Sparse interpolation over the integers with an application

Bruno Grenet[1]

LJK – Université Grenoble-Alpes

Séminaire Maths Info, Toulouse
May 23., 2024

[1]Based on joint works with P. Giorgi, A. Perret du Cray and D. S. Roche

(Vague) definition of the problem

Sparse interpolation

Inputs: A way to *evaluate* a sparse polynomial $f \in R[x]$
Bounds $D \geq \deg(f)$, $H \geq f_\infty$ and/or $T \geq f_\#$ (optional)
Output: The sparse representation of f

Sparse representation

$$f = \sum_{i=0}^{t-1} c_i x^{e_i}, \, c_i \in R_{\neq 0}$$

Degree: $\deg(f) = \max_i e_i$
Sparsity: $f_\# = t$
Height: $f_\infty = \max_i H(c_i)$ where $H(p_i/q_i) = \max(|p_i|, |q_i|)$ if $c_i \in \mathbb{Q}$

Many variants

Ring of coefficients

- $\mathbb{Z}$ or $\mathbb{Q}$ — size growth $\to$ modular techniques
- $\mathbb{R}$ or $\mathbb{C}$ — precision issues
- Finite fields — large/small size/characteristic
- Modular rings — non-units

Number of variables

- Univariate polynomials
- Multivariate polynomials — Kronecker substitution $\to$ univariate case

Input representation

- Fixed evaluations
- Black box
- Arithmetic circuit / SLP

Many variants

Ring of coefficients

- $\mathbb{Z}$ or $\mathbb{Q}$ — size growth $\to$ modular techniques
- $\mathbb{R}$ or $\mathbb{C}$ — precision issues
- Finite fields — large/small size/characteristic
- Modular rings — non-units

Number of variables

- Univariate polynomials
- Multivariate polynomials — Kronecker substitution $\to$ univariate case

Input representation

- Fixed evaluations
- Black box
- Arithmetic circuit / SLP

Contents

Contents

Sparse polynomials & linearly recurrent sequences

$$f = \sum_{i=0}^{t-1} c_i x^{e_i} \to \begin{pmatrix} f(1) \\ f(\omega) \\ \vdots \\ f(\omega^n) \end{pmatrix} = \begin{pmatrix} 1 & \cdots & 1 \\ \omega^{e_0} & \cdots & \omega^{e_{t-1}} \\ \vdots & & \vdots \\ \omega^{ne_0} & \cdots & \omega^{ne_{t-1}} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{t-1} \end{pmatrix}$$

Theorem [Blahut (1979)]

Let $f = \sum_{i=0}^{t-1} c_i x^{e_i} \in R[X]_{<D}$ where R is an integral domain and $\omega \in R$ be a principal root of unity of order $\geq D$. Then the minimal polynomial of $(f(\omega^j))_{j\geq 0}$ is $\Lambda(x) = \prod_{i=0}^{t-1}(x - \omega^{e_i})$.

Proof sketch

- Minimal polynomial of $(c_i \omega^{je_i})_j : x - \omega^{e_i}$
- Minimal polynomial of a sum = LCM of their minimal polynomials

From $\overrightarrow{F} = (f(1), \ldots, f(\omega^{2t-1}))$, compute $\Lambda = \prod_{i=0}^{t-1}(x - \omega^{e_i})$ to get $e_0, \ldots, e_{t-1}$

Sparse interpolation with known exponents

$$f = \sum_{i=0}^{t-1} c_i x^{e_i} \to \begin{pmatrix} f(1) \\ f(\omega) \\ \vdots \\ f(\omega^n) \end{pmatrix} = \begin{pmatrix} 1 & \cdots & 1 \\ \omega^{e_0} & \cdots & \omega^{e_{t-1}} \\ \vdots & & \vdots \\ \omega^{ne_0} & \cdots & \omega^{ne_{t-1}} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{t-1} \end{pmatrix}$$

Remark

Sparse interpolation on geometric sequence with known exponents

$\iff$ transposed Vandermonde system solving

Fast algorithm

- Vandermonde system solving $\iff$ (dense) polynomial interpolation
  - $O(\mathsf{M}(t) \log t)$ [Borodin-Moenck (1974)]
- Transposition $\to$ same complexity [Kaltofen-Lakshman (1992), Bostan-Lecerf-Schost (2003)]

From $\overrightarrow{F}$ and $e_0, \ldots, e_{t-1}$, compute $c_0, \ldots, c_{t-1}$

Algorithm *à la* Prony / Ben-Or–Tiwari

[Prony (1795), Ben-Or–Tiwari (1988), …]

Algorithm

Inputs: Black box for $f \in \mathbb{F}_q[x]$, $q \geq \deg(f)$
Bound $T \geq f_\#$

1. Evaluate f at $1, \omega, \dots, \omega^{2T-1}$ *where ω has order $\geq 2T$*
2. Compute the minimal polynomial Λ of $(f(\omega^j))_j$
3. Compute its roots $\beta_0, \dots, \beta_{t-1}$ and obtain the exponents $e_0, \dots, e_{t-1}$
4. Solve the transposed Vandermonde system to get the coefficients $c_0, \dots, c_{t-1}$

Complexity analysis

1. $2T$ black box evaluations
2. $O(\mathsf{M}(T) \log T)$ [Berlekamp (1968), Massey (1969), Beckermann-Labahn (1994)]
3. i. $O(\mathsf{M}(t) \log t \log q)$ [Berlekamp (1970), Rabin (1980)]
   ii. $O(\sqrt{D})$ [Shanks (1971), Heiman (1992)]
4. $O(\mathsf{M}(t) \log t)$ [Kaltofen-Lakshman (1992), Bostan-Lecerf-Schost (2003)]

Algorithm *à la* Prony / Ben-Or–Tiwari

[Prony (1795), Ben-Or–Tiwari (1988), …]

Algorithm

Inputs: Black box for $f \in \mathbb{F}_q[x]$, $q \geq \deg(f)$
Bound $T \geq f_\#$

1. Evaluate f at $1, \omega, \dots, \omega^{2T-1}$ *where ω has order $\geq 2T$*
2. Compute the minimal polynomial Λ of $(f(\omega^j))_j$
3. Compute its roots $\beta_0, \dots, \beta_{t-1}$ and obtain the exponents $e_0, \dots, e_{t-1}$
4. Solve the transposed Vandermonde system to get the coefficients $c_0, \dots, c_{t-1}$

Complexity analysis

1. $2T$ black box evaluations
2. $O(\mathsf{M}(T) \log T)$ [Berlekamp (1968), Massey (1969), Beckermann-Labahn (1994)]
3. i. $O(\mathsf{M}(t) \log t \log q)$ [Berlekamp (1970), Rabin (1980)]
   ii. $O(\sqrt{D})$ [Shanks (1971), Heiman (1992)]
4. $O(\mathsf{M}(t) \log t)$ [Kaltofen-Lakshman (1992), Bostan-Lecerf-Schost (2003)]

Remarks on Prony / Ben-Or–Tiwari algorithm

Complexity

- Quasi-linear in T, linear (optimal) number of evaluations
- Polynomial in D, rather than $\log D \to$ not polynomial in the output size
- Bound $T \geq f_\#$ not required $\to$ *early termination* [Kaltofen-Lee (2003)]

Other base rings

- Original Ben-Or–Tiwari's algorithm for $\mathbb{Z}[x_1, \dots, x_n]$
  - large evaluations $\to$ bit size $O(D)$
  - replace ω by $(p_1, \dots, p_n)$
- Small finite fields $\to$ use an extension *extended black box*
- Rings: works as long as ω is a *principal* root of unity of large order
- Fast variant over $\mathbb{Q}$ [Kaltofen (1988/2010)]
  - Compute *modulo* p where $p - 1$ is smooth
  - Use fast discrete logarithm [Pohlig-Hellman (1978)]
  - Complexity polynomial in T and $\log D$

Contents

Using cyclic extensions

- From an SLP, f can be computed explicitly in time $O(D)$ *expression swell*
- Compute $f \bmod x^p - 1 = \sum_i c_i x^{e_i \bmod p}$ for some prime p [Garg-Schost (2009)]

Loss of information

- Exponents known only *modulo p*
- Possible *collisions* between monomials

Reconstruction of full exponents

- Use several p_j's and (polynomial) Chinese remaindering, *diversification*, …
  [Garg-Schost (2009), Giesbrecht-Roche (2011), …]
- Embed exponents into coefficients [Arnold-Roche (2015), Huang (2019)]

Deal with collisions

- Large enough prime and/or many primes to avoid any collision [Garg-Schost (2009)]
- Accept some collisions and correct errors [Arnold-Giesbrecht-Roche (2013), Huang (2019)]

Using cyclic extensions

- From an SLP, f can be computed explicitly in time $O(D)$ *expression swell*
- Compute $f \bmod x^p - 1 = \sum_i c_i x^{e_i \bmod p}$ for some prime p [Garg-Schost (2009)]

Loss of information

- Exponents known only *modulo* p
- Possible *collisions* between monomials

Reconstruction of full exponents

- Use several p_j's and (polynomial) Chinese remaindering, *diversification*, …
  [Garg-Schost (2009), Giesbrecht-Roche (2011), …]
- Embed exponents into coefficients [Arnold-Roche (2015), Huang (2019)]

Deal with collisions

- Large enough prime and/or many primes to avoid any collision [Garg-Schost (2009)]
- Accept some collisions and correct errors [Arnold-Giesbrecht-Roche (2013), Huang (2019)]

Embedding exponents into coefficients

Using derivatives

[Huang 2019]

- If $f = \sum_i c_i x^{e_i}$, $x \cdot f'(x) = \sum_i c_i e_i x^{e_i}$
- Use of automatic differentiation [Baur–Strassen (1983)]

À la Paillier

[Arnold–Roche (2015)]

- If $f \in \mathbb{F}_q[x]$, evaluate $f((1+q)x)$ over $\mathbb{Z}/q^2\mathbb{Z}$
- *Modulo* q^2, $(1+q)^{e_i} = 1 + e_i q$

$$f(x) = \sum_i c_i x^{e_i} \rightsquigarrow f((1+q)x) = \sum_i c_i(1+e_i q)x^{e_i}$$

- *Remark:* $f((1+q)x) - f(x) = \sum_i c_i e_i q x^{e_i} = qx \cdot f'(x)$

Requirements

- Both techniques require e_i to be exactly representable in $\mathbb{F}_q$
- $\mathbb{F}_q$ should have characteristic $\geq \deg(f)$

Managing collisions

- Collision: monomials x^{e_i}, x^{e_j} such that $e_i \equiv e_j \bmod p$
- Collision-free monomial: x^{e_i} such that $e_i \not\equiv e_j \bmod p$ for $j \neq i$

Avoiding or limiting collisions

Let p be a random prime in $[\lambda, 2\lambda]$

- For $\lambda = \Omega(\frac{1}{\varepsilon} t^2 \log D)$, there is no collision with prob. $\geq 1 - \varepsilon$
- For $\lambda = \Omega(\frac{1}{\varepsilon} t \log D)$, there are $\geq \frac{2}{3} t$ collision-free monomials with prob. $\geq 1 - \varepsilon$

Dealing with collisions

- With $\geq \frac{2}{3} t$ collision-free monomials, there are at most $\frac{1}{6} t$ collisions
- Each collision may produce one error
- If each collision-free monomial is correctly reconstructed, we get f^* such that

$$(f - f^*)_\# \leq \frac{1}{3} f_\# + \frac{1}{6} f_\# = \frac{1}{2} f_\#$$

Algorithm *à la* Garg–Schost

[Garg-Schost (2009), Huang (2019)]

Algorithm

Inputs: SLP for $f \in \mathbb{F}_q[x]$, $char(\mathbb{F}_q) \geq \deg(f)$
Bounds $T \geq f_\#$, $D \geq \deg f$

Output: The sparse representation of f *w.h.p.*

1. $f^* \leftarrow 0$
2. Repeat $\log(T)$ times:
3. $\quad p \leftarrow$ random prime in $[\lambda, 2\lambda]$ for $\lambda = O(T \log D \log T)$
4. $\quad (f_p^{(0)}, f_p^{(1)}) \leftarrow (f \bmod x^p - 1, x \cdot f' \bmod x^p - 1)$ — *SLP for f' + dense arith.*
5. $\quad$ For each pair $\begin{cases} cx^d & \in f_p^{(0)} \\ c'x^d & \in f_p^{(1)} \end{cases}$: add $c \cdot x^{c'/c}$ to f^* — *if $c'/c \in \{0, \ldots, D-1\}$*
6. Return f^*

Complexity analysis

- $O(\log T)$ *probes* of the circuit $\to O(s \cdot \mathsf{M}(p) \cdot \log(T))$ — *s: SLP size*
- $\tilde{O}(sT \log D)$ operations in $\mathbb{F}_q \to \tilde{O}(sT \log D \log q)$ bit operations

Remarks on Garg–Schost algorithm

Almost quasi-linear!

- Output size: $O(T(\log D + \log q))$, complexity: $\tilde{O}(T \log D \log q)$
- Hard to avoid: *probing* the circuit is already non-quasi-linear

Other base rings

- Smaller characteristic
  - No exponent embedding anymore
  - Several techniques, such as *diversification*
  - Best complexity: $O(sT \log^2 D(\log D + \log q))$ [Arnold-Giesbrecht-Roche (2014)]
- Over the integers
  - Coefficient growth $\to$ modular techniques
  - Best complexity: $O(sT \log^3 D \log H)$ where $H \geq f_\infty$ [Perret du Cray (2023)]

Contents

Result

Inputs: *Modular black box* for $f \in \mathbb{Z}[x]$
Bounds $T \geq f_{\#}$, $D \geq \deg(f)$, $H \geq f_{\infty}$
Complexity: $\tilde{O}(T(\log D + \log H))$ bit operations

Modular black box

- Given α and m, compute $f(\alpha) \bmod m$
- Can be implemented given an arithmetic circuit / SLP
- Pure black box: evaluations on $\mathbb{Z} \setminus \{0, \pm 1\}$ have size $\Omega(D)$

General idea

- General structure: *à la* Garg-Schost
- Computing $f \bmod x^p - 1$: *à la* Prony / Ben-Or–Tiwari
- Work over several rings of different sizes to make it efficient

First ingredient: compute exponents of $f \bmod x^p - 1$

Evaluations in a small field $\mathbb{F}_q$

- If ω has order p in $\mathbb{F}_q$, $f(\omega^j) = (f \bmod x^p - 1)(\omega^j)$
- Small q for efficiency reasons
  - Only require coefficients to be nonzero mod q — $q = \mathrm{poly}(T \log H)$
- Prevent too many collisions — $p = O(T \log D)$

Algorithm

Input: a p-PRU $\omega \in \mathbb{F}_q$ — *to be computed*

1. Evaluate f at $1, \omega, \dots, \omega^{2T-1}$ — $2T$ queries
2. Compute the minimal polynomial of $(f(\omega^j))_j$ — $\tilde{O}(T \log q)$
3. Compute its roots and get the exponents by evaluation — $\tilde{O}(p \log q)$

Complexity analysis

2. $\tilde{O}(T \log(T \log H))$
3. $\tilde{O}(T \log(D) \log(T \log H))$ $\quad \to \tilde{O}(T \log D \log\log H)$

Second ingredient: compute $f \bmod x^p - 1$

Evaluations in a larger ring

- $\mathbb{F}_q$ is too small $\rightarrow$ coefficients known modulo q
- Use larger ring where coefficients can be represented
- Using large finite field is too costly (primality testing, etc.)

$\rightarrow$ Ring $\mathbb{Z}/q^k\mathbb{Z}$ where $q^k > 2H$ $\qquad k = O(\log H / \log q)$

Algorithm

Input: a p-PRU $\omega_k \in \mathbb{Z}/q^k\mathbb{Z}$ *to be computed*

1. Evaluate f at $1, \omega_k, \ldots, \omega_k^{T-1}$ $\qquad T$ queries
2. Solve a transposed Vandermonde system, build using the exponents $\qquad \tilde{O}(Tk \log q)$

Complexity analysis

2. $\tilde{O}(T \log H)$

Third ingredient: Embed exponents into coefficients

Compute both $f(x)$ and $f((1+q^k)x)$ modulo $\langle x^p - 1, q^{2k} \rangle$

Paillier-like embedding

- $(1+q^k)^{e_i} = 1 + e_i q^k \mod q^{2k}$
- If $f = \sum_i c_i x^{e_i}$,

$$f((1+q^k)x) \bmod \langle q^{2k}, x^p - 1 \rangle = \sum_i (c_i(1+e_i q^k)) x^{e_i \bmod p}$$

Collisions

- If $c_i x^{e_i}$ is collision-free modulo $x^p - 1 \to$ reconstruct both c_i and e_i
- Possibly noisy terms from collisions $e_i = e_j \bmod p$

$\to$ Compute f^* such that $(f - f^*)_\# \leq \frac{1}{2} f_\#$ w.h.p.

Fourth ingredient: p-PRU in $\mathbb{F}_q$ and $\mathbb{Z}/q^{2k}\mathbb{Z}$

Produce p, q and ω together

1. Sample a random prime $p \in [\lambda, 2\lambda]$ with $\lambda = O(T \log D)$
2. Sample a random prime $q \in \{kp + 1 : 1 \leq k \leq \lambda^5\}$ *effective Dirichlet theorem*
3. Sample a random α such that $\omega = \alpha^{(q-1)/p} \neq 1$
4. Return (p, q, ω)

Complexity: $\log^{O(1)}(\lambda) = \log^{O(1)}(T \log D)$

Lift $\omega \in \mathbb{F}_q$ to $\omega_{2k} \in \mathbb{Z}/q^{2k}\mathbb{Z}$

- If ω_{2i} is a p-PRU modulo q^{2i}, $\omega_{2i} \bmod q^i$ is a p-PRU modulo q^i
- Newton iteration to *lift* $\omega \in \mathbb{F}_q$ to $\omega_{2k} \in \mathbb{Z}/q^{2k}\mathbb{Z}$

Complexity: $\tilde{O}(k \log p \log q) = \tilde{O}(\log H \log(T \log D))$

Full algorithm

Algorithm

1. $f^* \leftarrow 0$
2. Repeat $\log T$ times :
3. Compute p, q, $\omega \in \mathbb{F}_q$, $\omega_{2k} \in \mathbb{Z}/q^{2k}\mathbb{Z}$ — Fourth ingredient
4. Compute exponents of $(f - f^*) \bmod \langle x^p - 1, q\rangle$ — First ingredient
5. Compute $(f - f^*) \bmod \langle x^p - 1, q^{2k}\rangle$ — Second ingredient
6. Compute $(f - f^*)((1 + q^k)x) \bmod \langle x^p - 1, q^{2k}\rangle$ — Second ingredient
7. Reconstruct collision-free monomials plus some noise — Third ingredient
8. Update f^*
9. Return f^*

Theorem

[Giorgi-G.-Perret du Cray-Roche (2022)]

Given a modular black box for $f \in \mathbb{Z}[x]$ and bounds T, D, H, the algorithm returns the sparse representation of f with probability $\geq \frac{2}{3}$, and has bit complexity $\tilde{O}(T(\log D + \log H))$

Getting rid of the sparsity bound

Early termination technique

- Given $(\alpha_j)_{j\geq 0}$, find its minimal polynomial without any bound on its degree
- *Berlekamp–Massey with early termination* [Kaltofen-Lee (2003)]
- Works over $\mathbb{F}_q$ with $q = \Omega(D^4)$
- Complexity: $2t$ evaluations and $\tilde{O}(t)$ operations over $\mathbb{F}_q$

And over $\mathbb{Z}$?

- Perform *early termination* modulo q, where $q = \Omega(D^4)$
- Finding such a prime is too costly $\rightarrow O(\log^3 D)$

Prime numbers without primality testing

[Giorgi-G.-Perret du Cray-Roche (2022)]

- Take a random number m and pretend it be prime
  - With good prob., its largest prime factor is $\geq \sqrt{m}$
- For each test "$a = 0 \bmod m$?" $\rightarrow$ compute $\textsc{gcd}(a, m)$ and update m
- We show that algorithms (even randomized) have the same behavior

Contents

What is the complexity of multiplying two degree-d polynomials over $\mathbb{Z}$?

Algebraic complexity over a ring

- $O(d^2)$ [folklore]
- $O(d^{1.585}), \ldots, O(d^{1+o(1)})$ [Karatsuba-Ofman (1962), Toom (1963), Cook (1966), …]
- $O(d \log d \log\log d)$ [Schönhage-Strassen (1971), Cantor-Kaltofen (1991)]

Bit complexity bounds

If $g, h \in \mathbb{Z}[x]_{\leq d}$ have height $\leq H$, gh has height $\leq dH^2$

1. Direct use of algebraic algorithms
   - Algebraic complexity $\times\, O(\log(dH) \log\log(dH))$ [Harvey-van der Hoeven (2021)]
2. Computation *modulo* a prime $p \geq 2dH^2$
   - Algebraic complexity $\times\, \tilde{O}(\log p) + \tilde{O}(\log^3 p)$ *prime generation*
3. Use Kronecker substitution ($x \mapsto 2dH^2$) and integer multiplication
   - Multiplication of integers of size $O(d \log(dH))$

Product of degree-d polynomials of height $\leq H$ in time $\tilde{O}(d \log(H))$

The case of *unbalanced* polynomials

$$
\begin{array}{rr}
 & (x^7 + 3x^6 + 213672289012x^5 - 3x^4 - 4x^3 - 7x^2 + x - 3) \\
\times & (x^7 + 3x^6 - 213672289006x^5 - 3x^4 - 4x^3 - x^2 + x - 3) \\
= & x^{14} + 6x^{13} + 15x^{12} + 12x^{11} - 45655847090345622202098x^{10} - 50x^9 - 37x^8 \\
 & +1282033734054x^7 + 28x^6 + 8x^5 + 17x^4 + 16x^3 + 25x^2 - 6x + 9
\end{array}
$$

The case of *unbalanced* polynomials

$$
\begin{array}{rr}
 & (x^7 + 3x^6 + 213672289012x^5 - 3x^4 - 4x^3 - 7x^2 + x - 3) \\
\times & (x^7 + 3x^6 - 213672289006x^5 - 3x^4 - 4x^3 - x^2 + x - 3) \\
= & x^{14} + 6x^{13} + 15x^{12} + 12x^{11} - 45655847090345622202098x^{10} - 50x^9 - 37x^8 \\
 & +1282033734054x^7 + 28x^6 + 8x^5 + 17x^4 + 16x^3 + 25x^2 - 6x + 9
\end{array}
$$

Quadratic complexity

- Let $f = \sum_{i=0}^{d} f_i x^i \to s = \textsc{bitlen}(f) \simeq \sum_i \log |f_i|$
- $H = \max |f_i| \to d + \log H \leq s \leq d \log H$
- Complexity $\tilde{O}(d \log H) = \tilde{O}(s^2)$ if $d \simeq \log H$

Can we multiply two polynomials of bit-length s in time $\tilde{O}(s)$?

Interpolation-based multiplication

The problem

Given $g, h \in \mathbb{Z}[x]$
Compute $f = g \times h$

Reinterpretation

Given an implicit representation of $f \in \mathbb{Z}[x]$ as $g \times h$
Compute the explicit (dense or sparse) representation of f

New problem

Given a way to evaluate $f \in \mathbb{Z}[x]$
Interpolate f in dense or sparse representation

Remarks

- The polynomial f can be unbalanced
- Complexity should be quasi-linear in $s = \textsc{bitlen}(f)$
- Evaluations of g and h are not for free!

Interpolation of unbalanced polynomials *first try*

Given a modular black box for $f \in \mathbb{Z}[x]$, compute f

Natural approach

1. Interpolate $f^* = f \bmod m$ for some smallish m
   - f^* contains the small coefficients of f
   - $f - f^*$ is sparser than f
2. Recursively compute $(f - f^*) \bmod m$ for increasing values of m
   - Use sparse interpolation in rings $\mathbb{Z}/m\mathbb{Z}$
   - Ring size increases while sparsity decreases

It does not work...

- At some point we know f^* of *small* height
- We need to interpolate $(f - f^*) \bmod m$ for some *large* m
- Requires to evaluate f^* on some large values $\to \tilde{O}(s^2)$

Interpolation of unbalanced polynomials *second try*

Given a modular black box for $f \in \mathbb{Z}[x]$, compute f

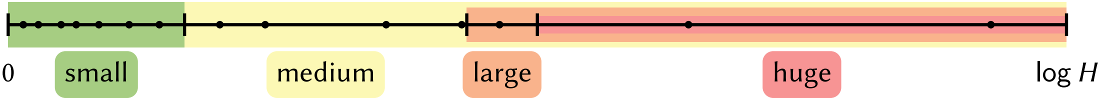
Top-down approach

1. First interpolate the *large* terms f^* of f
   - Use sparse interpolation, and *pretend* $f = f^*$
   - Smaller terms only slightly modify the evaluations
2. Recursively interpolate $f - f^*$
   - $f - f^*$ has smaller coefficients and is more balanced than f
   - Ring size decreases while sparsity increases

Main difficulties

- Deal with pertubated evaluations
- Cost of evaluations

Computing the huge terms



Technical result

- Let $f_p^{(0)} = f \bmod \langle x^p - 1, m\rangle$ and $f_p^{(1)} = x \cdot f' \bmod \langle x^p - 1, m\rangle$
- Let cx^e be a *large* term of f, $c^{(0)}$ and $c^{(1)}$ be the coefficients of $x^{e \bmod p}$ in $f_p^{(0)}$ and $f_p^{(1)}$
- If cx^e only collides with *small* terms modulo p, *and some conditions on m are satisfied,*

$$\left\lceil c^{(1)}/c^{(0)} \right\rfloor = e$$

Algorithm sketch

1. Compute a superset $\mathcal{T}$ of the large terms exponents
   - Take p so that most large terms only collide with small terms
   - Repeat with several p's for each large to be preserved at least once
2. Compute the huge terms using $\mathcal{T}$
   - Use $\mathcal{T}$ to detect collisions between large terms
   - Only keep huge coefficients: all huge terms and some large terms

Full algorithm

Algorithm

Inputs: Modular black box for $f \in \mathbb{Z}[x]$
Bounds $s \geq \textsc{bitlen}(f)$, $D \geq \deg(f)$

1. $H \leftarrow 2^s, f^* \leftarrow 0$
2. While $H \gg \log s$ and $\log D$:
3. Compute the huge terms of $f - f^*$ and update f^*
4. $H \leftarrow \sqrt{H}$
5. Compute the remaining terms of $f - f^*$ *via* (balanced) sparse interpolation

Theorem

[Giorgi-G.-Perret du Cray-Roche (2024)]

Given a modular black box for $f \in \mathbb{Z}[x]$ and bounds s and D, the algorithm returns the explicit representation of f with probability $\geq \frac{2}{3}$, and has bit complexity $\tilde{O}(s \log D)$

Remark

- Quasi-linear for *dense* or *moderately sparse* polynomials — if $\log D = \operatorname{poly}(\log s)$
- Not quasi-linear for *very sparse* polynomials — if $\log D = \operatorname{poly}(s)$

Back to polynomial multiplication

Theorem [Giorgi-G.-Perret du Cray-Roche (2024)]

There exists an algorithm that, given $g, h \in \mathbb{Z}[x]$, computes the product $f = g \times h$ with probability of success $\geq 1 - 1/s$ and expected bit complexity $\tilde{O}(s \log d)$, where $s = \textsc{bitlen}(f) + \textsc{bitlen}(g) + \textsc{bitlen}(h)$ and $d = \deg(f)$

Main ingredients

- Unbalanced interpolation with tentative bound $s \geq \textsc{bitlen}(f)$
- *Check* whether $f = g \times h$ [Giorgi-G.-Perret du Cray (2023)]
- Start with small s and double it until f is correctly computed

Remark

- Quasi-linear for *dense* or *moderately sparse* polynomials
- Not quasi-linear for *very sparse* polynomials

Summary of results

$f \in \mathbb{Z}[x]$, $D = \deg(f)$, $T = f_{\#}$, $H = f_{\infty}$, $s = \textsc{bitlen}(f)$

Sparse interpolation over the integers

- Interpolate f from a modular black box in time $\tilde{O}(T(\log D + \log H))$
- Corollaries:
  - Quasi-linear sparse multiplication algorithm [Giorgi-G.-Perret du Cray (2020)]
  - Quasi-linear exact sparse division algorithm [Giorgi-G.-Perret du Cray-Roche (2021-22)]

Unbalanced interpolation over the integers

- Interpolate f from a modular black box in time $\tilde{O}(s \log D)$
- Corollary:
  - Unbalanced polynomial multiplication in time $\tilde{O}(s \log D)$

Open problems

Quasi-linear interpolation algorithm over $\mathbb{F}_q$

- large characteristic / large field $\rightarrow$ black box? circuit?
- small field $\rightarrow$ only circuit make sense

Quasi-linear unbalanced interpolation / multiplication over $\mathbb{Z}$

- Replace $\tilde{O}(s \log D)$ by $\tilde{O}(s)$
- Remove the need for *a priori* bounds on s and D

Practical efficiency?

Open problems

Quasi-linear interpolation algorithm over $\mathbb{F}_q$

- large characteristic / large field $\to$ black box? circuit?
- small field $\to$ only circuit make sense

Quasi-linear unbalanced interpolation / multiplication over $\mathbb{Z}$

- Replace $\tilde{O}(s \log D)$ by $\tilde{O}(s)$
- Remove the need for *a priori* bounds on s and D

Practical efficiency?

Thank you!