Adaptive decoding for evaluation/interpolation codes

Clément PERNET
joint work with
M. Khonji, J-L. Roch, T. Roche et T. Stalinski
INRIA-MOAIS, LIG, Grenoble Université

Claude Shannon Institute, University College Dublin
Thursday June 3, 2010

Plan

Introduction

- High performance exact computations
- Security of distributed computations
- Fault tolerance
- Exact linear algebra
- Chinese remaindering

Redundant residues codes

- Over Z: Mandelbaum algorithm
- Over K[X]: Reed Solomon point of view
- Generalization

Adaptive approach

- A first approach
- Detecting a gap
- Experiments
- Terminaison anticipée

Plan

Introduction
- High performance exact computations
- Security of distributed computations
- Fault tolerance
- Exact linear algebra
- Chinese remaindering

Redundant residues codes
- Over Z: Mandelbaum algorithm
- Over K[X]: Reed Solomon point of view
- Generalization

Adaptive approach
- A first approach
- Detecting a gap
- Experiments
- Terminaison anticipée

High performance exact computations

Domain of Computation

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

High performance exact computations

Domain of Computation

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

Application domains:

Computational number theory:

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

Crypto:

- Algebraic attacks (Quadratic sieves, Groebner bases, index calculus,...)
- Recherches de grands nombres premiers

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

Plan

Introduction
- High performance exact computations
- Security of distributed computations
- Fault tolerance
- Exact linear algebra
- Chinese remaindering

Redundant residues codes
- Over Z: Mandelbaum algorithm
- Over K[X]: Reed Solomon point of view
- Generalization

Adaptive approach
- A first approach
- Detecting a gap
- Experiments
- Terminaison anticipée

Parallel computation

Trend towards massive parallelism

Personal computers $\Rightarrow$ multi/many cores

- End of the CPU frequency race
- Multi-core: 2, 4, 6, ...
- Many-cores: near future, already in GPUs
- multi-GPU

Parallel computation

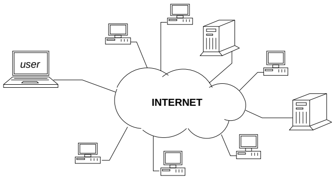
Trend towards massive parallelism

Personal computers $\Rightarrow$ multi/many cores

- End of the CPU frequency race
- Multi-core: 2, 4, 6, ...
- Many-cores: near future, already in GPUs
- multi-GPU

HPC $\Rightarrow$ Distributed computation, global computing

- servers
- clusters, grids
- volunteer computing, P2P
- ambiant, cloud computing



Security of computations

Peer to peer

- Popular projects: Mersenne Prime search, SETI@Home, Folding@Home, BOINC
- Petaflops (670 000 PS3s) in 2007

Security of computations

Peer to peer

- Popular projects: Mersenne Prime search, SETI@Home, Folding@Home, BOINC
- Petaflops (670 000 PS3s) in 2007
- What level of confidence?
  - machine badly configured, overclocking
  - malicious program
  - large scale corruption possible (client patched and redistributed)

Security of computations

Peer to peer

- Popular projects: Mersenne Prime search, SETI@Home, Folding@Home, BOINC
- Petaflops (670 000 PS3s) in 2007
- What level of confidence?
  - machine badly configured, overclocking
  - malicious program
  - large scale corruption possible (client patched and redistributed)

Ambiant distributed computation

- Connection / Disconnection of resources
- Corruption (forged tasks)

Plan

Introduction
- High performance exact computations
- Security of distributed computations
- Fault tolerance
- Exact linear algebra
- Chinese remaindering

Redundant residues codes
- Over Z: Mandelbaum algorithm
- Over K[X]: Reed Solomon point of view
- Generalization

Adaptive approach
- A first approach
- Detecting a gap
- Experiments
- Terminaison anticipée

Fault tolerance

Several kinds of faults

- Fail stop (crash, disconnection...)
- Network congestion
- Malicious attacks

Fault tolerance

Several kinds of faults

- Fail stop (crash, disconnection...)
- Network congestion
- Malicious attacks

⇒Byzantine fault model (not always wrong)

- Most of the time correct ⇒blacklisting is not an option

Fault tolerance

Several kinds of faults

- Fail stop (crash, disconnection...)
- Network congestion
- Malicious attacks

$\Rightarrow$Byzantine fault model (not always wrong)

- Most of the time correct $\Rightarrow$blacklisting is not an option

Several approaches:

- **Replication**: vote, spot-checking, blacklisting, partial execution on safe resources
- **Verification using post-conditions** on the output
- **ABFT**: Algorithm-based fault tolerance

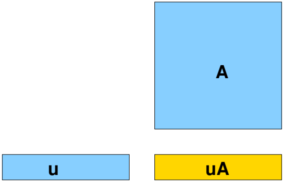
ABFT: Algorithmic Based Fault Tolerance

Idea: incorporate redundancy in the algorithm
$\Rightarrow$use properties specific to the problem

ABFT: Algorithmic Based Fault Tolerance

Idea: incorporate redundancy in the algorithm
$\Rightarrow$use properties specific to the problem

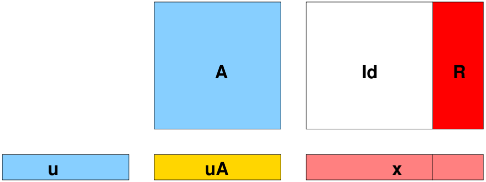
Example: Matrix-vector product [Dongarra & Al. 2006]



ABFT: Algorithmic Based Fault Tolerance

Idea: incorporate redundancy in the algorithm
$\Rightarrow$use properties specific to the problem

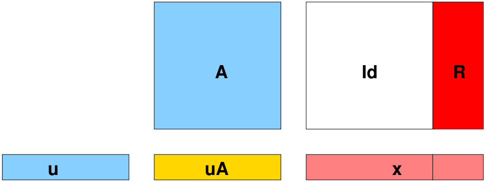
Example: Matrix-vector product [Dongarra & Al. 2006]



ABFT: Algorithmic Based Fault Tolerance

Idea: incorporate redundancy in the algorithm
$\Rightarrow$use properties specific to the problem

Example: Matrix-vector product [Dongarra & Al. 2006]



- pre-compute the product $B = A \times \begin{bmatrix} I & R \end{bmatrix}$
- compute $x = uB$ in parallel
- decode/correct x

Plan

Introduction
- High performance exact computations
- Security of distributed computations
- Fault tolerance
- Exact linear algebra
- Chinese remaindering

Redundant residues codes
- Over Z: Mandelbaum algorithm
- Over K[X]: Reed Solomon point of view
- Generalization

Adaptive approach
- A first approach
- Detecting a gap
- Experiments
- Terminaison anticipée

Exact linear algebra

> *Mathematics is the art of reducing any problem to linear algebra*
>
> *W. Stein*

$\Rightarrow$One of the building blocks to be optimized in numerous applications

Exact linear algebra

> *Mathematics is the art of reducing any problem to linear algebra*
>
> *W. Stein*

$\Rightarrow$One of the building blocks to be optimized in numerous applications

Optimizing exact linear algebra:

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

Exact linear algebra

> *Mathematics is the art of reducing any problem to linear algebra*
>
> *W. Stein*

$\Rightarrow$One of the building blocks to be optimized in numerous applications

Optimizing exact linear algebra:

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

Plan

Introduction
- High performance exact computations
- Security of distributed computations
- Fault tolerance
- Exact linear algebra
- Chinese remaindering

Redundant residues codes
- Over Z: Mandelbaum algorithm
- Over K[X]: Reed Solomon point of view
- Generalization

Adaptive approach
- A first approach
- Detecting a gap
- Experiments
- Terminaison anticipée

Chinese remainder algorithm

$$\mathbb{Z}/(n_1 \dots n_k)\mathbb{Z} \equiv \mathbb{Z}/n_1\mathbb{Z} \times \cdots \times \mathbb{Z}/n_k\mathbb{Z}$$

Computation of $y = f(x)$ over $\mathbb{Z}$

begin
- Compute a bound β on $\max(|f|)$;
- Pick $n_1, \dots n_k$, pairwise prime, s.t. $n_1 \dots n_k > \beta$;
- **for** $i = 1 \dots k$ **do**
  - Compute $y_i = f(x \mod n_i) \mod n_i$
- Compute $y = \mathrm{CRT}(y_1, \dots, y_k)$

end

$$\begin{array}{rrcl} \mathrm{CRT}: & \mathbb{Z}/n_1\mathbb{Z} \times \cdots \times \mathbb{Z}/n_k\mathbb{Z} & \to & \mathbb{Z}/(n_1 \dots n_k)\mathbb{Z} \\ & (x_1, \dots, x_k) & \mapsto & \sum_{i=1}^{k} x_i \Pi_i Y_i \mod \Pi \end{array}$$

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

Chinese remainder algorithm

$$\mathbb{Z}/(n_1 \dots n_k)\mathbb{Z} \equiv \mathbb{Z}/n_1\mathbb{Z} \times \dots \times \mathbb{Z}/n_k\mathbb{Z}$$

Computation of $y = f(x)$ over $\mathbb{Z}$

begin
- Compute a bound β on $\max(|f|)$;
- Pick $n_1, \dots n_k$, pairwise prime, s.t. $n_1 \dots n_k > \beta$;
- **for** $i = 1 \dots k$ **do**
  - Compute $y_i = f(x \mod n_i) \mod n_i$; `/* Evaluation */`
- Compute $y = \mathrm{CRT}(y_1, \dots, y_k)$; `/* Interpolation */`

end

$$\begin{array}{rrcl} \mathrm{CRT}: & \mathbb{Z}/n_1\mathbb{Z} \times \dots \times \mathbb{Z}/n_k\mathbb{Z} & \rightarrow & \mathbb{Z}/(n_1 \dots n_k)\mathbb{Z} \\ & (x_1, \dots, x_k) & \mapsto & \sum_{i=1}^k x_i \Pi_i Y_i \mod \Pi \end{array}$$

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

Chinese remaindering and evaluation/interpolation

Evaluate P in a $\quad\leftrightarrow\quad$ 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: $P = \sum_{i=1}^{k} \frac{\prod_{j \neq i}(X - a_j)}{\prod_{j \neq i}(a_i - a_j)}$	

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: $P = \sum_{i=1}^{k} \frac{\prod_{j \neq i}(X - a_j)}{\prod_{j \neq i}(a_i - a_j)}$	$N = \sum_{i=1}^{k} a_i \prod_{j \neq i} m_j (\prod_{j \neq i} m_j)^{-1 [m_i]}$

Chinese remaindering and evaluation/interpolation

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

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

Analogy: complexities over $\mathbb{Z} \leftrightarrow$ over $K[X]$

- size of coefficients
- $\mathcal{O}\left(\log \|\text{result}\| \times T_{\text{algebr.}}\right)$

- degree of polynomials
- $\mathcal{O}\left(\deg(\text{result}) \times T_{\text{algebr.}}\right)$

Chinese remaindering and evaluation/interpolation

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

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

Analogy: complexities over $\mathbb{Z} \leftrightarrow$ over $K[X]$

- size of coefficients
- $\mathcal{O}\left(\log \|\text{result}\| \times T_{\text{algebr.}}\right)$
- $\det(n, \|A\|) = \tilde{\mathcal{O}}\left(n \log |A\| \times n^{\omega}\right)$

- degree of polynomials
- $\mathcal{O}\left(\deg(\text{result}) \times T_{\text{algebr.}}\right)$
- $\det(n, d) = \tilde{\mathcal{O}}\left(nd \times n^{\omega}\right)$

Early termination

Classic Chinese remaindering

- bound β on the result
- Choice of the n_i: such that $n_1 \dots n_k > \beta$

$\Rightarrow$deterministic algorithm

Early termination

Classic Chinese remaindering

- bound β on the result
- Choice of the n_i: such that $n_1 \dots n_k > \beta$

$\Rightarrow$deterministic algorithm

Early termination

- For each new modulo n_i:
  - reconstruct $y_i = f(x) \mod n_1 \times \cdots \times n_i$
  - If $y_i = y_{i-1}$ $\Rightarrow$terminated

$\Rightarrow$probabilistic Monte Carlo

Early termination

Classic Chinese remaindering

- bound β on the result
- Choice of the n_i: such that $n_1 \ldots n_k > \beta$

$\Rightarrow$deterministic algorithm

Early termination

- For each new modulo n_i:
  - reconstruct $y_i = f(x) \mod n_1 \times \cdots \times n_i$
  - If $y_i = y_{i-1}$ $\Rightarrow$terminated

$\Rightarrow$probabilistic Monte Carlo

Advantage:

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

Plan

Introduction
- High performance exact computations
- Security of distributed computations
- Fault tolerance
- Exact linear algebra
- Chinese remaindering

Redundant residues codes
- Over Z: Mandelbaum algorithm
- Over K[X]: Reed Solomon point of view
- Generalization

Adaptive approach
- A first approach
- Detecting a gap
- Experiments
- Terminaison anticipée

Redundant residues codes

Principle:

- Chinese remaindering based parallelization
- Byzantines faults affecting some modular computations
- Fault tolerant reconstruction
  $\Rightarrow$Algorithm Based Fault Tolerance (ABFT)

Plan

Introduction
- High performance exact computations
- Security of distributed computations
- Fault tolerance
- Exact linear algebra
- Chinese remaindering

Redundant residues codes
- Over Z: Mandelbaum algorithm
- Over K[X]: Reed Solomon point of view
- Generalization

Adaptive approach
- A first approach
- Detecting a gap
- Experiments
- Terminaison anticipée

Mandelbaum algorithm over $\mathbb{Z}$

Chinese Remainder Theorem

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

where $p_1 \times \cdots \times p_k > x$ and $x_i = x \mod p_i \, \forall i$

Mandelbaum algorithm over $\mathbb{Z}$

Chinese Remainder Theorem

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

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

where $p_1 \times \cdots \times p_n > x$ and $x_i = x \mod p_i \; \forall i$

Mandelbaum algorithm over $\mathbb{Z}$

Chinese Remainder Theorem

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

where $p_1 \times \cdots \times p_n > x$ and $x_i = x \mod p_i \, \forall i$

Definition

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

$$\left\{(x_1, \dots, x_n) \in \mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_n} \text{ s.t. } \exists x, \left\{ \begin{array}{ccc} x & < & p_1 \dots p_k \\ x_i & = & x \mod p_i \, \forall i \end{array} \right. \right\}$$

Principle

Property

$$X \in C \text{ iff } X < \Pi_k .$$

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

Redundancy : $r = n - k$

Principle

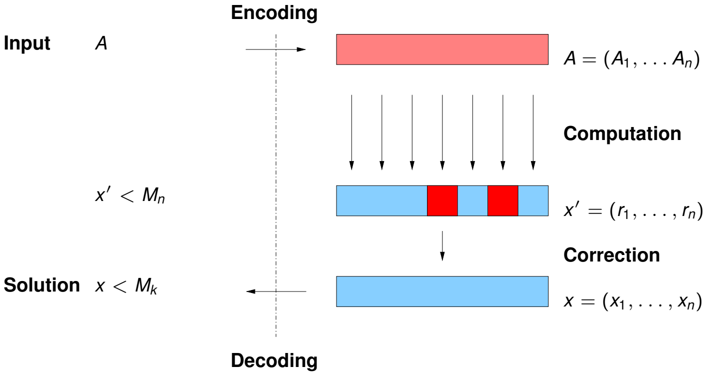
Transmission Channel $\equiv$ Computation

Principle

Noisy Transmission Channel $\equiv$ Unsecured Computation

Principle

Noisy Transmission Channel $\equiv$ Unsecured Computation



Properties of the code

Error model:

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

Properties of the code

Error model:

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

Detects up to r errors:

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

$$X' > \Pi_k.$$

- 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

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

Mandelbaum 78: rational reconstruction

$$X = X' - E = X' - Z\Pi_v$$
$$\frac{X}{\Pi} = \frac{X'}{\Pi} - \frac{Z}{\Pi_F}$$

$\Rightarrow \left|\frac{X'}{\Pi} - \frac{Z}{\Pi_F}\right| \leq \frac{1}{2\Pi_F^2}$
$\Rightarrow \frac{Z}{\Pi_F} = \frac{E}{\Pi}$ is a convergent of $\frac{X'}{\Pi}$
$\Rightarrow$rational reconstruction of $X' \mod \Pi$

Correction capacity

Mandelbaum 78:

- 1 symbol = 1 residue
- Polynomial algorithm if $e \leq (n-k)\frac{\log p_{\min} - \log 2}{\log p_{\max} + \log p_{\min}}$
- 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 algorithm if $e \leq (n-k)\frac{\log p_{\min} - \log 2}{\log p_{\max} + \log p_{\min}}$
- 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 $\Pi_F = \prod_{i \in I} p_i$
- Example: $X = 20, p_1 = 2, p_2 = 3, p_3 = 101$
  - 1 error on $X \mod 2$, or $X \mod 3$, can be corrected
  - but not on $X \mod 101$

Plan

Introduction
- High performance exact computations
- Security of distributed computations
- Fault tolerance
- Exact linear algebra
- Chinese remaindering

Redundant residues codes
- Over Z: Mandelbaum algorithm
- Over K[X]: Reed Solomon point of view
- Generalization

Adaptive approach
- A first approach
- Detecting a gap
- Experiments
- Terminaison anticipée

Analogy with Reed Solomon

Gao02 Reed-Solomon decoding by extended Euclidean Alg:

- Chinese Remaindering over $K[X]$
- $p_i = X - a_i$
- Encoding = evaluation in a_i
- Decoding = interpolation
- Correction = Extended Euclidean algorithm étendu

Analogy with Reed Solomon

Gao02 Reed-Solomon decoding by extended Euclidean Alg:

- Chinese Remaindering over $K[X]$
- $p_i = X - a_i$
- Encoding = evaluation in a_i
- Decoding = interpolation
- Correction = Extended Euclidean algorithm étendu

$\Rightarrow$Generalization for p_i of degrees > 1
$\Rightarrow$Variable impact, depending on the degree of p_i
$\Rightarrow$Necessary unification [Sudan 01,...]

Plan

Introduction
- High performance exact computations
- Security of distributed computations
- Fault tolerance
- Exact linear algebra
- Chinese remaindering

Redundant residues codes
- Over Z: Mandelbaum algorithm
- Over K[X]: Reed Solomon point of view
- Generalization

Adaptive approach
- A first approach
- Detecting a gap
- Experiments
- Terminaison anticipée

Generalized point of view: amplitude code

- Over a Euclidean ring $\mathcal{A}$ with a Euclidean function ν
- Distance

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

Definition

(n,k) amplitude code $C = \{x \in \mathcal{A} : \nu(x) < \kappa\}$,
$n = \log_2 \Pi, k = \log_2 \kappa$.

Generalized point of view: amplitude code

- Over a Euclidean ring $\mathcal{A}$ with a Euclidean function ν
- Distance

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

Definition

(n,k) amplitude code $C = \{x \in \mathcal{A} : \nu(x) < \kappa\}$*,*
$n = \log_2 \Pi, k = \log_2 \kappa$*.*

Property (Quasi MDS)

$d > n - k$ *in general, and* $d \geq n - k + 1$ *over* $K[X]$*.*

$\Rightarrow$correction rate = 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

Interpretation of Mandelbaum's algorithm

Remark

Rational reconstruction $\Rightarrow$ *Partial Extended Euclidean Algorithm*

Property

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

$$\begin{array}{c|c} u_{i-1}E + v_{i-1}\Pi = \Pi_v & u_{i-1}X' + v_{i-1}\Pi = r_{i-1} \\ u_iE + v_i\Pi = 0 & u_iX' + v_i\Pi = r_i \end{array}$$

$$\Rightarrow u_iX = r_i$$

Amplitude decoding, with static correction capacity

Amplitude based decoder over R

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

- $\alpha_0 = 1, \beta_0 = 0, r_0 = \Pi$;
- $\alpha_1 = 0, \beta_1 = 1, r_1 = X'$;
- $i = 1$;
- **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;
  - $\alpha_{i+1} = \alpha_{i-1} - q_i \alpha_i$;
  - $\beta_{i+1} = \beta_{i-1} - q_i \beta_i$;
  - $i = i + 1$;
- **return** $X = -\frac{r_i}{\beta_i}$

end

- reaches the quasi-maximal correction capacity

Amplitude decoding, with static correction capacity

Amplitude based decoder over R

Donnée: Π, X'

Donnée: $\tau \in \mathbb{R}_+ \mid \tau < \frac{\nu(\Pi)}{2}$: bound on the maximal error amplitude

Résultat: $X \in R$: corrected message s.t. $\nu(X)4\tau^2 \leq \nu(\Pi)$

begin

$\quad \alpha_0 = 1, \beta_0 = 0, r_0 = \Pi;$

$\quad \alpha_1 = 0, \beta_1 = 1, r_1 = X';$

$\quad i = 1;$

$\quad$ **while** $(\nu(r_i) > \nu(\Pi)/2\tau)$ **do**

$\quad\quad$ Let $r_{i-1} = q_i r_i + r_{i+1}$ be the Euclidean division of r_{i-1} by r_i;

$\quad\quad \alpha_{i+1} = \alpha_{i-1} - q_i \alpha_i;$

$\quad\quad \beta_{i+1} = \beta_{i-1} - q_i \beta_i;$

$\quad\quad i = i + 1;$

$\quad$ **return** $X = -\frac{r_i}{\beta_i}$

end

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

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

Plan

Introduction
- High performance exact computations
- Security of distributed computations
- Fault tolerance
- Exact linear algebra
- Chinese remaindering

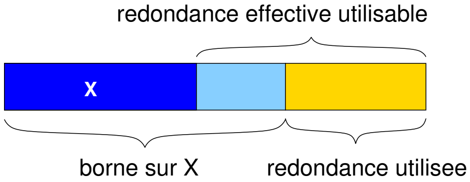
Redundant residues codes
- Over Z: Mandelbaum algorithm
- Over K[X]: Reed Solomon point of view
- Generalization

Adaptive approach
- A first approach
- Detecting a gap
- Experiments
- Terminaison anticipée

Adaptive approach

Multiple goals:

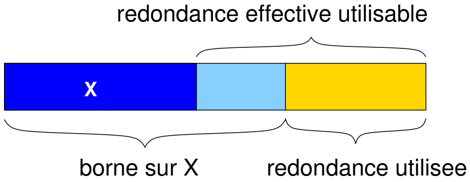
- With a fixed n, the correction capacity depends on a bound on X
  - $\Rightarrow$pessimistic estimate
  - $\Rightarrow$how to take advantage of all the available redundancy?



Adaptive approach

Multiple goals:

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



- Allow early termination: variable n and unknown bound

Plan

Introduction
High performance exact computations
Security of distributed computations
Fault tolerance
Exact linear algebra
Chinese remaindering

Redundant residues codes
Over Z: Mandelbaum algorithm
Over K[X]: Reed Solomon point of view
Generalization

Adaptive approach
A first approach
Detecting a gap
Experiments
Terminaison anticipée

A first adaptive approach

Termination criterion in the Extended Euclidean alg.:

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

A first adaptive approach

Termination criterion in the Extended Euclidean alg.:

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

Example

p_i	3	5	7
x_i	2	3	2

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

Plan

Introduction

High performance exact computations

Security of distributed computations

Fault tolerance

Exact linear algebra

Chinese remaindering

Redundant residues codes

Over Z: Mandelbaum algorithm

Over K[X]: Reed Solomon point of view

Generalization

Adaptive approach

A first approach

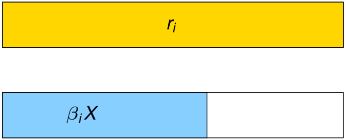
Detecting a gap

Experiments

Terminaison anticipée

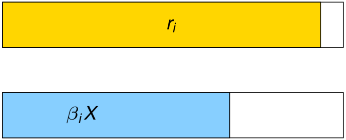
Detecting a gap

$$\alpha_i \Pi - \beta_i (X + E) = r_i \qquad \Rightarrow \qquad \alpha_i \Pi - \beta_i E = r_i + \beta_i X$$



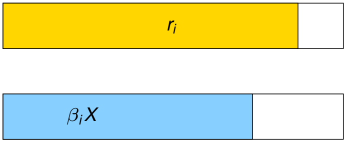
Detecting a gap

$$\alpha_i \Pi - \beta_i (X + E) = r_i \quad \Rightarrow \quad \alpha_i \Pi - \beta_i E = r_i + \beta_i X$$



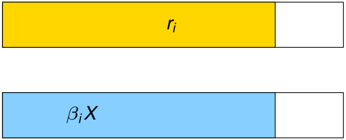
Detecting a gap

$$\alpha_i \Pi - \beta_i(X + E) = r_i \qquad \Rightarrow \qquad \alpha_i \Pi - \beta_i E = r_i + \beta_i X$$



Detecting a gap

$$\alpha_i \Pi - \beta_i(X + E) = r_i \qquad \Rightarrow \qquad \alpha_i \Pi - \beta_i E = r_i + \beta_i X$$

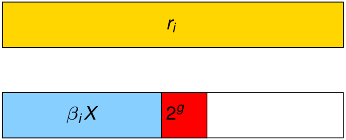


$$X = -r_i/\beta_i$$

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

Detecting a gap

$$\alpha_i \Pi - \beta_i(X + E) = r_i \qquad \Rightarrow \qquad \alpha_i \Pi - \beta_i E = r_i + \beta_i X$$

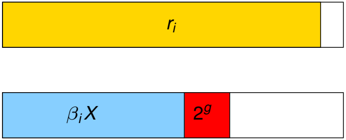


$X = -r_i/\beta_i$

- At the final iteration: $\nu(r_i) \approx \nu(\beta_i X)$
- 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

$$\alpha_i \Pi - \beta_i (X + E) = r_i \quad \Rightarrow \quad \alpha_i \Pi - \beta_i E = r_i + \beta_i X$$

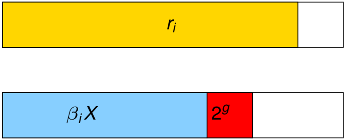


$X = -r_i / \beta_i$

- At the final iteration: $\nu(r_i) \approx \nu(\beta_i X)$
- 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

$$\alpha_i \Pi - \beta_i (X + E) = r_i \quad \Rightarrow \quad \alpha_i \Pi - \beta_i E = r_i + \beta_i X$$

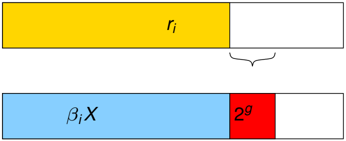


$$X = -r_i / \beta_i$$

- At the final iteration: $\nu(r_i) \approx \nu(\beta_i X)$
- 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

$$\alpha_i \Pi - \beta_i(X + E) = r_i \qquad \Rightarrow \qquad \alpha_i \Pi - \beta_i E = r_i + \beta_i X$$

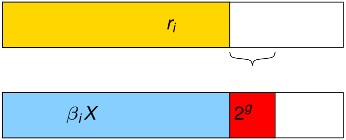


$$X = -r_i/\beta_i$$

- At the final iteration: $\nu(r_i) \approx \nu(\beta_i X)$
- 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

$$\alpha_i \Pi - \beta_i (X + E) = r_i \qquad \Rightarrow \qquad \alpha_i \Pi - \beta_i E = r_i + \beta_i X$$



$$X = -r_i / \beta_i$$

- At the final iteration: $\nu(r_i) \approx \nu(\beta_i X)$
- 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*

Plan

Introduction
High performance exact computations
Security of distributed computations
Fault tolerance
Exact linear algebra
Chinese remaindering

Redundant residues codes
Over Z: Mandelbaum algorithm
Over K[X]: Reed Solomon point of view
Generalization

Adaptive approach
A first approach
Detecting a gap
Experiments
Terminaison anticipée

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

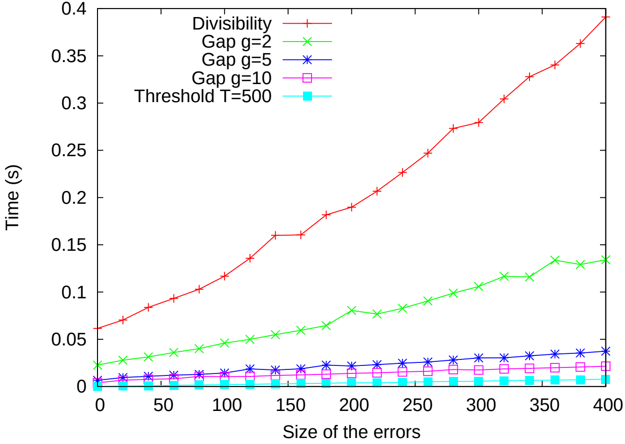


Figure: Comparison for $n \approx 26\,016$ ($m = 1300$ moduli of 20 bits), $\kappa \approx 6001$ (300 moduli) and $\tau \approx 10007$ (about 500 moduli).

Experiments

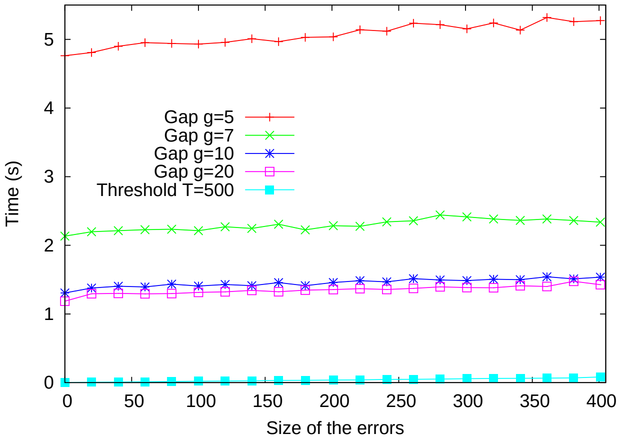


Figure: Comparison for $n \approx 200\,917$ ($m = 10000$ moduli of 20 bits), $\kappa \approx 17\,0667$ (8500 moduli) and $\tau \approx 10498$ (500 moduli).

Gap: Euclidean Algorithm down to the end $\Rightarrow$overhead

Plan

Introduction
- High performance exact computations
- Security of distributed computations
- Fault tolerance
- Exact linear algebra
- Chinese remaindering

Redundant residues codes
- Over Z: Mandelbaum algorithm
- Over K[X]: Reed Solomon point of view
- Generalization

Adaptive approach
- A first approach
- Detecting a gap
- Experiments
- Terminaison anticipée

Early termination

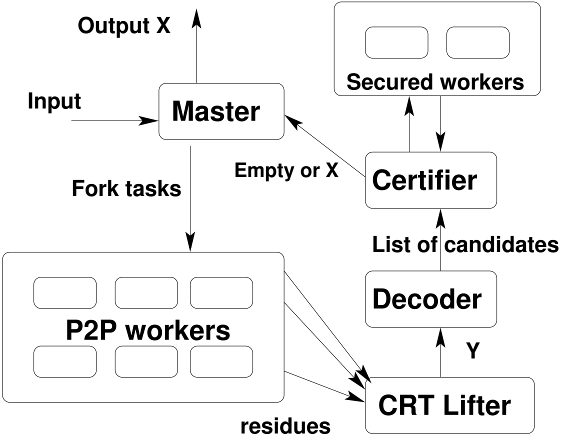


Figure: Fault tolerant distributed computation with early termination

Conclusion

New metric for redundant residue codes:

- Unification
- Finer bounds on the correction capacities
- Enable adaptive decoding

Adaptative decoding and early termination

- Several approaches
- Gap method: limited overhead, better performances
- Insertion in a global framework for early termination

Perspective

- Theoretical explanation of the efficiency of the *gap* method: average distribution of the quotients q_i in the EEA).
- Better correction capacities over $\mathbb{Z}$ and $K[X]$ only.
- Generalization to adaptive list decoding [Sudan, Guruswami]