input: $(a_0, \ldots, a_{n-1})$ a sequence of field elements.
Output: $\Lambda(z) = \sum_{i=0}^{L_n} \lambda_i z^i$ a monic polynomial of minimal degree $L_n \leq n$ such that $\sum_{i=0}^{L_n} \lambda_i a_{i+j} = 0$ for $j = 0, \ldots, n - L_n - 1$.

- Guarantee : BMA finds Λ of degree t from $\leq 2t$ entries.

Problem Statement

Berlkamp/Massey with errors

Suppose $(a_0, a_1, \dots)$ is linearly generated by $\Lambda(z)$ of degree t where $\Lambda(0) \neq 0$.
Given $(b_0, b_1, \dots) = (a_0, a_1, \dots) + \varepsilon$, where weight$(\varepsilon) \leq E$:

1. How to recover $\Lambda(z)$ and $(a_0, a_1, \dots)$?
2. How many entries required for
   - a unique solution ?
   - a list of solutionse including $(a_0, a_1, \dots)$?

Problem Statement

Berlkamp/Massey with errors

Suppose $(a_0, a_1, \dots)$ is linearly generated by $\Lambda(z)$ of degree t where $\Lambda(0) \neq 0$.
Given $(b_0, b_1, \dots) = (a_0, a_1, \dots) + \varepsilon$, where weight$(\varepsilon) \leq E$:

1. How to recover $\Lambda(z)$ and $(a_0, a_1, \dots)$?
2. How many entries required for
   - a unique solution ?
   - a list of solutionse including $(a_0, a_1, \dots)$?

Coding Theory formulation

Let $\mathcal{C}$ be the set of all sequences of linear complexity t.

1. How to decode $\mathcal{C}$?
2. What are the best correction capacities ?
   - for unique decoding
   - list decoding

How many entries to guarantee uniqueness?

Case $E = 1, t = 2$

$$\begin{array}{c|l} (a_i) & \Lambda(z) \\ (0,\ 1,\ 0,\ \ 1,\ 0,\ \ 1,\ 0,\ -1,\ 0,\ 1,\ 0) & 2 - 2z^2 + z^4 + z^6 \end{array}$$

Where is the error?

How many entries to guarantee uniqueness?

Case $E = 1, t = 2$

$$\begin{array}{ccccccccccc|l} & & & & & (a_i) & & & & & & \Lambda(z) \\ (0, & 1, & 0, & 1, & 0, & 1, & 0, & -1, & 0, & 1, & 0) & 2 - 2z^2 + z^4 + z^6 \\ (0, & 1, & 0, & 1, & 0, & 1, & 0, & \color{red}{1}, & 0, & 1, & 0) & -1 + z^2 \end{array}$$

Where is the error?

How many entries to guarantee uniqueness?

Case $E = 1, t = 2$

$$\begin{array}{ccccccccccc|l}
 & & & & & (a_i) & & & & & & \Lambda(z) \\
(0, & 1, & 0, & 1, & 0, & 1, & 0, & -1, & 0, & 1, & 0) & 2 - 2z^2 + z^4 + z^6 \\
(0, & 1, & 0, & 1, & 0, & 1, & 0, & \color{red}{1}, & 0, & 1, & 0) & -1 + z^2 \\
(0, & 1, & 0, & \color{red}{-1}, & 0, & 1, & 0, & -1, & 0, & 1, & 0) & 1 + z^2
\end{array}$$

Where is the error?

How many entries to guarantee uniqueness?

Case $E = 1, t = 2$

$$
\begin{array}{ccccccccccc|l}
 & & & & & (a_i) & & & & & & \Lambda(z) \\
(0, & 1, & 0, & 1, & 0, & 1, & 0, & -1, & 0, & 1, & 0) & 2 - 2z^2 + z^4 + z^6 \\
(0, & 1, & 0, & 1, & 0, & 1, & 0, & \color{red}{1}, & 0, & 1, & 0) & -1 + z^2 \\
(0, & 1, & 0, & \color{red}{-1}, & 0, & 1, & 0, & -1, & 0, & 1, & 0) & 1 + z^2
\end{array}
$$

Where is the error?

A unique solution is not guaranteed with $t = 2, E = 1$ and $n = 11$

Generalization to any $E \geq 1$

Let $\overline{0} = (\overbrace{0, \ldots, 0}^{t-1 \text{ times}})$. Then

$$s = (\overline{0}, 1, \overline{0}, 1, \overline{0}, 1, \overline{0}, -1)$$

is generated by $z^t - 1$ or $z^t + 1$ up to $E = 1$ error.
Then

$$(\overbrace{s, s, \ldots, s}^{E \text{ times}}, \overline{0}, 1, \overline{0})$$

is generated by $z^t - 1$ or $z^t + 1$ up to E errors.
$\Rightarrow$ambiguity with $n = 2t(2E + 1) - 1$ values.

Generalization to any $E \geq 1$

Let $\bar{0} = (\overbrace{0, \ldots, 0}^{t-1 \text{ times}})$. Then

$$s = (\bar{0}, 1, \bar{0}, 1, \bar{0}, 1, \bar{0}, -1)$$

is generated by $z^t - 1$ or $z^t + 1$ up to $E = 1$ error.
Then

$$(\overbrace{s, s, \ldots, s}^{E \text{ times}}, \bar{0}, 1, \bar{0})$$

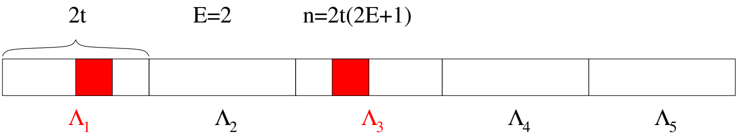
is generated by $z^t - 1$ or $z^t + 1$ up to E errors.
$\Rightarrow$ambiguity with $n = 2t(2E + 1) - 1$ values.

Theorem

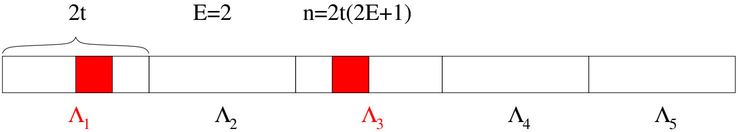
Necessary condition for unique decoding:

$$n \geq 2t(2E + 1)$$

The Majority Rule Berlekamp/Massey algorithm



The Majority Rule Berlekamp/Massey algorithm



Input: $(a_0, \ldots, a_{n-1}) + \varepsilon$, where $n = 2t(2E+1)$, $weight(\varepsilon) \le E$,
and $(a_0, \ldots, a_{n-1})$ minimally generated by Λ of degree t,
where $\Lambda(0) \neq 0$.

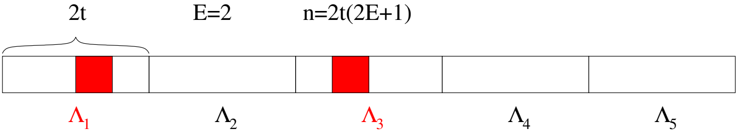
Output: $\Lambda(z)$ and $(a_0, \ldots, a_{n-1})$.

begin

Run BMA on $2E+1$ segments of $2t$ entries and record $\Lambda_i(z)$ on each segment;

Perform majority vote to find $\Lambda(z)$;

The Majority Rule Berlekamp/Massey algorithm



Input: $(a_0, \ldots, a_{n-1}) + \varepsilon$, where $n = 2t(2E+1)$, $weight(\varepsilon) \leq E$,
and $(a_0, \ldots, a_{n-1})$ minimally generated by Λ of degree t,
where $\Lambda(0) \neq 0$.

Output: $\Lambda(z)$ and $(a_0, \ldots, a_{n-1})$.

begin

Run BMA on $2E+1$ segments of $2t$ entries and record $\Lambda_i(z)$ on each segment;

Perform majority vote to find $\Lambda(z)$;

Use a *clean* segment to *clean-up* the sequence ;

return $\Lambda(z)$ and $(a_0, a_1, \ldots)$;

Algorithm `SequenceCleanUp`

Input: $\Lambda(z) = z^t + \sum_{i=0}^{t-1} \lambda_i x^i$ where $\Lambda(0) \neq 0$
Input: $(a_0, \ldots, a_{n-1})$, where $n \geq t+1$
Input: E, the maximum number of corrections to make
Input: k, such that (a_k, a_{k+2t-1}) is clean
Output: $(b_0, \ldots, b_{n-1})$ generated by Λ at distance $\leq E$ to $(a_0, \ldots, a_{n-1})$

Algorithm `SequenceCleanUp`

Input: $\Lambda(z) = z^t + \sum_{i=0}^{t-1} \lambda_i x^i$ where $\Lambda(0) \neq 0$
Input: $(a_0, \ldots, a_{n-1})$, where $n \geq t+1$
Input: E, the maximum number of corrections to make
Input: k, such that (a_k, a_{k+2t-1}) is clean
Output: $(b_0, \ldots, b_{n-1})$ generated by Λ at distance $\leq E$ to $(a_0, \ldots, a_{n-1})$

begin
- $(b_0, \ldots, b_{n-1}) \leftarrow (a_0, \ldots, a_{n-1})$; $e, j \leftarrow 0$;
- $i \leftarrow k + 2t$;
- **while** $i \leq n-1$ *and* $e \leq E$ **do**
  - **if** Λ *does not satisfy* $(b_{i-t+1}, \ldots, b_i)$ **then**
    - Fix b_i using $\Lambda(z)$ as a LFSR; $e \leftarrow e + 1$;
- **return** $(b_0, \ldots, b_{n-1}), e$

Algorithm `SequenceCleanUp`

Input: $\Lambda(z) = z^t + \sum_{i=0}^{t-1} \lambda_i x^i$ where $\Lambda(0) \neq 0$
Input: $(a_0, \ldots, a_{n-1})$, where $n \geq t+1$
Input: E, the maximum number of corrections to make
Input: k, such that (a_k, a_{k+2t-1}) is clean
Output: $(b_0, \ldots, b_{n-1})$ generated by Λ at distance $\leq E$ to $(a_0, \ldots, a_{n-1})$

begin
  $(b_0, \ldots, b_{n-1}) \leftarrow (a_0, \ldots, a_{n-1})$; $e, j \leftarrow 0$;
  $i \leftarrow k + 2t$;
  while $i \leq n-1$ *and* $e \leq E$ **do**
    if Λ *does not satisfy* $(b_{i-t+1}, \ldots, b_i)$ **then**
      Fix b_i using $\Lambda(z)$ as a LFSR; $e \leftarrow e+1$;
  $i \leftarrow k-1$;
  while $i \geq 0$ *and* $e \leq E$ **do**
    if Λ *does not satisfy* $(b_i, \ldots, b_{i+t-1})$ **then**
      Fix b_i using $z^t\Lambda(1/z)$ as a LFSR; $e \leftarrow e+1$;
  return $(b_0, \ldots, b_{n-1}), e$

Algorithm `SequenceCleanUp`

Input: $\Lambda(z) = z^t + \sum_{i=0}^{t-1} \lambda_i x^i$ where $\Lambda(0) \neq 0$
Input: $(a_0, \ldots, a_{n-1})$, where $n \geq t+1$
Input: E, the maximum number of corrections to make
Input: k, such that (a_k, a_{k+2t-1}) is clean
Output: $(b_0, \ldots, b_{n-1})$ generated by Λ at distance $\leq E$ to $(a_0, \ldots, a_{n-1})$

begin
- $(b_0, \ldots, b_{n-1}) \leftarrow (a_0, \ldots, a_{n-1})$; $e, j \leftarrow 0$;
- $i \leftarrow k + 2t$;
- **while** $i \leq n-1$ *and* $e \leq E$ **do**
  - **if** Λ *does not satisfy* $(b_{i-t+1}, \ldots, b_i)$ **then**
    - Fix b_i using $\Lambda(z)$ as a LFSR; $e \leftarrow e+1$;
- $i \leftarrow k-1$;
- **while** $i \geq 0$ *and* $e \leq E$ **do**
  - **if** Λ *does not satisfy* $(b_i, \ldots, b_{i+t-1})$ **then**
    - Fix b_i using $z^t\Lambda(1/z)$ as a LFSR; $e \leftarrow e+1$;
- **return** $(b_0, \ldots, b_{n-1}), e$

Finding a clean segment: case $E = 1$

$\Rightarrow$only one error

$$(a_0, \ldots, a_{k-2}, b_{k-1} \neq a_{k-1}, a_k, a_{k+1}, a_{2t-1})$$

will be identified by the majority vote (2-to-1 majority).

Finding a clean segment: case $E \geq 2$

Multiple errors on one segment can still be generated by $\Lambda(z)$
$\Rightarrow$deceptive segments: not good for `SequenceCleanUp`

Example

$E = 3$: $(0, 1, 0, 2, 0, 4, 0, 8, \dots)$ $\Rightarrow \Lambda(z) = z^2 - 2$

Finding a clean segment: case $E \geq 2$

Multiple errors on one segment can still be generated by $\Lambda(z)$
$\Rightarrow$deceptive segments: not good for `SequenceCleanUp`

Example

$E = 3$: $(0, 1, 0, 2, 0, 4, 0, 8, \dots)$ $\Rightarrow \Lambda(z) = z^2 - 2$

$$(\mathbf{1}, 1, \mathbf{2}, 2, \mathbf{4}, 4, 0, 8, 0, 16, 0, 32, \dots)$$

Finding a clean segment: case $E \geq 2$

Multiple errors on one segment can still be generated by $\Lambda(z)$
$\Rightarrow$deceptive segments: not good for `SequenceCleanUp`

Example

$E = 3$: $(0, 1, 0, 2, 0, 4, 0, 8, \dots)$ $\Rightarrow \Lambda(z) = z^2 - 2$

$$(\underbrace{\mathbf{1}, 1, \mathbf{2}, 2}_{z^2-2}, \underbrace{\mathbf{4}, 4, 0, 8}_{z^2+2z-2}, \underbrace{0, 16, 0, 32}_{z^2-2}, \dots)$$

Finding a clean segment: case $E \geq 2$

Multiple errors on one segment can still be generated by $\Lambda(z)$
$\Rightarrow$deceptive segments: not good for `SequenceCleanUp`

Example

$E = 3$: $(0, 1, 0, 2, 0, 4, 0, 8, \dots)$ $\Rightarrow \Lambda(z) = z^2 - 2$

$$(\underbrace{\mathbf{1}, 1, \mathbf{2}, 2}_{z^2-2}, \underbrace{\mathbf{4}, 4, 0, 8}_{z^2+2z-2}, \underbrace{0, 16, 0, 32}_{z^2-2}, \dots)$$

$(1, 1, 2, 2)$ is deceptive. Applying `SequenceCleanUp` with this clean segment produces

$$(\mathbf{1}, 1, \mathbf{2}, 2, \mathbf{4}, 4, 8, 8, 16, 16, 32, 32, 64, \dots)$$

Finding a clean segment: case $E \geq 2$

Multiple errors on one segment can still be generated by $\Lambda(z)$
$\Rightarrow$deceptive segments: not good for `SequenceCleanUp`

Example

$E = 3$: $(0, 1, 0, 2, 0, 4, 0, 8, \dots)$ $\Rightarrow \Lambda(z) = z^2 - 2$

$$(\underbrace{\mathbf{1}, 1, \mathbf{2}, 2}_{z^2-2}, \underbrace{\mathbf{4}, 4, 0, 8}_{z^2+2z-2}, \underbrace{0, 16, 0, 32}_{z^2-2}, \dots)$$

$(1, 1, 2, 2)$ is deceptive. Applying `SequenceCleanUp` with this clean segment produces

$$(\mathbf{1}, 1, \mathbf{2}, 2, \mathbf{4}, 4, 8, 8, 16, 16, 32, 32, 64, \dots)$$

$E > 3$? contradiction. Try $(0, 16, 0, 32)$ as a clean segment instead.

Success of the sequence clean-up

Theorem

If $n \geq t(2E+1)$, then a deceptive segment will necessarily be exposed by a failure of the condition $e \leq E$ in algorithm `SequenceCleanUp`.

Success of the sequence clean-up

Theorem

If $n \geq t(2E+1)$, then a deceptive segment will necessarily be exposed by a failure of the condition $e \leq E$ in algorithm `SequenceCleanUp`.

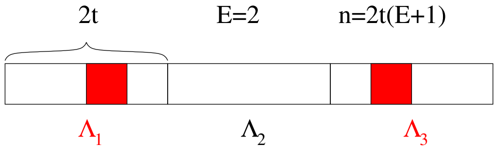
Corollary

$n \geq 2t(2E+1)$ *is a necessary and sufficient condition for unique decoding of Λ and the corresponding sequence.*

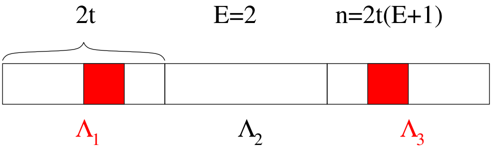
Remark

Also works with an upper bound $t \leq T$ on $\deg \Lambda$.

List decoding for $n \geq 2t(E+1)$



List decoding for $n \geq 2t(E+1)$



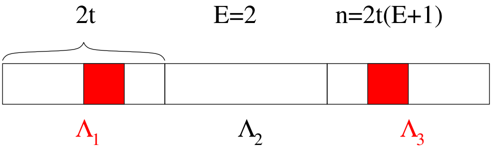
Input: $(a_0, \ldots, a_{n-1}) + \varepsilon$, where $n = 2t(E+1)$, $weight(\varepsilon) \leq E$, and $(a_0, \ldots, a_{n-1})$ minimally generated by Λ of degree t, where $\Lambda(0) \neq 0$.

Output: $(\Lambda_i(z), s_i = (a_0^{(i)}, \ldots, a_{n-1}^{(i)}))_i$ a list of $\leq E$ candidates

begin

Run BMA on $E+1$ segments of $2t$ entries and record $\Lambda_i(z)$ on each segment;

List decoding for $n \geq 2t(E+1)$



Input: $(a_0, \ldots, a_{n-1}) + \varepsilon$, where $n = 2t(E+1)$, $weight(\varepsilon) \leq E$, and $(a_0, \ldots, a_{n-1})$ minimally generated by Λ of degree t, where $\Lambda(0) \neq 0$.

Output: $(\Lambda_i(z), s_i = (a_0^{(i)}, \ldots, a_{n-1}^{(i)}))_i$ a list of $\leq E$ candidates

begin

Run BMA on $E+1$ segments of $2t$ entries and record $\Lambda_i(z)$ on each segment;

foreach $\Lambda_i(z)$ **do**

Use a *clean* segment to *clean-up* the sequence;

Withdraw Λ_i if no clean segment can be found.

return the list $(\Lambda_i(z), (a_0^{(i)}, \ldots, a_{n-1}^{(i)}))_i$;

Properties

- The list contains the right solution $(\Lambda, (a_0, \ldots, a_{n-1}))$

Properties

- The list contains the right solution $(\Lambda, (a_0, \ldots, a_{n-1}))$
- $n \geq 2t(E+1)$ is the tightest bound to ensure to enable syndrome decoding (BMA on a clean sequence of length $2t$).

Example

$n = 2t(E+1) - 1$ and $\varepsilon = (\underbrace{0, \ldots, 0}_{2t-1}, 1, \underbrace{0, \ldots, 0}_{2t-1}, 1 \ldots, 1, \underbrace{0, \ldots, 0}_{2t-1})$.

Then $(a_0, \ldots, a_{n-1}) + \varepsilon$ has no length $2t$ clean segment.

Sparse Polynomial Interpolation

$$f = \sum_{i=1}^{t} c_i x^{e_i}$$

Problem

Recover a t-sparse polynomial f given a black-box computing evaluations of it.

Sparse Polynomial Interpolation

$$f = \sum_{i=1}^{t} c_i x^{e_i}$$

Problem

Recover a t-sparse polynomial f given a black-box computing evaluations of it.

Ben-Or/Tiwari 1988:

- Let $a_i = f(p^i)$ for p a primitive element,
- and let $\Lambda(z) = \prod_{i=1}^{t}(z - p^{e_i})$.
- Then $\Lambda(z)$ is the minimal generator of $(a_0, a_1, \ldots)$.

$\Rightarrow$only need $2t$ entries to find $\Lambda(z)$ (using BMA)

Sparse Polynomial Interpolation

$$f = \sum_{i=1}^{t} c_i x^{e_i}$$

Problem

Recover a t-sparse polynomial f given a black-box computing evaluations of it.

Ben-Or/Tiwari 1988:

- Let $a_i = f(p^i)$ for p a primitive element,
- and let $\Lambda(z) = \prod_{i=1}^{t}(z - p^{e_i})$.
- Then $\Lambda(z)$ is the minimal generator of $(a_0, a_1, \ldots)$.

$\Rightarrow$only need $2t$ entries to find $\Lambda(z)$ (using BMA)

$\Rightarrow$only need $2T(2E+1)$ with $e \leq E$ errors and $t \leq T$.

Ben-Or & Tiwari's Algorithm

Input: $(a_0, \ldots, a_{2t-1})$ where $a_i = f(p^i)$
Input: t, the numvber of (non-zero) terms of $f(x) = \sum_{j=1}^{t} c_j x^{e_j}$
Output: $f(x)$
begin

Run BMA on $(a_0, \ldots, a_{2t-1})$ to find $\Lambda(z)$
Find roots of $\Lambda(z)$ (polynomial factorization)
Recover e_j by repeated division (by p)
Recover c_j by solving the transposed Vandermonde system

$$\begin{bmatrix} (p^0)^{e_1} & (p^0)^{e_2} & \ldots & (p^0)^{e_t} \\ (p^1)^{e_1} & (p^1)^{e_2} & \ldots & (p^1)^{e_t} \\ \vdots & \vdots & & \vdots \\ (p^t)^{e_1} & (p^t)^{e_2} & \ldots & (p^t)^{e_t} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_t \end{bmatrix} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{t-1} \end{bmatrix}$$

Blahut's theorem

Theorem (Blahut)

The D.F.T of a vector of weight t has linear complexity at most t

$$\mathrm{DFT}_{\omega}(v) = \mathrm{Vandemonde}(\omega^0, \omega^1, \omega^2, \dots)v = Eval_{\omega^0,\omega^1,\omega^2,\dots}(v)$$

Blahut's theorem

Theorem (Blahut)

The D.F.T of a vector of weight t has linear complexity at most t

$\mathsf{DFT}_\omega(v) = \mathsf{Vandemonde}(\omega^0, \omega^1, \omega^2, \dots)v = Eval_{\omega^0,\omega^1,\omega^2,\dots}(v)$

- Univariate Ben-Or & Tiwari as a corollary

Blahut's theorem

Theorem (Blahut)

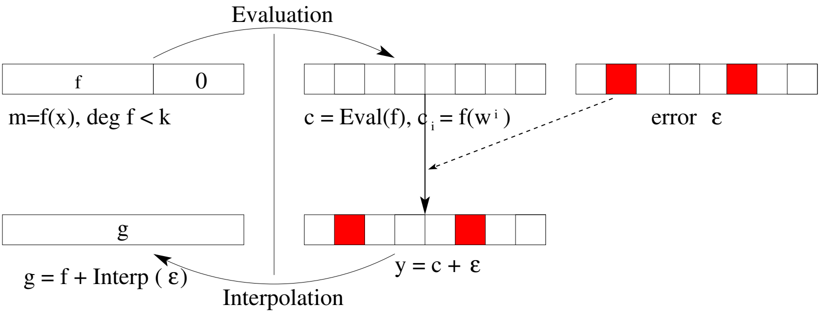
The D.F.T of a vector of weight t has linear complexity at most t

$\mathrm{DFT}_{\omega}(v) = \mathrm{Vandemonde}(\omega^0, \omega^1, \omega^2, \dots)v = Eval_{\omega^0,\omega^1,\omega^2,\dots}(v)$

- Univariate Ben-Or & Tiwari as a corollary
- Reed-Solomon codes: evaluation of a sparse error $\Rightarrow$BMA

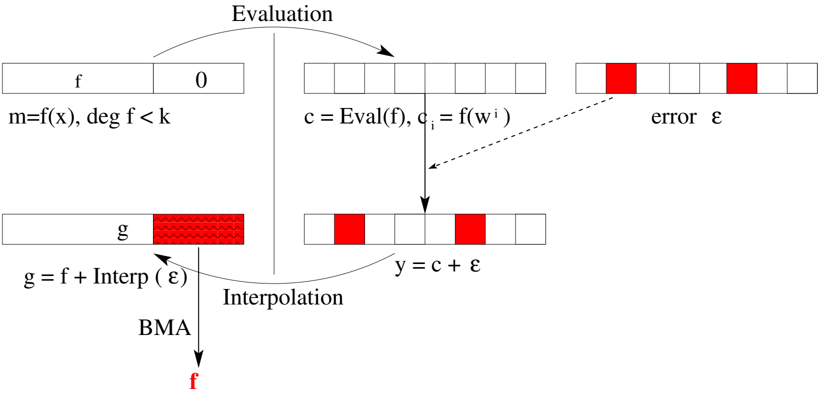
Reed-Solomon codes as Evaluation codes

$$\mathcal{C} = \{(f(\omega^1), \ldots, f(\omega^n)) \mid \deg f < k\}$$



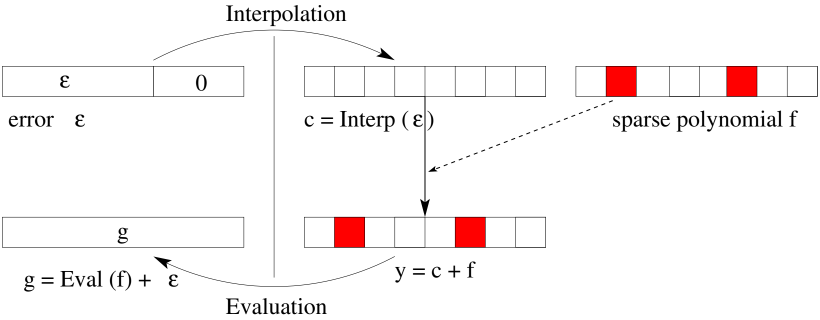
Reed-Solomon codes as Evaluation codes

$$\mathcal{C} = \{(f(\omega^1), \ldots, f(\omega^n)) \mid \deg f < k\}$$



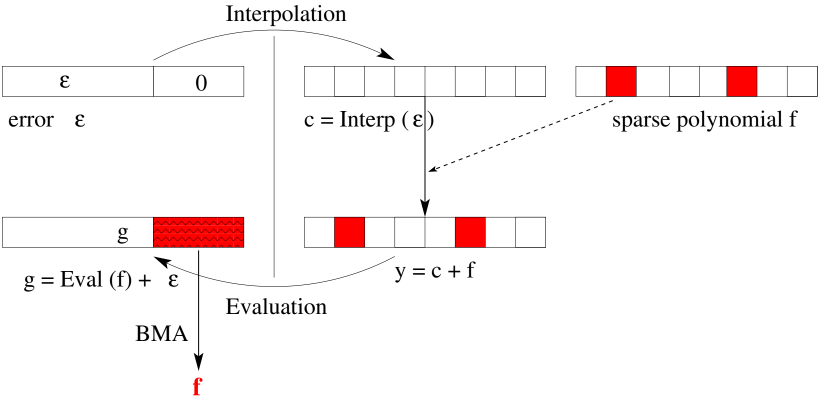
Sparse interpolation with errors

Find f from $(f(w^1), \ldots, f(w^n)) + \varepsilon$



Sparse interpolation with errors

Find f from $(f(w^1),\ldots,f(w^n)) + \varepsilon$



Same problems?

Interchanging **Evaluation** and **Interpolation**

Let $V_\omega = \text{Vandermonde}(\omega, \omega^2, \ldots, \omega^n)$. Then $(V_\omega)^{-1} = \frac{1}{n} V_{\omega^{-1}}$

Given g, find f, t-sparse and an error ε such that

$$\begin{aligned} g &= V_\omega f + \varepsilon \\ V_{\omega^{-1}} g &= nf + V_{\omega^{-1}} \epsilon \end{aligned}$$

Same problems?

Interchanging Evaluation and Interpolation

Let $V_\omega = \text{Vandermonde}(\omega, \omega^2, \ldots, \omega^n)$. Then $(V_\omega)^{-1} = \frac{1}{n} V_{\omega^{-1}}$

Given g, find f, t-sparse and an error ε such that

$$\begin{aligned} g &= V_\omega f + \varepsilon \\ V_{\omega^{-1}} g &= \underbrace{nf}_{\text{weight t error}} + \underbrace{V_{\omega^{-1}} \epsilon}_{\text{RS code word}} \end{aligned}$$

Reed-Solomon decoding: unique solution provided ε has $2t$ consecutive trailing 0's

$\Leftrightarrow$ clean segment of length $2t$

$\Leftrightarrow n \geq 2t(E+1)$

Same problems?

Interchanging **Evaluation** and **Interpolation**

Let $V_\omega = \text{Vandermonde}(\omega, \omega^2, \ldots, \omega^n)$. Then $(V_\omega)^{-1} = \frac{1}{n} V_{\omega^{-1}}$

Given g, find f, t-sparse and an error ε such that

$$\begin{aligned} g &= V_\omega f + \varepsilon \\ V_{\omega^{-1}} g &= \underbrace{nf}_{\text{weight t error}} + \underbrace{V_{\omega^{-1}} \epsilon}_{\text{RS code word}} \end{aligned}$$

Reed-Solomon decoding: unique solution provided ε has $2t$ consecutive trailing 0's
$\Leftrightarrow$ clean segment of length $2t$
$\Leftrightarrow n \geq 2t(E+1)$

BUT: location of the syndrome, is a priori unknown
$\Rightarrow$no uniqueness

Numeric Sparse Interpolation

- numerical evaluations (with noise) of a sparse polynomial
- and **outliers**

Symbolic numeric approach [Giesbrecht, Labahn &Lee'06] [Kaltofen, Lee, Yang'11]:

- Interpolation/correction using Berlekamp-Massey
- Termination (zero-discrepancy) is ill-conditioned

Numeric Sparse Interpolation

- numerical evaluations (with noise) of a sparse polynomial
- and **outliers**

Symbolic numeric approach [Giesbrecht, Labahn &Lee'06] [Kaltofen, Lee, Yang'11]:

- Interpolation/correction using Berlekamp-Massey
- Termination (zero-discrepancy) is ill-conditioned
- But the conditioning is the termination criteria

Numeric Sparse Interpolation

- numerical evaluations (with noise) of a sparse polynomial
- and **outliers**

Symbolic numeric approach [Giesbrecht, Labahn &Lee'06] [Kaltofen, Lee, Yang'11]:

- Interpolation/correction using Berlekamp-Massey
- Termination (zero-discrepancy) is ill-conditioned
- But the conditioning is the termination criteria
- Better: track two perturbed executions
  $\Rightarrow$divergence = termination

Outline

Introduction
- High performance exact computations
- Chinese remaindering
- Motivation

Sparse Interpolation with errors
- Berlekamp/Massey algorithm with errors
- Sparse Polynomial Interpolation with errors
- Relations to Reed-Solomon decoding

Dense interpolation with errors
- Decoding CRT codes: Mandelbaum algorithm
- Amplitude codes
- Adaptive decoding
- Experiments

CRT codes : Mandelbaum algorithm over $\mathbb{Z}$

Chinese Remainder Theorem

$$x \in \mathbb{Z} \longleftrightarrow \boxed{x_1 \mid x_2 \mid \quad \dots \quad \mid x_k}$$

where $m_1 \times \cdots \times m_k > x$ and $x_i = x \mod m_i \, \forall i$

CRT codes : Mandelbaum algorithm over $\mathbb{Z}$

Chinese Remainder Theorem

$$x \in \mathbb{Z} \quad \longleftrightarrow \quad \boxed{x_1}\boxed{x_2}\boxed{\quad \dots \quad}\boxed{x_k}\boxed{x_{k+1}}\boxed{\dots}\boxed{x_n}$$

where $m_1 \times \cdots \times m_n > x$ and $x_i = x \mod m_i \, \forall i$

CRT codes : Mandelbaum algorithm over $\mathbb{Z}$

Chinese Remainder Theorem

$x \in \mathbb{Z} \longleftrightarrow$

x_1	x_2	$\dots$	x_k	x_{k+1}	$\dots$	x_n

where $m_1 \times \cdots \times m_n > x$ and $x_i = x \mod m_i \; \forall i$

Definition

(n,k)-*code:* $C =$

$$\left\{ (x_1, \dots, x_n) \in \mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_n} \text{ s.t. } \exists x, \left\{ \begin{array}{ccc} x & < & m_1 \dots m_k \\ x_i & = & x \mod m_i \; \forall i \end{array} \right. \right\}$$

Principle

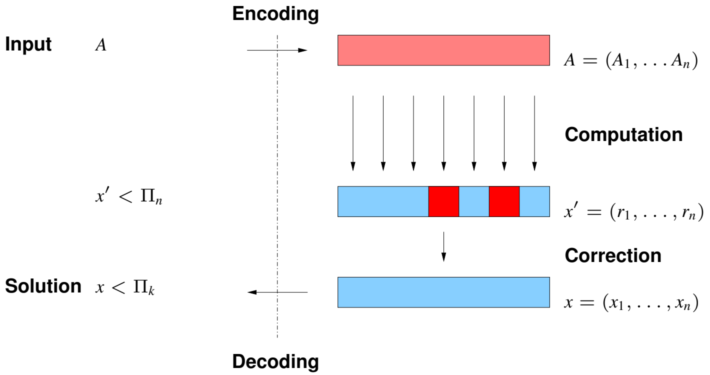
Property

$X \in C$ *iff* $X < \Pi_k$.

$$\overbrace{\underbrace{\boxed{p_1}\boxed{p_2}\boxed{\quad\dots\quad}\boxed{p_k}}_{\Pi_k = p_1 \times \cdots \times p_k}\boxed{p_{k+1}}\boxed{\dots}\boxed{p_n}}^{\Pi_n = p_1 \times \cdots \times p_n}$$

Redundancy : $r = n - k$

ABFT with Chinese remainder algorithm



Properties of the code

Error model:

- Error: $E = X' - X$
- Error support: $I = \{i \in 1 \ldots n, E \neq 0 \mod m_i\}$
- Impact of the error: $\Pi_F = \prod_{i \in I} m_i$

Properties of the code

Error model:

- Error: $E = X' - X$
- Error support: $I = \{i \in 1 \dots n, E \neq 0 \mod m_i\}$
- Impact of the error: $\Pi_F = \prod_{i \in I} m_i$

Detects up to r errors:

If $X' = X + E$ with $X \in C, \#I \leq r$,

then $X' > \Pi_k$.

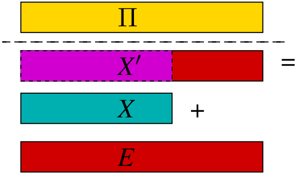
- Redundancy $r = n - k$, distance: $r + 1$
- $\Rightarrow$can correct up to $\left\lfloor \frac{r}{2} \right\rfloor$ errors in theory
- More complicated in practice...

Correction

- $\forall i \notin I : E \mod m_i = 0$
- E is a multiple of Π_V: $E = Z\Pi_V = Z\prod_{i \notin I} m_i$
- gcd$(E, \Pi) = \Pi_V$

Property

The Extended Euclidean Algorithm, applied to (Π, E) *and to* $(X' = X + E, \Pi)$*, performs the same first iterations until* $r_i < \Pi_V$*.*



$$\begin{array}{c|c}
u_0\Pi + v_0E = \Pi & u_0\Pi + v_0X' = X' \\
\vdots & \vdots \\
u_{i-1}\Pi + v_{i-1}E = \Pi_v & u_{i-1}\Pi + v_{i-1}X' = r_{i-1} \\
u_i\Pi + v_iE = 0 & u_i\Pi + v_iX' = r_i
\end{array}$$

$$\Rightarrow v_iX = r_i$$

Correction capacity

Mandelbaum 78:

- 1 symbol = 1 residue
- Polynomial time algorithm if $e \leq (n-k)\frac{\log m_{\min} - \log 2}{\log m_{\max} + \log m_{\min}}$
- worst case: exponential (random perturbation)

Goldreich Ron Sudan 99 weighted residues $\Rightarrow$equivalent

Guruswami Sahai Sudan 00 invariably polynomial time

Correction capacity

Mandelbaum 78:

- 1 symbol = 1 residue
- Polynomial time algorithm if $e \leq (n-k)\frac{\log m_{\min} - \log 2}{\log m_{\max} + \log m_{\min}}$
- worst case: exponential (random perturbation)

Goldreich Ron Sudan 99 weighted residues $\Rightarrow$equivalent

Guruswami Sahai Sudan 00 invariably polynomial time

Interpretation:

Errors have variable weights depending on their impact $\prod_{i \in I} m_i$

Example: $m_1 = 3, m_2 = 5, m_3 = 3001$

- Mandelbaum: only corrects 1 error provided $X < 3$
- Adaptive: also corrects
  - 1 error mod 3 if $X < 333$
  - 1 error mod 5 if $X < 120$
  - 2 errors mod 2 and 3 if $X < 13$

Generalized point of view: amplitude code

Over a Euclidean ring $\mathcal{A}$ with a Euclidean function ν, multiplicative and sub-additive, ie such that

$$\begin{aligned} \nu(ab) &= \nu(a)\nu(b) \\ \nu(a+b) &\leq \nu(a)+\nu(b) \end{aligned}$$

eg.

- over $\mathbb{Z}$: $\nu(x) = |x|$
- over $K[X]$: $\nu(P) = 2^{\deg(P)}$

Definition

Error impact between x and y: $\Pi_F = \prod_{i|x \neq y[m_i]} m_i$

Error amplitude: $\nu(\Pi_F)$

Amplitude codes

Distance

$$\begin{array}{rccl} \Delta : & \mathcal{A} \times \mathcal{A} & \to & \mathbb{R}_+ \\ & (x, y) & \mapsto & \displaystyle\sum_{i \mid x \neq y[m_i]} \log_2 \nu\left(m_i\right) \end{array}$$

$$\Delta(x, y) = \log_2 \nu(\Pi_F)$$

Definition ((n, k)-amplitude code)

Given $\{m_i\}_{i \leq m}$ *pairwise rel. prime, and* $\kappa \in \mathbb{R}_+$ *The set*

$$C = \{x \in \mathcal{A} : \nu(x) < \kappa\},$$

$n = \log_2 \prod_{i \leq m} m_i, k = \log_2 \kappa$. *is a* (n, k)*-amplitude code.*

Definition ((n,k)-amplitude code)

Given $\{m_i\}_{i\leq m}$ *pairwise rel. prime, and* $\kappa \in \mathbb{R}_+$ *The set*

$$C = \{x \in \mathcal{A} : \nu(x) < \kappa\},$$

$n = \log_2 \prod_{i\leq m} m_i, k = \log_2 \kappa$. *is a* (n,k)*-amplitude code.*

Property (Quasi MDS)

$\forall (x,y) \in C$

$$\Delta(x,y) > n - k - 1$$

$\Rightarrow$correction capacity = maximal amplitude of an error that can be corrected

Definition ((n,k)-amplitude code)

Given $\{m_i\}_{i\leq m}$ *pairwise rel. prime, and* $\kappa \in \mathbb{R}_+$ *The set*

$$C = \{x \in \mathcal{A} : \nu(x) < \kappa\},$$

$n = \log_2 \prod_{i\leq m} m_i, k = \log_2 \kappa$. *is a* (n,k)*-amplitude code.*

Property (Quasi MDS)

$\forall (x,y) \in C,\ \mathcal{A} = K[X]$

$$\Delta(x,y) \geq n - k + 1$$

$\sim$ *Singleton bound*

$\Rightarrow$correction capacity = maximal amplitude of an error that can be corrected

Advantages

- Generalization over any Euclidean ring
- Natural representation of the amount of information
- No need to sort moduli
- Finer correction capacities

Advantages

- Generalization over any Euclidean ring
- Natural representation of the amount of information
- No need to sort moduli
- Finer correction capacities
- **Adaptive decoding:** taking advantage of all the available redundancy
- **Early termination:** with no a priori knowledge of a bound on the result

Amplitude decoding, with static correction capacity

Amplitude based decoder over R

Input: Π, X'
Input: $\tau \in \mathbb{R}_+ \mid \tau < \frac{\nu(\Pi)}{2}$: bound on the maximal error amplitude
Output: $X \in R$: corrected message s.t. $\nu(X)4\tau^2 \leq \nu(\Pi)$
begin

$u_0 = 1, v_0 = 0, r_0 = \Pi;$
$u_1 = 0, v_1 = 1, r_1 = X';$
$i = 1;$
while $(\nu(r_i) > \nu(\Pi)/2\tau)$ **do**

Let $r_{i-1} = q_i r_i + r_{i+1}$ be the Euclidean division of r_{i-1} by r_i;
$u_{i+1} = u_{i-1} - q_i u_i;$
$v_{i+1} = v_{i-1} - q_i v_i;$
$i = i + 1;$

return $X = \frac{r_i}{v_i}$

- reaches the quasi-maximal correction capacity

Amplitude decoding, with static correction capacity

Amplitude based decoder over R

Input: Π, X'
Input: $\tau \in \mathbb{R}_+ \mid \tau < \frac{\nu(\Pi)}{2}$: bound on the maximal error amplitude
Output: $X \in R$: corrected message s.t. $\nu(X) 4\tau^2 \leq \nu(\Pi)$
begin
- $u_0 = 1, v_0 = 0, r_0 = \Pi$;
- $u_1 = 0, v_1 = 1, r_1 = X'$;
- $i = 1$;
- **while** $(\nu(r_i) > \nu(\Pi)/2\tau)$ **do**
  - Let $r_{i-1} = q_i r_i + r_{i+1}$ be the Euclidean division of r_{i-1} by r_i;
  - $u_{i+1} = u_{i-1} - q_i u_i$;
  - $v_{i+1} = v_{i-1} - q_i v_i$;
  - $i = i + 1$;
- **return** $X = \frac{r_i}{v_i}$

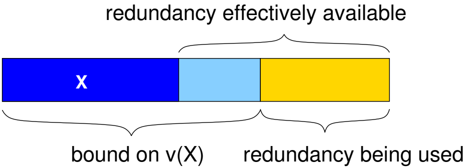
- reaches the quasi-maximal correction capacity
- requires an *a priori* knowledge of τ

  $\Rightarrow$How to make the correction capacity adaptive?

Adaptive approach

Multiple goals:

- With a fixed n, the correction capacity depends on a bound on $\nu(X)$
  $\Rightarrow$pessimistic estimate
  $\Rightarrow$how to take advantage of all the available redundancy?



A first adaptive approach: divisibility check

Termination criterion in the Extended Euclidean alg.:

- $u_{i+1}\Pi + v_{i+1}E = 0$
  $\Rightarrow E = -u_{i+1}\Pi / v_{i+1}$
  $\Rightarrow$test if v_j divides Π
- check if X satisfies: $\nu(X) \leq \frac{\nu(\Pi)}{4\nu(v_j)^2}$
- But several candidates are possible
  $\Rightarrow$discrimination by a post-condition on the result

A first adaptive approach: divisibility check

Termination criterion in the Extended Euclidean alg.:

- $u_{i+1}\Pi + v_{i+1}E = 0$
  $\Rightarrow E = -u_{i+1}\Pi / v_{i+1}$
  $\Rightarrow$test if v_j divides Π
- check if X satisfies: $\nu(X) \leq \frac{\nu(\Pi)}{4\nu(v_j)^2}$
- But several candidates are possible
  $\Rightarrow$discrimination by a post-condition on the result

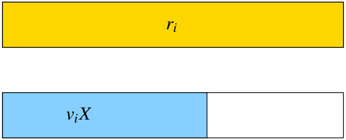
Example

m_i	3	5	7
x_i	2	3	2

- $x = 23$ *with 0 error*
- $x = 2$ *with 1 error*

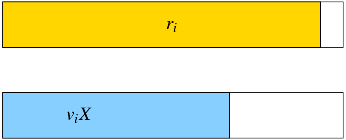
Detecting a gap

$$u_i\Pi + v_i(X+E) = r_i \quad \Rightarrow \quad u_i\Pi + v_iE = r_i - v_iX$$



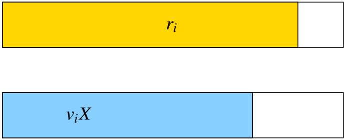
Detecting a gap

$$u_i \Pi + v_i(X + E) = r_i \quad \Rightarrow \quad u_i \Pi + v_i E = r_i - v_i X$$



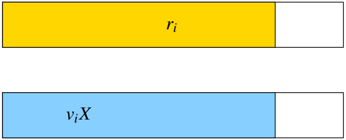
Detecting a gap

$$u_i\Pi + v_i(X+E) = r_i \qquad \Rightarrow \qquad u_i\Pi + v_iE = r_i - v_iX$$



Detecting a gap

$$u_i\Pi + v_i(X+E) = r_i \quad \Rightarrow \quad u_i\Pi + v_iE = r_i - v_iX$$

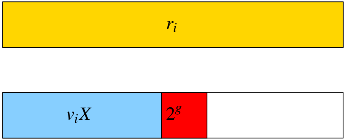


$$X = -r_i/v_i$$

- At the final iteration: $\nu(r_i) = \nu(v_iX)$
- If necessary, a gap appears between r_{i-1} and r_i.

Detecting a gap

$$u_i\Pi + v_i(X+E) = r_i \quad \Rightarrow \quad u_i\Pi + v_iE = r_i - v_iX$$

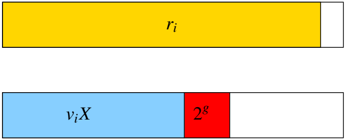


$X = -r_i/v_i$

- At the final iteration: $\nu(r_i) = \nu(v_iX)$
- If necessary, a gap appears between r_{i-1} and r_i.
- $\Rightarrow$Introduce a *blank* 2^g in order to detect a gap $> 2^g$

Detecting a gap

$$u_i\Pi + v_i(X+E) = r_i \quad \Rightarrow \quad u_i\Pi + v_iE = r_i - v_iX$$

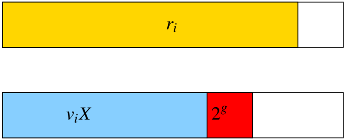


$X = -r_i/v_i$

- At the final iteration: $\nu(r_i) = \nu(v_iX)$
- If necessary, a gap appears between r_{i-1} and r_i.
- $\Rightarrow$Introduce a *blank* 2^g in order to detect a gap $> 2^g$

Detecting a gap

$$u_i\Pi + v_i(X+E) = r_i \quad \Rightarrow \quad u_i\Pi + v_iE = r_i - v_iX$$

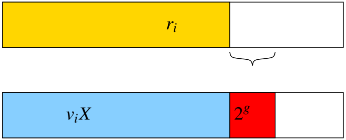


$X = -r_i/v_i$

- At the final iteration: $\nu(r_i) = \nu(v_iX)$
- If necessary, a gap appears between r_{i-1} and r_i.
- $\Rightarrow$Introduce a *blank* 2^g in order to detect a gap $> 2^g$

Detecting a gap

$u_i\Pi + v_i(X+E) = r_i \quad \Rightarrow \quad u_i\Pi + v_iE = r_i - v_iX$

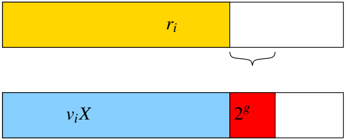


$X = -r_i/v_i$

- At the final iteration: $\nu(r_i) = \nu(v_iX)$
- If necessary, a gap appears between r_{i-1} and r_i.
- $\Rightarrow$Introduce a *blank* 2^g in order to detect a gap $> 2^g$

Detecting a gap

$$u_i\Pi + v_i(X+E) = r_i \quad \Rightarrow \quad u_i\Pi + v_iE = r_i - v_iX$$



$X = -r_i/v_i$

- At the final iteration: $\nu(r_i) = \nu(v_iX)$
- If necessary, a gap appears between r_{i-1} and r_i.
- $\Rightarrow$Introduce a *blank* 2^g in order to detect a gap $> 2^g$

Property

- *Loss of correction capacity: very small in practice*
- *Test of the divisibility for the remaining candidates*
- *Strongly reduces the number of divisibility tests*

Experiments

Size of the error	10	50	100	200	500	1000
$g = 2$	1/446	1/765	1/1118	2/1183	2/4165	1/7907
$g = 3$	1/244	1/414	1/576	2/1002	2/2164	1/4117
$g = 5$	1/53	1/97	1/153	2/262	1/575	1/1106
$g = 10$	1/1	1/3	1/9	1/14	1/26	1/35
$g = 20$	1/1	1/1	1/1	1/1	1/1	1/1

Table: Number of remaining candidates after the gap detection: c/d means d candidates with a gap $> 2^g$, and c of them passed the divisibility test. $n \approx 6001$ (3000 moduli), $\kappa \approx 201$ (100 moduli).

Experiments

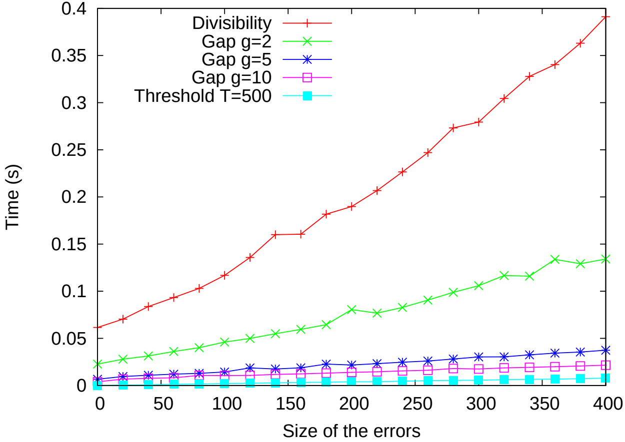


Figure: Comparison for $n \approx 26\,016$ ($m = 1300$ moduli of 20 bits), $\kappa \approx 6001$ (300 moduli) and $\tau \approx 10007$ (about 500 moduli).

Conclusion

Adaptive decoding of CRT codes

- finer bounds on the correction capacity
- adaptive decoding using the best effective redundancy
- efficient termination heuristics

Sparse interpolation code over $K[X]$

- lower bound on the necessary number of evaluations
- optimal unique decoding algorihtm
- list decoding variant

Perspectives

- Generalization to adaptive list decoding of CRT codes
- Tight bound on the size of the list when $n \geq 2t(E+1)$,
- Sparse Cauchy interpolation with errors.

Bonus : Dense rational function interpolation with errors (Cauchy interpolation)

$$y_i = \frac{f(x_i)}{g(x_i)}$$

Rational function interpolation: Pade approximant

- Find $h \in K[X]$ s.t. $h(x_i) = y_i$ (interpolation)
- Find f, g s.t. $hg = f \mod \prod (X - x_i)$ (Pade approx)

Bonus : Dense rational function interpolation with errors (Cauchy interpolation)

$$y_i = \frac{f(x_i)}{g(x_i)}$$

Rational function interpolation: Pade approximant

- Find $h \in K[X]$ s.t. $h(x_i) = y_i$ (interpolation)
- Find f, g s.t. $hg = f \mod \prod(X - x_i)$ (Pade approx)

Introducing an error of impact $\Pi_F = \prod_{i \in I}(X - x_i)$:

$$hg\Pi_F = f\Pi_F \mod \prod(X - x_i)$$

Property

If $n \geq \deg f + \deg g + 2e$, one can interpolate with at most e errors