Fast interpolation and multiplication of unbalanced polynomials

Pascal Giorgi, Bruno Grenet, Armelle Perret du Cray, Daniel S. Roche

ISSAC'24, Raleigh, NC, USA
July 16–19, 2024

Motivations

Polynomial multiplication complexity over $\mathcal{R}[x]$ is well understood

Algebraic complexity

- $O(d^{\log_2 3}), \ldots, O(d^{1+o(1)})$ [Karatsuba 1962, Toom 1963, Cook 1966]
- $O(d \log d \log\log d)$ [Schönhage-Strassen 1971, Cantor-Kaltofen 1991]
- $O(d \log d)$ when $\mathcal{R} = \mathsf{F}_q$ [Harvey-van der Hoeven 2022]

Motivations

Polynomial multiplication complexity over $\mathcal{R}[x]$ is well understood

Algebraic complexity

- $O(d^{\log_2 3}), \ldots, O(d^{1+o(1)})$ [Karatsuba 1962, Toom 1963, Cook 1966]
- $O(d \log d \log\log d)$ [Schönhage-Strassen 1971, Cantor-Kaltofen 1991]
- $O(d \log d)$ when $\mathcal{R} = \mathsf{F}_q$ [Harvey-van der Hoeven 2022]

But, bit complexity for $\mathbb{Z}[x]$: $\tilde{O}(d \log H)$ [Schönhage 1982]

$\hookrightarrow$ $f, g \in \mathbb{Z}[x]_{<d}$ with height at most H

Motivations

Polynomial multiplication complexity over $\mathcal{R}[x]$ is well understood

Algebraic complexity

- $O(d^{\log_2 3}), \ldots, O(d^{1+o(1)})$ [Karatsuba 1962, Toom 1963, Cook 1966]
- $O(d \log d \log\log d)$ [Schönhage-Strassen 1971, Cantor-Kaltofen 1991]
- $O(d \log d)$ when $\mathcal{R} = \mathsf{F}_q$ [Harvey-van der Hoeven 2022]

But, bit complexity for $\mathbb{Z}[x]$: $\tilde{O}(d \log H)$ [Schönhage 1982]

$\hookrightarrow$ $f, g \in \mathbb{Z}[x]_{<d}$ with height at most H

☺ quasi-linear if balanced coefficients

Motivations

Polynomial multiplication complexity over $\mathcal{R}[x]$ is well understood

Algebraic complexity

- $O(d^{\log_2 3}), \ldots, O(d^{1+o(1)})$ [Karatsuba 1962, Toom 1963, Cook 1966]
- $O(d \log d \log\log d)$ [Schönhage-Strassen 1971, Cantor-Kaltofen 1991]
- $O(d \log d)$ when $\mathcal{R} = \mathsf{F}_q$ [Harvey-van der Hoeven 2022]

But, bit complexity for $\mathbb{Z}[x]$: $\tilde{O}(d \log H)$ [Schönhage 1982]

$\hookrightarrow$ $f, g \in \mathbb{Z}[x]_{<d}$ with height at most H

☺ quasi-linear if balanced coefficients

☹ not quasi-linear with unbalanced coefficients, e.g. $||f||_1 = ||g||_1 = ||fg||_1 = O(H)$

Motivations

Polynomial multiplication complexity over $\mathcal{R}[x]$ is well understood

Algebraic complexity

- $O(d^{\log_2 3}), \ldots, O(d^{1+o(1)})$ [Karatsuba 1962, Toom 1963, Cook 1966]
- $O(d \log d \log\log d)$ [Schönhage-Strassen 1971, Cantor-Kaltofen 1991]
- $O(d \log d)$ when $\mathcal{R} = \mathsf{F}_q$ [Harvey-van der Hoeven 2022]

But, bit complexity for $\mathbb{Z}[x]$: $\tilde{O}(d \log H)$ [Schönhage 1982]

$\hookrightarrow$ $f, g \in \mathbb{Z}[x]_{<d}$ with height at most H

☺ quasi-linear if balanced coefficients

☹ not quasi-linear with unbalanced coefficients, e.g. $||f||_1 = ||g||_1 = ||fg||_1 = O(H)$

$\Rightarrow$ only few works, e.g. Toom-Cook with structured unbalancedness [Bodrato-Zanoni 2020]

Unbalanced polynomials

Example

$$\begin{aligned} & x^7 + 3x^6 - 1000000000x^5 - 3x^4 - 4x^3 - x^2 + x - 3 \\ \times \quad & x^7 + 3x^6 + 1000000006x^5 - 3x^4 - 4x^3 - 7x^2 + x - 3 \end{aligned}$$

$$= x^{14} + 6x^{13} + 15x^{12} + 12x^{11} - 10000000060000000026x^{10} - 50x^9 - 37x^8 + 6000000018x^7 + 28x^6 + 8x^5 + 17x^4 + 16x^3 + 25x^2 - 6x + 9$$

Unbalanced polynomials

Example

$$\begin{aligned} & x^7 + 3x^6 - 1000000000x^5 - 3x^4 - 4x^3 - x^2 + x - 3 \\ \times \quad & x^7 + 3x^6 + 1000000006x^5 - 3x^4 - 4x^3 - 7x^2 + x - 3 \\ \hline = \; & x^{14} + 6x^{13} + 15x^{12} + 12x^{11} - 10000000060000000026x^{10} - 50x^9 - 37x^8 + 6000000018x^7 + 28x^6 + 8x^5 + 17x^4 + 16x^3 + 25x^2 - 6x + 9 \end{aligned}$$

Let $f = \sum_{i=0}^{d} f_i x^i \Longrightarrow \begin{cases} s = \text{bitlen}(f) \approx \sum \log |f_i| > d \\ H = \max |f_i| \leq 2^s \end{cases}$

- Balanced case: $\log H = \Theta(s/d) \hookrightarrow$ bit complexity $O^{\sim}(s)$ ☺ quasi-optimal
- Unbalanced case: $\log H = \Theta(s) \hookrightarrow$ bit complexity $O^{\sim}(sd)$ ☹ possibly quadratic

❓ Can we always compute the polynomial product $f \times g$ in time $O^{\sim}(\text{bitlen}(f) + \text{bitlen}(g) + \text{bitlen}(f \times g))$?

Reinterpretation as a sparse interpolation problem

Imagine that you are given most of the coefficients of $f \times g$

$\Delta = x^{14} + 6x^{13} + 15x^{12} + 12x^{11} - 26x^{10} - 50x^9 - 37x^8 + 18x^7 + 28x^6 + 17x^4 + 16x^3 + 25x^2 - 6x + 9$

$\Rightarrow$ computing the whole product is a sparse interpolation problem:

$$f \times g - \Delta = -1000000006000000000x^{10} + 6000000000x^7 + 8x^5$$

Reinterpretation as a sparse interpolation problem

Imagine that you are given most of the coefficients of $f \times g$

$\Delta = x^{14} + 6x^{13} + 15x^{12} + 12x^{11} - 26x^{10} - 50x^9 - 37x^8 + 18x^7 + 28x^6 + 17x^4 + 16x^3 + 25x^2 - 6x + 9$

$\Rightarrow$ computing the whole product is a sparse interpolation problem:

$$f \times g - \Delta = -1000000006000000000x^{10} + 6000000000x^7 + 8x^5$$

Intermediate Problem

Given $f, g \in \mathbb{Z}[x]$

Compute $\Delta \approx f \times g$

Original Problem

Given $f, g \in \mathbb{Z}[x]$

Compute $h = f \times g$

New Problem

Given a way to evaluate $h - \Delta \in \mathbb{Z}[x]$

Interpolate $h - \Delta$ in dense or sparse rep.

Difficulties

- h and Δ may be unbalanced (and sparse)
- Evaluating f, g and Δ might be costly

Sparse interpolation: in a nutshell

Problem definition

Inputs: a way to evaluate $f \in \mathbb{Z}[x]$

bounds (D, H, T) s.t. $D \geq \deg f$; $H \geq \|f\|_\infty$; $T \geq \|f\|_0 = t$

Output: the sparse representation of $f = \sum_{i=1}^{t} f_i x^{e_i}$, with $f_i \neq 0$

Sparse interpolation: in a nutshell

Problem definition

Inputs: a way to evaluate $f \in \mathbb{Z}[x]$
bounds (D, H, T) s.t. $D \geq \deg f$; $H \geq \|f\|_\infty$; $T \geq \|f\|_0 = t$

Output: the sparse representation of $f = \sum_{i=1}^{t} f_i x^{e_i}$, with $f_i \neq 0$

Two main approaches with bit complexity $poly(T, \log D, \log H)$

- (Modular) Blackbox [Prony 1795; Ben-Or-Tiwary 1988, Kaltofen 2010]
- *Straight-line program* [Garg-Schost 2009]

Sparse interpolation: in a nutshell

Problem definition

Inputs: a way to evaluate $f \in \mathbb{Z}[x]$
bounds (D, H, T) s.t. $D \geq \deg f$; $H \geq \|f\|_\infty$; $T \geq \|f\|_0 = t$
Output: the sparse representation of $f = \sum_{i=1}^{t} f_i x^{e_i}$, with $f_i \neq 0$

Two main approaches with bit complexity $poly(T, \log D, \log H)$

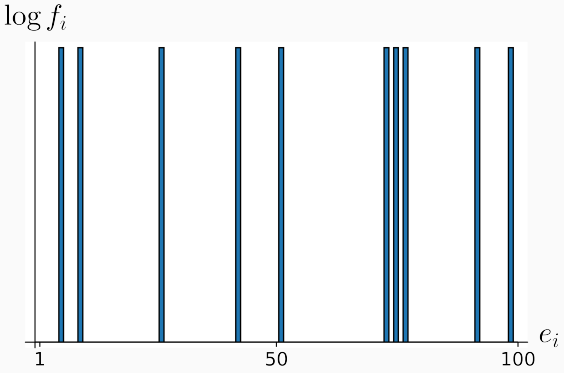
- (Modular) Blackbox [Prony 1795; Ben-Or-Tiwary 1988, Kaltofen 2010]
- *Straight-line program* [Garg-Schost 2009]

Many works to make more efficient algorithms

⇒ avoiding some bounds [Kaltofen-Lee 2003]
⇒ quasi-linear in one, few, or all parameters [Arnold-Giesbrecht-Roche 2013–2016; Huang 2019]
⇒ first fully quasi-linear [G.-Grenet-Perret du Cray-Roche 2022]

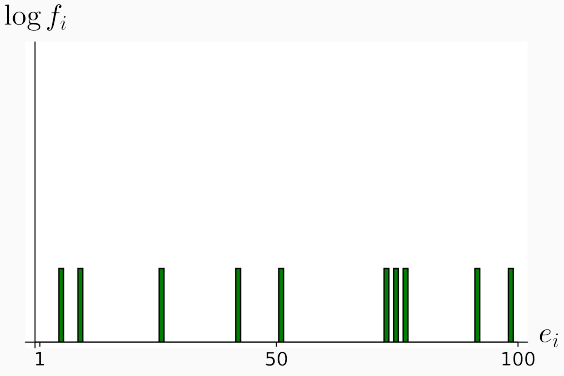
☹ all results assume f is balanced: $\text{bitlen}(f) = O(T \log D + T \log H)$

Fast balanced sparse interpolation [G.-Grenet-Perret du Cray-Roche (ISSAC 2022)]



Given bounds T, D, H and a modular black box for $f = \sum f_i x^{e_i} \in \mathbb{Z}[x]$

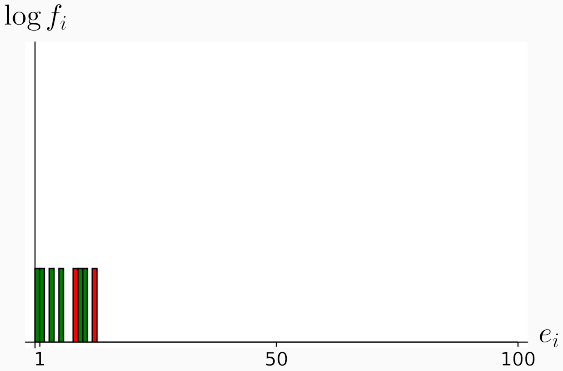
Fast balanced sparse interpolation [G.-Grenet-Perret du Cray-Roche (ISSAC 2022)]



Given bounds T, D, H and a modular black box for $f = \sum f_i x^{e_i} \in \mathbb{Z}[x]$

1. $f \bmod q$

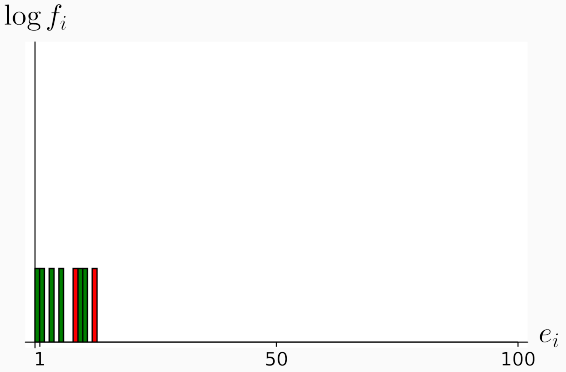
Fast balanced sparse interpolation [G.-Grenet-Perret du Cray-Roche (ISSAC 2022)]



Given bounds T, D, H and a modular black box for $f = \sum f_i x^{e_i} \in \mathbb{Z}[x]$

1. $f \bmod q$
2. $f \bmod x^p - 1$, i.e. $f(\omega)$ with ω a p-PRU in F_q

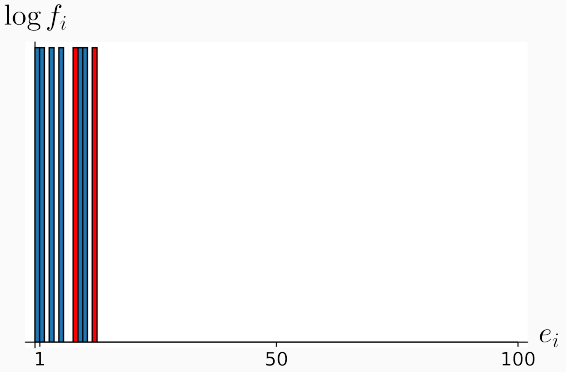
Fast balanced sparse interpolation [G.-Grenet-Perret du Cray-Roche (ISSAC 2022)]



Given bounds T, D, H and a modular black box for $f = \sum f_i x^{e_i} \in \mathbb{Z}[x]$

1. $f \bmod q$
2. $f \bmod x^p - 1$, i.e. $f(\omega)$ with ω a p-PRU in $\mathrm{F_q}$
3. Get supports of $f \bmod \langle x^p - 1, q \rangle$ *à la Prony*
   $\Rightarrow$ complexity $\tilde{O}(p \log q)$ is quasi-linear

Fast balanced sparse interpolation [G.-Grenet-Perret du Cray-Roche (ISSAC 2022)]

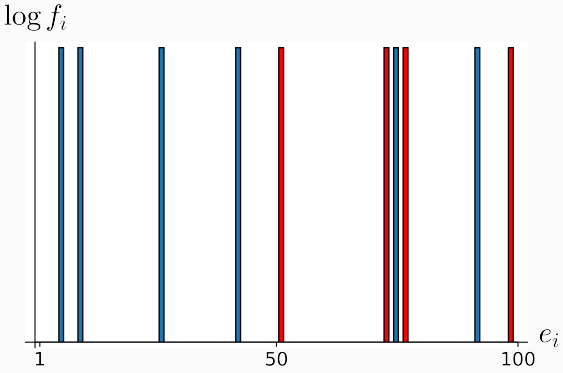


Given bounds T, D, H and a modular black box for $f = \sum f_i x^{e_i} \in \mathbb{Z}[x]$

1. $f \bmod q$
2. $f \bmod x^p - 1$, i.e. $f(\omega)$ with ω a p-PRU in $\mathrm{F_q}$
3. Get supports of $f \bmod \langle x^p - 1, q \rangle$ *à la Prony*
   $\Rightarrow$ complexity $\tilde{O}(p \log q)$ is quasi-linear

4. Sparse interpolation from known support modulo $m = q^k \geq 2DH$ $\Rightarrow$ $f \bmod x^p - 1$

Fast balanced sparse interpolation [G.-Grenet-Perret du Cray-Roche (ISSAC 2022)]

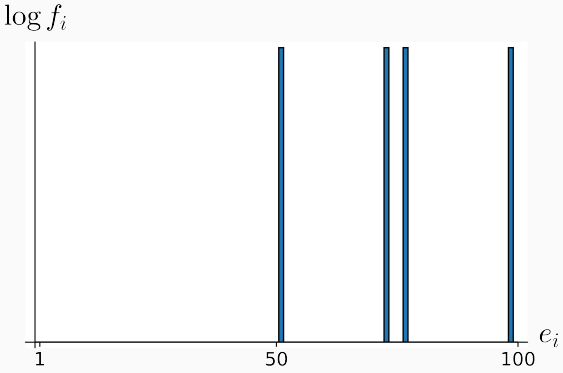


Given bounds T, D, H and a modular black box for $f = \sum f_i x^{e_i} \in \mathbb{Z}[x]$

1. $f \bmod q$
2. $f \bmod x^p - 1$, i.e. $f(\omega)$ with ω a p-PRU in $\mathrm{F_q}$
3. Get supports of $f \bmod \langle x^p - 1, q \rangle$ *à la Prony*
   $\Rightarrow$ complexity $\tilde{O}(p \log q)$ is quasi-linear

4. Sparse interpolation from known support modulo $m = q^k \geq 2DH$ $\Rightarrow$ $f \bmod x^p - 1$
5. embed e_i into coefficients : step 4 with $xf'(x) = \frac{f((1+m)x) - f(x)}{m} \bmod m^2$ $\Rightarrow$ $xf' \bmod x^p - 1$
   $\Rightarrow e_i$'s via simple division: $xf'(x)[i]/f(x)[i]$

Fast balanced sparse interpolation [G.-Grenet-Perret du Cray-Roche (ISSAC 2022)]

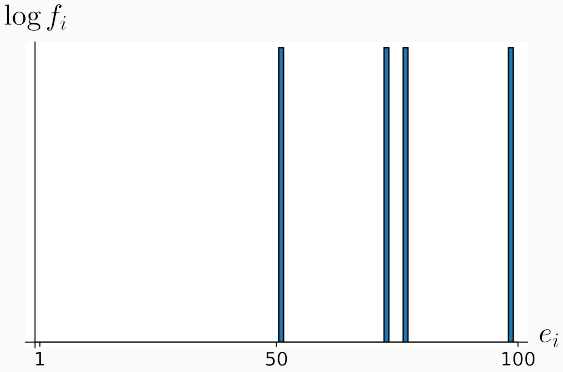


Given bounds T, D, H and a modular black box for $f = \sum f_i x^{e_i} \in \mathbb{Z}[x]$

1. $f \bmod q$
2. $f \bmod x^p - 1$, i.e. $f(\omega)$ with ω a p-PRU in $\mathrm{F_q}$
3. Get supports of $f \bmod \langle x^p - 1, q \rangle$ *à la Prony*
   $\Rightarrow$ complexity $\tilde{O}(p \log q)$ is quasi-linear

4. Sparse interpolation from known support modulo $m = q^k \geq 2DH$ $\Rightarrow$ $f \bmod x^p - 1$
5. embed e_i into coefficients : step 4 with $xf'(x) = \frac{f((1+m)x) - f(x)}{m} \bmod m^2$ $\Rightarrow$ $xf' \bmod x^p - 1$
   $\Rightarrow e_i$'s via simple division: $xf'(x)[i]/f(x)[i]$
6. Recurse on the new MBB $\pi = f - f^*$ but with sparsity bound $T/2$

Fast balanced sparse interpolation [G.-Grenet-Perret du Cray-Roche (ISSAC 2022)]



Given bounds T, D, H and a modular black box for $f = \sum f_i x^{e_i} \in \mathbb{Z}[x]$

1. $f \bmod q$
2. $f \bmod x^p - 1$, i.e. $f(\omega)$ with ω a p-PRU in $\mathrm{F_q}$
3. Get supports of $f \bmod \langle x^p - 1, q \rangle$ *à la Prony*
   $\Rightarrow$ complexity $\tilde{O}(p \log q)$ is quasi-linear

4. Sparse interpolation from known support modulo $m = q^k \geq 2DH$ $\Rightarrow$ $f \bmod x^p - 1$
5. embed e_i into coefficients : step 4 with $xf'(x) = \frac{f((1+m)x) - f(x)}{m} \bmod m^2$ $\Rightarrow$ $xf' \bmod x^p - 1$
   $\Rightarrow e_i$'s via simple division: $xf'(x)[i]/f(x)[i]$
6. Recurse on the new MBB $\pi = f - f^*$ but with sparsity bound $T/2$

$\Rightarrow$ Monte-Carlo algorithm (proba $\geq 2/3$), $O(T)$ probes to MBB and $\tilde{O}(T \log(DH))$ bit op.

Our contributions

Interpolation of sparse polynomials with unbalanced coefficients

Given a modular black box for $f \in \mathbb{Z}[x]$ with bounds $s \geq \text{bitlen}(f)$ and $D \geq \deg f$

- Our ISSAC'22 algo $\rightarrow \tilde{O}(s^2)$ since $T \leq 2s/\log s$ and $H \leq 2^s$

Our contributions

Interpolation of sparse polynomials with unbalanced coefficients

Given a modular black box for $f \in \mathbb{Z}[x]$ with bounds $s \geq \text{bitlen}(f)$ and $D \geq \deg f$

- Our ISSAC'22 algo $\rightarrow$ $\tilde{O}(s^2)$ since $T \leq 2s/\log s$ and $H \leq 2^s$
- ISSAC'24: new Monte-Carlo algorithm with proba $\geq 1 - \frac{1}{s}$
  $\hookrightarrow$ $O(s \log D \log s)$ MBB evaluations and $\tilde{O}(s \log D)$ extra bit operations

Our contributions

Interpolation of sparse polynomials with unbalanced coefficients

Given a modular black box for $f \in \mathbb{Z}[x]$ with bounds $s \geq \text{bitlen}(f)$ and $D \geq \deg f$

- Our ISSAC'22 algo $\rightarrow$ $\tilde{O}(s^2)$ since $T \leq 2s/\log s$ and $H \leq 2^s$
- ISSAC'24: new Monte-Carlo algorithm with proba $\geq 1 - \frac{1}{s}$
  $\hookrightarrow$ $O(s \log D \log s)$ MBB evaluations and $\tilde{O}(s \log D)$ extra bit operations

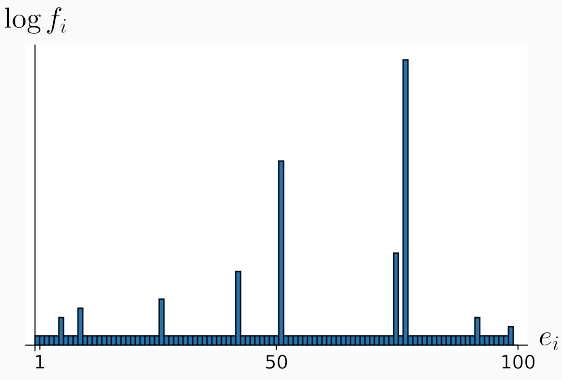
Polynomial multiplication of unbalanced polynomials

Given explicit (dense or sparse) polynomials $f, g \in \mathbb{Z}[x]$

- Probably correct and probably fast algorithm with proba $\geq 1 - \frac{1}{s}$
- Expected bit complexity $\tilde{O}(s \log D)$

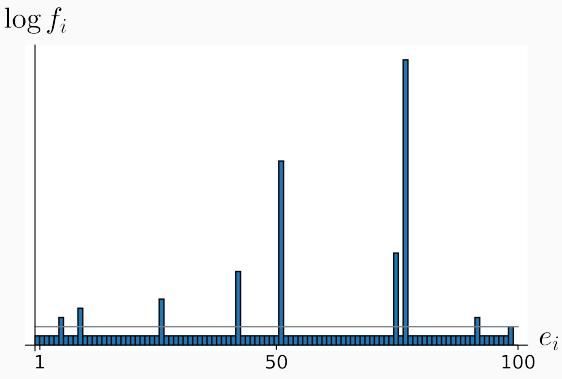
where $s = \text{bitlen}(f) + \text{bitlen}(g) + \text{bitlen}(fg)$ and $D = \deg fg$

Unbalanced (sparse) interpolation: the wrong easy solution



Given bounds $s \geq \text{bitlen}(f), D \geq \deg f$ and a MBB for unbalanced $f = \sum f_i x^{e_i} \in \mathbb{Z}[x]$

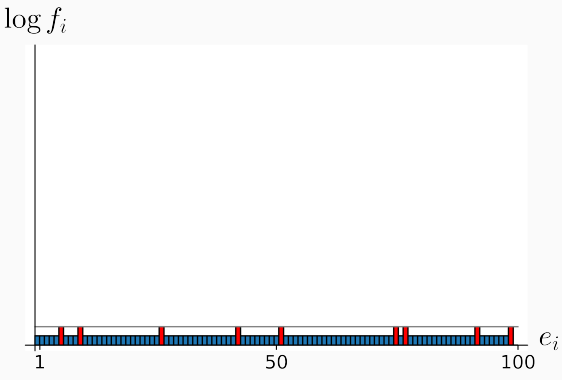
Unbalanced (sparse) interpolation: the wrong easy solution



Given bounds $s \geq \text{bitlen}(f), D \geq \deg f$ and a MBB for unbalanced $f = \sum f_i x^{e_i} \in \mathbb{Z}[x]$

1. Pick smallish m and interpolate $f^* = f \bmod m$ (*balanced case*)

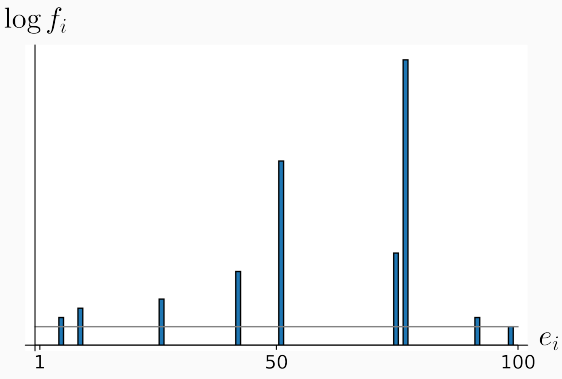
Unbalanced (sparse) interpolation: the wrong easy solution



Given bounds $s \geq \text{bitlen}(f), D \geq \deg f$ and a MBB for unbalanced $f = \sum f_i x^{e_i} \in \mathbb{Z}[x]$

1. Pick smallish m and interpolate $f^* = f \bmod m$ (*balanced case*)

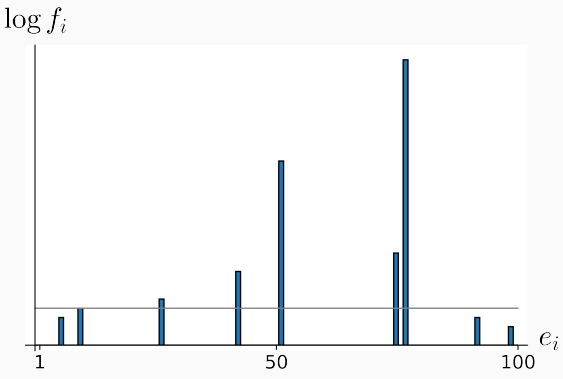
Unbalanced (sparse) interpolation: the wrong easy solution



Given bounds $s \geq \text{bitlen}(f), D \geq \deg f$ and a MBB for unbalanced $f = \sum f_i x^{e_i} \in \mathbb{Z}[x]$

1. Pick smallish m and interpolate $f^* = f \bmod m$ (*balanced case*)
2. $f - f^*$: more sparse

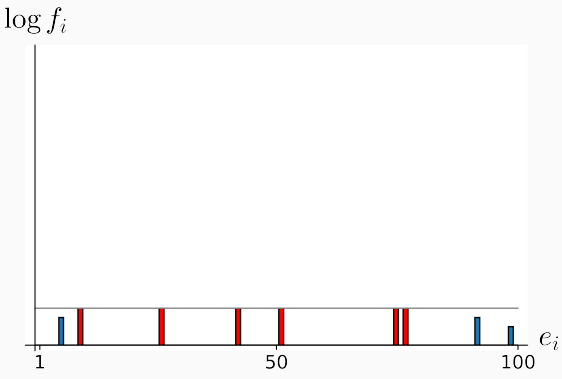
Unbalanced (sparse) interpolation: the wrong easy solution



Given bounds $s \geq \text{bitlen}(f), D \geq \deg f$ and a MBB for unbalanced $f = \sum f_i x^{e_i} \in \mathbb{Z}[x]$

1. Pick smallish m and interpolate $f^* = f \bmod m$ (*balanced case*)
2. $f - f^*$: more sparse
3. Interpolate $\pi = f - f^* \bmod m$ for double size m

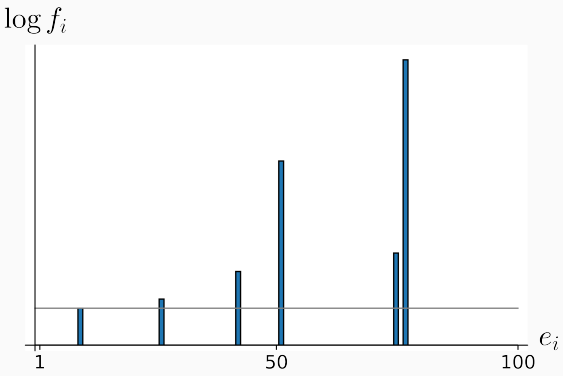
Unbalanced (sparse) interpolation: the wrong easy solution



Given bounds $s \geq \text{bitlen}(f), D \geq \deg f$ and a MBB for unbalanced $f = \sum f_i x^{e_i} \in \mathbb{Z}[x]$

1. Pick smallish m and interpolate $f^* = f \bmod m$ (*balanced case*)
2. $f - f^*$: more sparse
3. Interpolate $\pi = f - f^* \bmod m$ for double size m

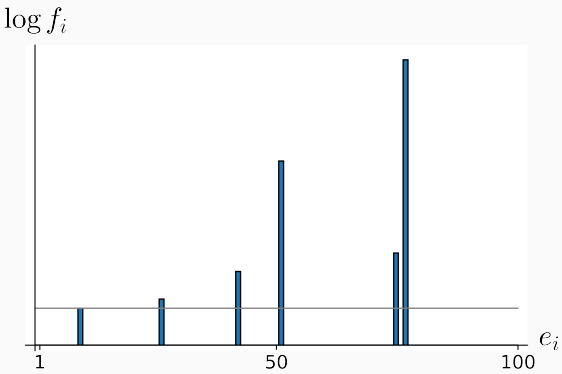
Unbalanced (sparse) interpolation: the wrong easy solution



Given bounds $s \geq \text{bitlen}(f), D \geq \deg f$ and a MBB for unbalanced $f = \sum f_i x^{e_i} \in \mathbb{Z}[x]$

1. Pick smallish m and interpolate $f^* = f \bmod m$ (*balanced case*)
2. $f - f^*$: more sparse
3. Interpolate $\pi = f - f^* \bmod m$ for double size m
4. Update π and go to step 3

Unbalanced (sparse) interpolation: the wrong easy solution

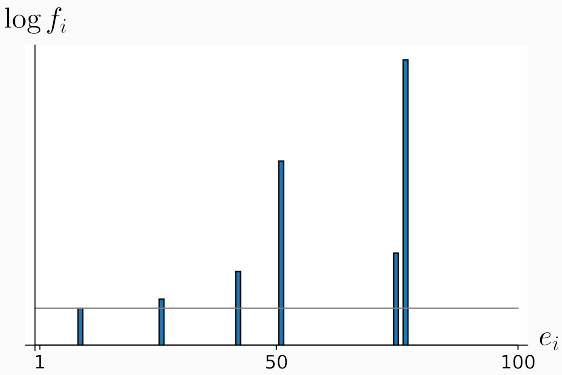


Given bounds $s \geq \text{bitlen}(f), D \geq \deg f$ and a MBB for unbalanced $f = \sum f_i x^{e_i} \in \mathbb{Z}[x]$

1. Pick smallish m and interpolate $f^* = f \bmod m$ (*balanced case*)
2. $f - f^*$: more sparse
3. Interpolate $\pi = f - f^* \bmod m$ for double size m
4. Update π and go to step 3

☺ $O(\log s)$ balanced interpolations with $T \approx s/2^k$ and $\log H \approx 2^k$

Unbalanced (sparse) interpolation: the wrong easy solution



Given bounds $s \geq \text{bitlen}(f)$, $D \geq \deg f$ and a MBB for unbalanced $f = \sum f_i x^{e_i} \in \mathbb{Z}[x]$

1. Pick smallish m and interpolate $f^* = f \bmod m$ (*balanced case*)
2. $f - f^*$: more sparse
3. Interpolate $\pi = f - f^* \bmod m$ for double size m
4. Update π and go to step 3

☺ $O(\log s)$ balanced interpolations with $T \approx s/2^k$ and $\log H \approx 2^k$

☹ at some point $m = O(2^s)$, and sparsity of f^* might be $O(s)$
↪ one evaluation of π costs $\tilde{O}(s^2)$ due to explicit representation of f^*

!!! leads to a quadratic time complexity !!!

Unbalanced (sparse) interpolation: our efficient solution

Using a top-down approach

- First recover some *huge terms* f^* of f
- Recursively interpolate $f - f^* \Rightarrow$ *more balanced, smaller coeff, but same sparsity*

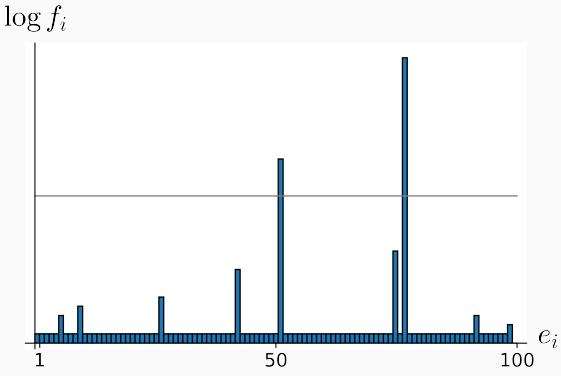
Unbalanced (sparse) interpolation: our efficient solution

Using a top-down approach

- First recover some *huge terms* f^* of f
- Recursively interpolate $f - f^* \Rightarrow$ *more balanced, smaller coeff, but same sparsity*

Illustration:

1. Account for sparsity of *huge terms*



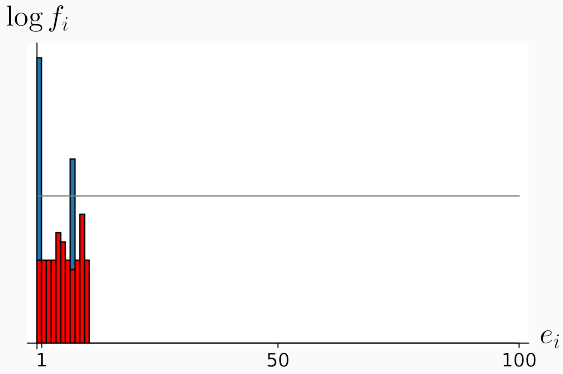
Unbalanced (sparse) interpolation: our efficient solution

Using a top-down approach

- First recover some *huge terms* f^* of f
- Recursively interpolate $f - f^* \Rightarrow$ *more balanced, smaller coeff, but same sparsity*

Illustration:

1. Account for sparsity of *huge terms*
2. Interpolate $(f, xf') \bmod \langle x^p - 1, m \rangle$
   $\hookrightarrow$ dense ok if $p \log m = \tilde{O}(s \log D)$



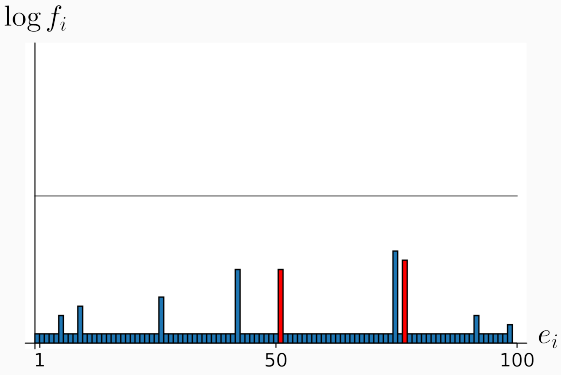
Unbalanced (sparse) interpolation: our efficient solution

Using a top-down approach

- First recover some *huge terms* f^* of f
- Recursively interpolate $f - f^* \Rightarrow$ *more balanced, smaller coeff, but same sparsity*

Illustration:

1. Account for sparsity of *huge terms*
2. Interpolate $(f, xf') \bmod \langle x^p - 1, m \rangle$
   $\hookrightarrow$ dense ok if $p \log m = \tilde{O}(s \log D)$
3. Identify and remove *huge terms*



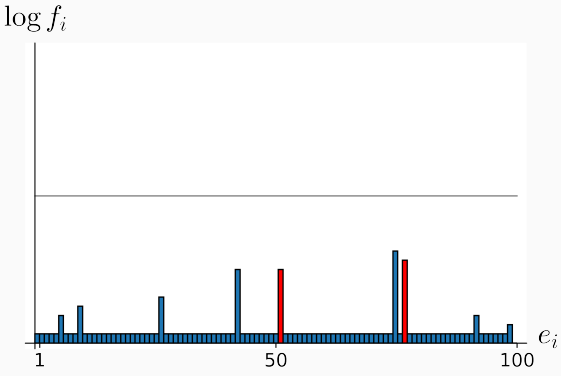
Unbalanced (sparse) interpolation: our efficient solution

Using a top-down approach

- First recover some *huge terms* f^* of f
- Recursively interpolate $f - f^* \Rightarrow$ *more balanced, smaller coeff, but same sparsity*

Illustration:

1. Account for sparsity of *huge terms*
2. Interpolate $(f, xf') \bmod \langle x^p - 1, m \rangle$
   $\hookrightarrow$ dense ok if $p \log m = \tilde{O}(s \log D)$
3. Identify and remove *huge terms*

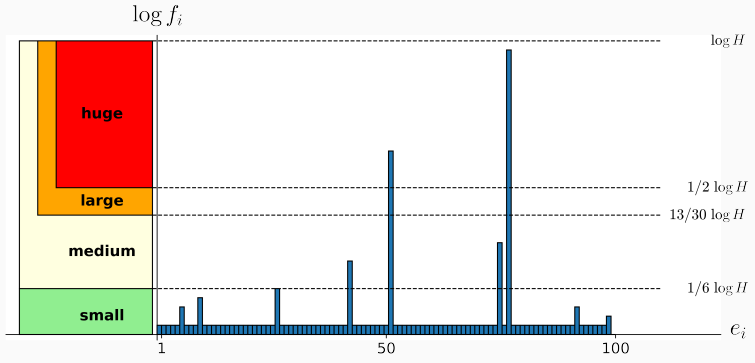


Difficulties

- Expression swell with collisions
- Embedding exponents into coeffs

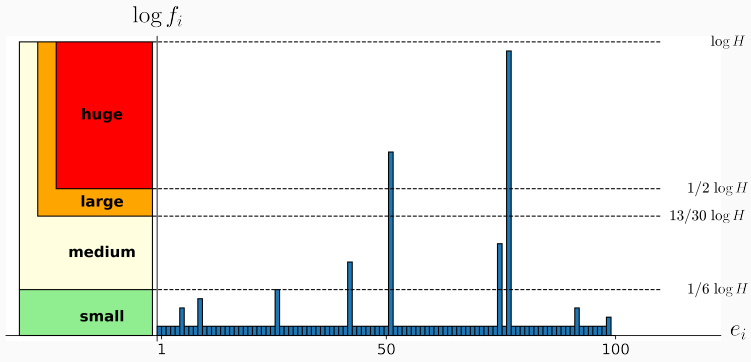
Recovering the huge terms

huge terms are those beyond half the maximal bit-length (e.g. $\geq 2^{s/2}$)



Recovering the huge terms

huge terms are those beyond half the maximal bit-length (e.g. $\geq 2^{s/2}$)

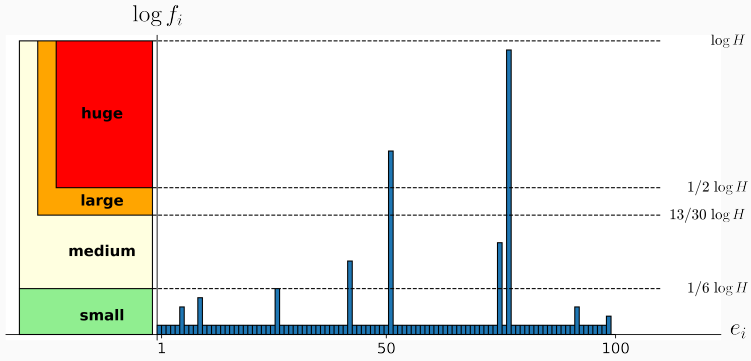


On terms' collision:

- **only** one large/many small still allows exponent decoding using $xf'(x)$
- **only** non-large/non-large do not produce erroneous huge terms

Recovering the huge terms

huge terms are those beyond half the maximal bit-length (e.g. $\geq 2^{s/2}$)

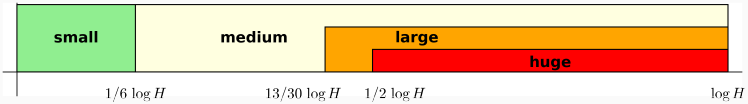


On terms' collision:

- **only** one large/many small still allows exponent decoding using $xf'(x)$
- **only** non-large/non-large do not produce erroneous huge terms

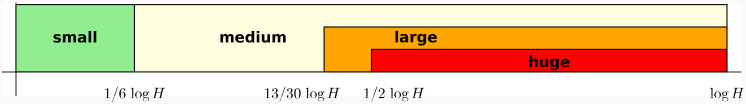
↪ reconstructed huge terms **must** avoid large/medium collisions

Recovering the huge terms: controlling the collisions



Remarks: it is hard to avoid all large / medium collision in $f \bmod x^p - 1$ with $p = O(s)$

Recovering the huge terms: controlling the collisions



Remarks: it is hard to avoid all large / medium collision in $f \bmod x^p - 1$ with $p = O(s)$

Our solution

1. Compute a superset $\mathcal{T}$ of the large terms exponents (erroneous= large / medium)
2. Use $\mathcal{T}$ to filter out large / medium collisions in the reconstruction of huge terms
   ↪ (somehow) interpolation with overestimated support

Recovering the huge terms: superset of large terms

Lemma [G.-Grenet-Perret du Cray-Roche 2024]

Let $c_0 x^{e_0}$ be any large terms of f. If $c_0 x^{e_0}$ only collides with many small terms $c_i x^{e_i}$:

$$e_0 = \left\lfloor \frac{c_0 e_0 + \sum_{i \in \mathcal{S}} c_i e_i}{c_0 + \sum_{i \in \mathcal{S}} c_i} \right\rceil \quad \text{assuming } m \geq 4H^{7/6}, \quad \log H = \Omega(\log s + \log D)$$

Recovering the huge terms: superset of large terms

Lemma [G.-Grenet-Perret du Cray-Roche 2024]

Let $c_0 x^{e_0}$ be any large terms of f. If $c_0 x^{e_0}$ only collides with many small terms $c_i x^{e_i}$:

$$e_0 = \left\lfloor \frac{c_0 e_0 + \sum_{i \in \mathcal{S}} c_i e_i}{c_0 + \sum_{i \in \mathcal{S}} c_i} \right\rceil \quad \text{assuming } m \geq 4H^{7/6}, \quad \log H = \Omega(\log s + \log D)$$

Algorithm sketch for computing $\mathcal{T}$:

- Take p so that most collisions are large/small — $p = O(s \log D / \log H)$
- Dense interpolation of $(f, xf') \bmod \langle x^p - 1, m \rangle$ — $\log m = O(\log H)$
  ↪ add to $\mathcal{T}$ the coefficient-wise divisions
- Repeat for $O(\log s)$ random primes p

Recovering the huge terms: superset of large terms

Lemma [G.-Grenet-Perret du Cray-Roche 2024]

Let $c_0 x^{e_0}$ be any large terms of f. If $c_0 x^{e_0}$ only collides with many small terms $c_i x^{e_i}$:

$$e_0 = \left\lfloor \frac{c_0 e_0 + \sum_{i \in \mathcal{S}} c_i e_i}{c_0 + \sum_{i \in \mathcal{S}} c_i} \right\rceil \quad \text{assuming } m \geq 4H^{7/6}, \quad \log H = \Omega(\log s + \log D)$$

Algorithm sketch for computing $\mathcal{T}$:

- Take p so that most collisions are large/small — $p = O(s \log D / \log H)$
- Dense interpolation of $(f, xf') \bmod \langle x^p - 1, m \rangle$ — $\log m = O(\log H)$
  ↪ add to $\mathcal{T}$ the coefficient-wise divisions
- Repeat for $O(\log s)$ random primes p

Remark

- The superset is not too large: $\#\mathcal{T} = O(s \log s / \log H)$, overestimated by $O(\log s)$
- Bit complexity: $O^{\sim}(s \log D)$

Algorithm for computing all huge terms

Idea: use $\mathcal{T}$ to detect erroneous huge terms in $f \bmod \langle x^p - 1, m \rangle$

Algorithm for computing all huge terms

Idea: use $\mathcal{T}$ to detect erroneous huge terms in $f \bmod \langle x^p - 1, m \rangle$

Algorithm sketch:

- Take p so that most terms in $\mathcal{T}$ do not collide — $p = O(\#\mathcal{T} \log D)$
- Let $cx^{e \bmod p}$ be any huge term of $f \bmod \langle x^p - 1, m \rangle$ — $\log m = O(\log H)$
  ↪ add cx^e to f^* **only if** $e \bmod p$ is collision-free in $\mathcal{T} \bmod p$
- Repeat for $O(\log s)$ random primes p — *(abort if* $\text{bitlen}(f^*) > s$*)*

Algorithm for computing all huge terms

Idea: use $\mathcal{T}$ to detect erroneous huge terms in $f \bmod \langle x^p - 1, m \rangle$

Algorithm sketch:

- Take p so that most terms in $\mathcal{T}$ do not collide — $p = O(\#\mathcal{T} \log D)$
- Let $cx^{e \bmod p}$ be any huge term of $f \bmod \langle x^p - 1, m \rangle$ — $\log m = O(\log H)$
  ↪ add cx^e to f^* **only if** $e \bmod p$ is collision-free in $\mathcal{T} \bmod p$
- Repeat for $O(\log s)$ random primes p — *(abort if* $\text{bitlen}(f^*) > s$*)*

Remark

- All huge terms of f are added to f^*, plus some non-large collisions, still $\|f - f^*\|_\infty < \sqrt{H}$
- Bit complexity: $O(s \log D \log^3 s \log\log H) = O^{\sim}(s \log D)$

Our unbalanced sparse interpolation algorithm

Algorithm

Input: Modular black box for $f \in \mathbb{Z}[x]$ and bounds $s \geq \text{bitlen}(f)$, $D \geq \deg(f)$
Output: an explicit $f^* \in \mathbb{Z}[x]$ or error if $\text{bitlen}(f^*) > s$

1. $H \leftarrow 2^s, f^* \leftarrow 0$
2. while $H \geq \max(61, 15 \log s, 6 \log D)$
3. compute the huge terms of $(f - f^*)$ and update f^*
4. $H \leftarrow \sqrt{H}$
5. compute remaining terms of $(f - f^*)$ *via* (balanced) sparse interpolation

Theorem [G.-Grenet-Perret du Cray-Roche 2024]

The algorithm returns an explicit representation of f with probability $\geq 1 - 1/s$, using $\tilde{O}(s \log D)$ bit operations and $O(s \log D \log s)$ MBB evaluations.

Back to our application

Unbalanced polynomial multiplication

Inputs: unbalanced polynomials $f, g \in \mathbb{Z}[x]$ with $\text{bitlen}(f), \text{bitlen}(g) \leq \ell$

Output: $h = f \times g \in \mathbb{Z}[x]$ with $s = \text{bitlen}(h) + 2\ell$

Back to our application

Unbalanced polynomial multiplication

Inputs: unbalanced polynomials $f, g \in \mathbb{Z}[x]$ with $\text{bitlen}(f), \text{bitlen}(g) \leq \ell$
Output: $h = f \times g \in \mathbb{Z}[x]$ with $s = \text{bitlen}(h) + 2\ell$

- If s known $\Rightarrow$ direct application of our sparse interpolation: $\tilde{O}(s \log D)$
- Otherwise, need a bound estimation:

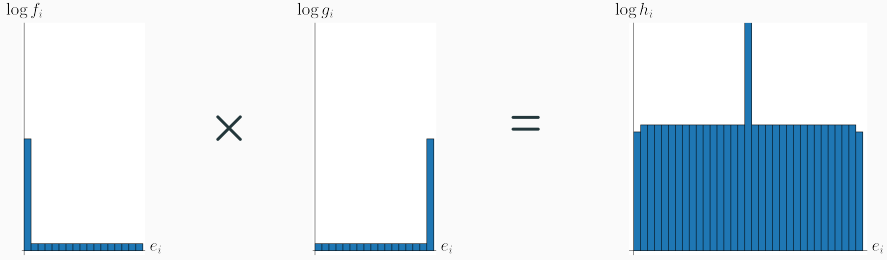
Back to our application

Unbalanced polynomial multiplication

Inputs: unbalanced polynomials $f, g \in \mathbb{Z}[x]$ with $\text{bitlen}(f), \text{bitlen}(g) \leq \ell$
Output: $h = f \times g \in \mathbb{Z}[x]$ with $s = \text{bitlen}(h) + 2\ell$

- If s known $\Rightarrow$ direct application of our sparse interpolation: $\tilde{O}(s \log D)$
- Otherwise, need a bound estimation: might be quadratic

$$\text{bitlen}(h) \leq 4\ell^2 / \log \ell \quad \text{and} \quad \|h\|_\infty \leq 4^\ell \ell$$



Our unbalanced polynomial multiplication algorithm

Idea: guess $s \geq \text{bitlen } h$ through probabilistic verification of $h = f \times g$

Our unbalanced polynomial multiplication algorithm

Idea: guess $s \geq \text{bitlen}\, h$ through probabilistic verification of $h = f \times g$

Algorithm sketch:

- Unbalanced sparse interpolation of MBB $f \times g$ with tentative bound $s \overset{?}{\geq} \text{bitlen}(fg)$
- *Probabilistically verify* whether $h = f \times g$ in $\tilde{O}(s \log D)$ [G.-Grenet-Perret du Cray 2023]
- Start with $s = \ell$ and *double* its value until verification succeeds or $4^\ell / \log \ell$ is reached

Our unbalanced polynomial multiplication algorithm

Idea: guess $s \geq \text{bitlen}\, h$ through probabilistic verification of $h = f \times g$

Algorithm sketch:

- Unbalanced sparse interpolation of MBB $f \times g$ with tentative bound $s \overset{?}{\geq} \text{bitlen}(fg)$
- *Probabilistically verify* whether $h = f \times g$ in $O\tilde{}(s \log D)$ [G.-Grenet-Perret du Cray 2023]
- Start with $s = \ell$ and *double* its value until verification succeeds or $4^\ell / \log \ell$ is reached

Probably correct and probably fast

- Both with probability $\geq 1 - 1/s$
- Expected bit complexity $O\tilde{}(s \log D \log^5 s (\log\log s)^2) = O\tilde{}(s \log D)$

Summary of results

Unbalanced (sparse or dense) polynomials

New complexity of $\tilde{O}(s \log D)$ bit operations

- **Interpolation**: Monte-Carlo algorithm with proba $\geq 1 - 1/s$
- **Multiplication**: probably correct probably fast algorithm with proba $\geq 1 - 1/s$

Summary of results

Unbalanced (sparse or dense) polynomials

New complexity of $\tilde{O}(s \log D)$ bit operations

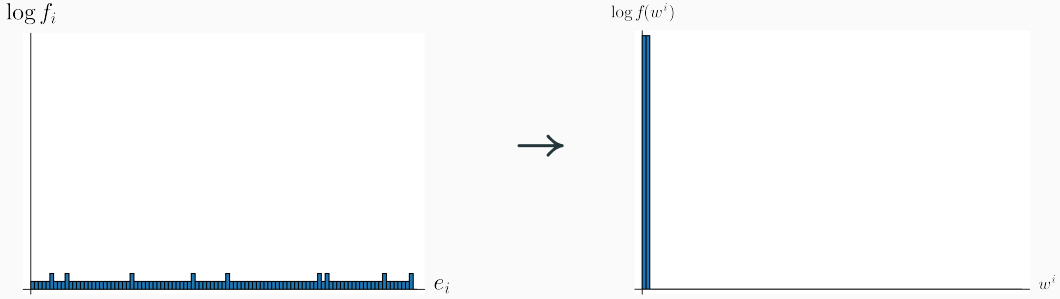
- **Interpolation**: Monte-Carlo algorithm with proba $\geq 1 - 1/s$
- **Multiplication**: probably correct probably fast algorithm with proba $\geq 1 - 1/s$

☺ quasi linear for dense and moderately sparse polynomials, i.e. $\log D = poly(\log s)$

☹ not for supersparse polynomials, i.e. $\log D = poly(s)$

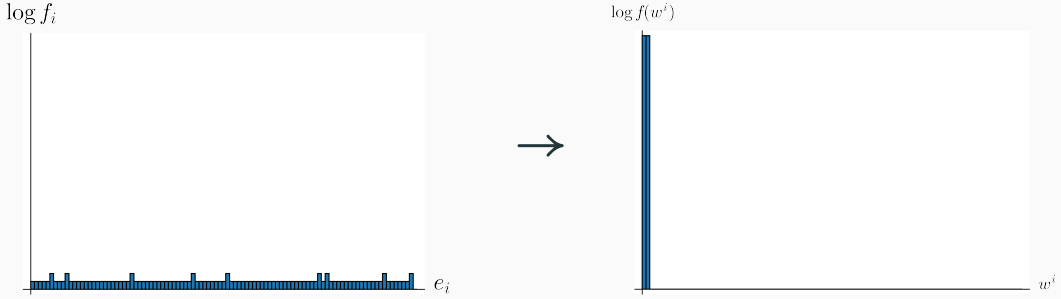
Open problems

- Algorithms of complexity $\tilde{O}(s)$ for unbalanced polynomials
  ↪ difficulty $p \geq \log D$ to avoid collisions
- How to remove the *a priori* bounds s and $\log D$ in the interpolation ?
- Fast evaluation of large low-height polynomial at few points to very high precision



Open problems

- Algorithms of complexity $\tilde{O}(s)$ for unbalanced polynomials
  ↪ difficulty $p \geq \log D$ to avoid collisions
- How to remove the *a priori* bounds s and $\log D$ in the interpolation ?
- Fast evaluation of large low-height polynomial at few points to very high precision



Thank you !

Funding

Ce travail a bénéficié d'une aide de l'État gérée par l'Agence Nationale de la Recherche au titre de France 2030 portant la référence ANR-22-PECY-0010

Under the carpet: MBB/polynomial evaluations

- The input is **always** $\pi = f - f^*$ $\quad \mathrm{bitlen}(f), \mathrm{bitlen}(f^*) \leq s$
- Need to evaluate π on $(1, \omega, \omega^2, \ldots, \omega^{p-1})$ for interpolating $\pi \bmod \langle x^p - 1, m \rangle$
  $\hookrightarrow$ ω a p-PRU in $\mathbb{Z}/m\mathbb{Z}$,
  $\hookrightarrow$ when m increases, p decreases s.t. $p \log m = O(s \log D \log H)$

Under the carpet: MBB/polynomial evaluations

- The input is **always** $\pi = f - f^*$ $\quad \mathrm{bitlen}(f), \mathrm{bitlen}(f^*) \leq s$
- Need to evaluate π on $(1, \omega, \omega^2, \ldots, \omega^{p-1})$ for interpolating $\pi \bmod \langle x^p - 1, m \rangle$
  $\hookrightarrow$ ω a p-PRU in $\mathbb{Z}/m\mathbb{Z}$,
  $\hookrightarrow$ when m increases, p decreases s.t. $p \log m = O(s \log D \log H)$

Evaluation cost for explicit polynomial f^*

1. First, reduce $f^* \bmod \langle x^p - 1, m \rangle \rightarrow O(s \log\log m + s \log\log p)$
2. Then, evaluate with (dense) Bluestein's FFT $\rightarrow O(p \log p \log m \log\log m)$

$\Rightarrow$ overall cost is $O\tilde{}(s \log D)$ bit op.

Under the carpet: MBB/polynomial evaluations

- The input is **always** $\pi = f - f^*$ $\quad \text{bitlen}(f), \text{bitlen}(f^*) \leq s$
- Need to evaluate π on $(1, \omega, \omega^2, \ldots, \omega^{p-1})$ for interpolating $\pi \bmod \langle x^p - 1, m \rangle$
  $\hookrightarrow$ ω a p-PRU in $\mathbb{Z}/m\mathbb{Z}$,
  $\hookrightarrow$ when m increases, p decreases s.t. $p \log m = O(s \log D \log H)$

Evaluation cost for explicit polynomial f^*

1. First, reduce $f^* \bmod \langle x^p - 1, m \rangle \rightarrow O(s \log\log m + s \log\log p)$
2. Then, evaluate with (dense) Bluestein's FFT $\rightarrow O(p \log p \log m \log\log m)$

$\Rightarrow$ overall cost is $O^{\sim}(s \log D)$ bit op.

Evaluation cost of the MBB for f

- Let $f(\alpha) \bmod m$ costs $O(B + L \log m \log\log m)$ bit op.
- Our algo need $O^{\sim}((B + L)s \log D)$ bit op.