Sparse polynomial interpolation

Bruno Grenet[1]

LJK – Université Grenoble-Alpes

ACT @ ICALP, Paderborn, July 10., 2023

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

(Vague) definition of the problem

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

where

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

Degree: $\deg(f) = \max_i e_i$
Height: $f_\infty = \max_i |c_i|$ for $c_i \in \mathbb{Z}$, q if $c_i \in \mathbb{F}_q$
Sparsity: $f_\# = t$

Many variants of the problem

Ring of coefficients

- $\mathbb{Z}$ or $\mathbb{Q}$: size growth $\to$ modular techniques
- Finite fields of *large characteristic*
- *Large* finite fields
- Small finite fields

Number of variables

- Univariate polynomials
- Multivariate polynomials

Input representation

- Evaluations
- Blackbox
- Arithmetic circuit / SLP

Outline

1. Blackbox algorithm *à la* Prony / Ben-Or–Tiwari

2. SLP algorithm *à la* Garg–Schost

3. A new quasi-linear algorithm over the integers

4. The multivariate case

Outline

1. Blackbox algorithm *à la* Prony / Ben-Or–Tiwari

2. SLP algorithm *à la* Garg–Schost

3. A new quasi-linear algorithm over the integers

4. The multivariate case

Settings

Input

- Blackbox access to $f = \sum_{i=0}^{t-1} c_i x^{e_i} \in \mathbb{F}_q$
- Bound: $T \geq t$
- Hypothesis: $q \geq \deg(f)$
- Input size: None

Output

- The sparse representation of f
- Output size: $O(t(\log q + \log \deg f))$

Complexity analysis

- Number of blackbox evaluations
- Number of operations in $\mathbb{F}_q$ or of bit operations
- Output sensitive complexity

Blahut's Theorem (1979)

Theorem

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 D-th principal root of unity. Then the minimal polynomial of $(f(\omega^j))_{j\geq 0}$ is $\Lambda(x) = \prod_{i=0}^{t-1}(x - \omega^{e_i})$.

Proof.

χ is a characteristic polynomial of $(\alpha_j)_j = (f(\omega^j))_j$

$\iff \forall j < D, \sum_{k=0}^{\ell} \chi_k \alpha_{j+k} = 0$ — *where* $\chi = \sum_k \chi_k x^k$

$\iff \overleftarrow{\chi} \times A = 0 \mod x^D - 1$ — *where* $\overleftarrow{\chi} = x^\ell \chi(\frac{1}{x})$ *and* $A = \sum_{j\leq D} \alpha_j x^j$

$\iff \forall j, \overleftarrow{\chi}(\omega^{-j}) \cdot A(\omega^{-j}) = 0$ — *by DFT on* ω^{-1}

$\iff \forall j, \chi(\omega^j) \cdot f_j = 0$ — *where* $f = \sum_{j=0}^{D-1} f_j x^j$

$\iff \forall j \in \{e_0, \ldots, e_{t-1}\}, \chi(\omega^j) = 0$ — *since* $f_j \neq 0$

$\iff \prod_{i=0}^{t-1}(x - \omega^{e_i})$ divides χ

Blahut's Theorem (1979)

Theorem

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 D-th principal root of unity. Then the minimal polynomial of $(f(\omega^j))_{j\geq 0}$ is $\Lambda(x) = \prod_{i=0}^{t-1}(x - \omega^{e_i})$.

Proof.

χ is a characteristic polynomial of $(\alpha_j)_j = (f(\omega^j))_j$

$\iff \forall j < D, \sum_{k=0}^{\ell} \chi_k \alpha_{j+k} = 0$ — *where* $\chi = \sum_k \chi_k x^k$

$\iff \overleftarrow{\chi} \times A = 0 \mod x^D - 1$ — *where* $\overleftarrow{\chi} = x^\ell \chi(\frac{1}{x})$ *and* $A = \sum_{j\leq D} \alpha_j x^j$

$\iff \forall j, \overleftarrow{\chi}(\omega^{-j}) \cdot A(\omega^{-j}) = 0$ — *by DFT on* ω^{-1}

$\iff \forall j, \chi(\omega^j) \cdot f_j = 0$ — *where* $f = \sum_{j=0}^{D-1} f_j x^j$

$\iff \forall j \in \{e_0, \ldots, e_{t-1}\}, \chi(\omega^j) = 0$ — *since* $f_j \neq 0$

$\iff \prod_{i=0}^{t-1}(x - \omega^{e_i})$ divides χ

Fast algorithm

[Berlekamp (1968), Massey (1969), ...]

- From A compute Λ as a Padé approximant $\rightarrow$ fast GCD algorithm $O(M(t) \log t)$

Sparse polynomials and transposed Vandermonde matrices

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

Corollary

- Sparse multipoint evaluation on geometric sequence
  $\iff$ transposed Vandermonde matrix-vector product
- Sparse interpolation on geometric sequence with known exponents
  $\iff$ transposed Vandermonde linear system solving

Fast algorithms

[Kaltofen-Lakshman (1992), Bostan-Lecerf-Schost (2003), …]

- Let $F = \sum_{i=0}^{t-1} c_i x^i \to (F(\omega^{e_0}), \ldots, F(\omega^{e_{t-1}}))^t = \Omega_t^t \cdot \vec{c}$
- *Transposed* dense multipoint evaluation / interpolation $\to O(M(t) \log t)$
  (*transposition principle*: problems and their transpose of same complexity)

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

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

Algorithm

Input: Blackbox for $f \in \mathbb{F}_q[x]$, $q \geq \deg(f)$; bound T on $f_\#$

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 a transposed Vandermonde system to get the coefficients $c_0, \dots, c_{t-1}$

Complexity analysis

1. $2T$ blackbox evaluations
2. $O(\mathsf{M}(T)\log T)$ *Padé approximant*
3. $O(\mathsf{M}(t)\log t \log q) + O(\sqrt{D})$ *root computation + discrete log.*
4. $O(\mathsf{M}(t)\log t)$ *transposed dense interpolation*

Remarks on Prony / Ben-Or–Tiwari algorithm

Complexity

- Quasi-linear in T, linear (optimal) number of evaluations
- Polynomial in D, rather than $\log D \rightarrow$ not polynomial in the output size

Other base rings

- Original Ben-Or–Tiwari's algorithm: over $\mathbb{Z}$
  - large evaluations $\rightarrow$ bit size $O(D)$
  - no discrete logarithm
  - originally for multivariate polynomials $\rightarrow$ factorization
- Small finite fields $\rightarrow$ use an extension
- Rings: works as long as ω is a *principal* root of unity of large order

Comparison with sparse FFT

Sparse FFT

- Given $\vec{v} \in \mathbb{C}^n$ and $k \ll n$, compute the k largest coefficients of $\mathrm{DFT}_\omega(\vec{v})$
- Complexity: $\tilde{O}(k \log n)$ floating-point operations in precision $O(n)$

[Hassanieh–Indyk–Katabi–Price (2012)]

Sparse FFT over $\mathbb{F}_q$

- No notion of coefficient size $\to$ assume $\mathrm{DFT}_\omega(\vec{v})$ has Hamming weight k
- Prony's / Ben-Or–Tiwari's algorithm computes a sparse FFT over $\mathbb{F}_q$

Lower bound

Over $\mathbb{F}_q$, sparse FFT is at least as hard as discrete logarithm

- Discrete log.: Given $\alpha, \omega \in \mathbb{F}_q$, find e such that $\alpha = \omega^e$
- Reduction to sparse FFT with $k = 1$:
  - Given α and ω, compute $\vec{v} = (1, \alpha, \alpha^2, \dots)$ and apply sparse interpolation $\to e$
- Remarks:
  - remains hard for $k > 1$ *add some known monomials*
  - both problems are polynomially equivalent

Polynomial time *incomplete* sparse interpolation

Incomplete sparse interpolation

Input: Blackbox for $f = \sum_{i=0}^{t-1} c_i x_i^e$ and bound $T \geq t$

Output: $(c_0, \ldots, c_{t-1})$ and $(\omega^{e_0}, \ldots, \omega^{e_{t-1}})$

- Same algorithm, without discrete log. computations
- Running time: $O(\mathsf{M}(t)\log(t)\log q)$ op. in $\mathbb{F}_q$

Open questions

- Incomplete sparse interpolation in *quasi-linear time*?
  - Difficulty: polynomial root finding $\rightarrow \tilde{O}(t \log q)$ op. in $\mathbb{F}_q$ *Rabin's algorithm*
- Are both problems computationally equivalent?
  - Given a polynomial p, use it to produce a linearly recurrent sequence
  - By Blahut's theorem, it is the *image* of a sparse polynomial
  - Its support gives the roots, in *log. representation*

$\rightarrow$ but computing roots from their log is not quasi-linear!

Outline

1. Blackbox algorithm *à la* Prony / Ben-Or–Tiwari

2. SLP algorithm *à la* Garg–Schost

3. A new quasi-linear algorithm over the integers

4. The multivariate case

Settings

Input

- Arithmetic circuit / SLP of size s for $f = \sum_{i=0}^{t-1} c_i x^{e_i} \in \mathbb{F}_q$
- Bounds: $T \geq t$, $D \geq \deg(f)$
- Hypothesis: $q \geq \deg(f)$

Output

- The sparse representation of f
- Output size: $O(t(\log q + \log \deg f))$

Complexity analysis

- Number of operations in $\mathbb{F}_q$ or of bit operations
- Input and output sensitive complexity

Remark

- Direct expansion of the circuit $\to$ complexity $O(D)$ *expression swell*

Use of cyclic extensions

Main idea and difficulties

[Garg-Schost (2009)]

- Compute explicitely $f \bmod x^p - 1 = \sum_i c_i x^{e_i \bmod p}$ for some prime p
- 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 *à la* Paillier or using derivatives

  [Arnold-Roche (2015), Huang (2019)]

Deal with collisions

- Large enough prime and/or many primes to avoid any collision [Garg-Schost (2009)]
- Accept few collisions and reconstruct f iteratively

  [Arnold-Giesbrecht-Roche (2013), Huang (2019)]

Embedding exponents into coefficients

Using derivatives

[Huang 2019]

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

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 mod p: pair (e_i, e_j) such that $e_i \equiv e_j \mod p$

Avoiding or limiting collisions

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

- For $\lambda = O(\frac{1}{\varepsilon} T^2 \log D)$, there is no collision with prob. $\geq 1 - \varepsilon$
- For $\lambda = O(\frac{1}{\varepsilon} T \log D)$, 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 *create* one fake monomial
- 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

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

1. $f^* \leftarrow 0$
2. Repeat $\log(T)$ times:
3. Take a random $p \in [\lambda, 2\lambda]$ for $\lambda = O(T \log D \log T)$
4. Compute $f \bmod x^p - 1$ and $f' \bmod x^p - 1$ using dense arithmetic *(circuit for f')*
5. For each pair of monomials $cx^d \in f \bmod x^p - 1$ and $c'x^{d-1} \in f' \bmod x^p - 1$:
6. if $c'/c \in \{0, \ldots, D-1\}$: add $c \cdot x^{c'/c}$ to f^*
7. Return f^*

Complexity analysis

- $O(\log T)$ *probes* of the circuit $\rightarrow O(s \cdot \mathsf{M}(p) \cdot \log(T))$
- $p = O(T \log D \log T)$

$\rightarrow \tilde{O}(sT \log D)$ operations in $\mathbb{F}_q$

$\tilde{O}(sT \log D \log q)$ binary 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)]

Outline

1. Blackbox algorithm *à la* Prony / Ben-Or–Tiwari

2. SLP algorithm *à la* Garg–Schost

3. A new quasi-linear algorithm over the integers

4. The multivariate case

Known results for sparse interpolation over $\mathbb{Z}$

$f = \sum_{i=0}^{t-1} c_i x^{e_i}$, $T \geq t$, $D \geq \deg(f)$, $H \geq f_\infty$

Already mentioned

- Blackbox interpolation: $\tilde{O}(\sqrt{D})$ — *Prony / Ben-Or–Tiwari*
- Arithmetic circuit: $\tilde{O}(sT \log^3 D \log H)$ — *Garg–Schost*

Mansour's algorithm

[Mansour (1995)]

- Input: blackbox over $\mathbb{C}$, or $(f(\omega^j))_{j\geq 0}$ where $\omega = e^{2i\pi/N}$
- Main idea:
  - Binary search of nonzero coefficients: define $f_{\alpha,\ell} = \sum_{i:e_i \equiv \alpha \bmod 2^\ell} c_i x^{e_i}$
  - Fast approximate computation of $\|f_{\alpha,\ell}\|_2^2$ using evaluations on random ω^j
- Complexity: polynomial in T, $\log D$, $\log H$
  - First polynomial-time sparse interpolation algorithm
  - Can be derandomized — [Alon-Mansour (1995)]
- Sparse FFT can be seen as an improvement of Mansour's algorithm
  - Bit complexity $\tilde{O}(T \log^2 D)$

The new algorithm

Input: A *modular blackbox* 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 blackbox

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

General idea

- Follow Garg–Schost general structure
- Compute $f \bmod x^p - 1$ *à la* Prony / Ben-Or–Tiwari
- Work over several rings to make it efficient

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

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

- If ω is a p-PRU in $\mathbb{F}_q$, $f(\omega^j) = (f \bmod x^p - 1)(\omega^j)$
- Small q for efficiency reasons
- Coefficients should remain nonzero modulo $q \to q = \text{poly}(T \log H)$

Algorithm

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

1. Evaluate f at $1, \omega, \ldots, \omega^{2T-1}$ — $2T$ 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 multipoint evaluation — $\tilde{O}(p \log q)$

Complexity

- $p = O(T \log D)$ as in Garg–Schost's algorithm

$$\to \tilde{O}(T \log D \log q) = \tilde{O}(T \log D \log \log H)$$

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

Evaluations in a larger ring

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

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

$\to$ Complexity: $\tilde{O}(T \log H)$

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 $\omega_k \in \mathbb{Z}/q^{2k}\mathbb{Z}$

- If ω_{2i} is a p-PRU modulo q^{2i}, $\omega_{2i} \bmod q^i$ is a p-PRU modulo q^i
- Write $\omega_{2i} = \omega_i + aq^i$:
  - $1 \equiv \omega_{2i}^p \equiv \omega_i^p + p\omega_i^{p-1}aq^i \bmod q^{2i} \Rightarrow 1 - \omega_i^p \equiv q^i \times ap\omega_i^{-1} \bmod q^{2i}$
  - $a = \left[\frac{1}{q^i}(1 - \omega_i^p \bmod q^{2i})\right] \times (\omega_i p^{-1}) \bmod q^i$
- Complexity: $\tilde{O}(k \log p \log q) = \tilde{O}(\log H \log(T \log D))$ binary operations

Complete algorithm

Algorithm

1. $f^* \leftarrow 0$
2. Repeat $\log T$ times :
3. Compute $p, q, \omega \in \mathbb{F}_q, \omega_k \in \mathbb{Z}/q^{2k}\mathbb{Z}$ — 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 blackbox 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

Outline

1. Blackbox algorithm *à la* Prony / Ben-Or–Tiwari

2. SLP algorithm *à la* Garg–Schost

3. A new quasi-linear algorithm over the integers

4. The multivariate case

Kronecker substitution

The substitution

[Kronecker (1882?)]

$f \in R[x_0, \ldots, x_{n-1}]$ with $\deg_{x_i}(f) < D \mapsto f_u(x) = f(x, x^D, x^{D^2}, \ldots, x^{D^{n-1}})$

- $\deg(f_u) < D^n$
- Easily computable and invertible
- Replaces $\log(D)$ with $n \log(D)$ in the complexities
- Generalization if $\deg_{x_i}(f) < d_i$: $f_u(x) = f(x, x^{d_0}, x^{d_0 d_1}, \ldots, x^{d_0 \cdots d_{n-2}})$

Caveats

- Over $\mathbb{F}_q$ where q must be $\geq D$: the condition becomes $q \geq D^n$ huge!
- Replace an evaluation point α by $(\alpha, \alpha^D, \ldots, \alpha^{D^{n-1}})$
  - n times more bits than α
  - a call to the (multivariate) blackbox is more expensive than to a univariate blackbox

Randomized Kronecker substitution

The substitution

[Arnold-Roche (2014)]

$f \in R[x_0, \ldots, x_{n-1}]$ with $\deg_{x_i}(f) < D \mapsto f_u(x) = f(x^{s_0}, \ldots, x^{s_{n-1}})$

- with random $s_0, \ldots, s_{n-1} = \tilde{O}(Tn \log D)$
- $\deg(f_u) = \tilde{O}(TnD)$
- possible collisions $\rightarrow$ non invertible
- use several random tuples $(s_0, \ldots, s_{n-1})$

Results

Sparse interpolation of $f \in \mathbb{F}_{q^s}[x_0, \ldots, x_{n-1}]$ in time

- $\tilde{O}(snT \log D \log q^s)$ if $q = \tilde{\Omega}(nDT)$ [Huang (2019)]
- $\tilde{O}(snt \log^2 D(\log D + \log q^s))$ otherwise [Huang–Gao (2020]

Conclusion

Results

Sparse interpolation over the integers

- First quasi-linear algorithm for modular blackbox
  - Complexity $\tilde{O}(sT(\log D + \log H))$ for arithmetic circuit of size s
- Corollaries:
  - First quasi-linear sparse multiplication algorithm [Giorgi-G.-Perret du Cray (2020)]
  - First quasi-linear exact sparse division algorithm [Giorgi-G.-Perret du Cray-Roche (2021-22)]

Sparse interpolation over $\mathbb{F}_q$, $char(q) \geq D$

- Huang's algorithm for arithmetic circuits: $\tilde{O}(sT\log(D)\log(q))$
- *À la* Prony / Ben-Or–Tiwari (extended blackbox): $\tilde{O}(T\log^2(q))$ [G. (unpublished)]
  - Incomplete sparse interpolation + exponent embedding

Many other results

- Derandomization [Klivans-Spielmann (2001), Bläser-Jindal (2014), …]
- Other fields [Kaltofen-Lakshman-Wiley (1990), Avendaño-Krick-Pacetti (2006), …]
- Parallel algorithms [Grigoriev-Karpinski-Singer (1990), Javadi-Monagan (2010), …]
- Very fast heuristic algorithms [van der Hoeven-Lecerf (2014, 2019, 2021, …)]

Open problems

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

- large characteristic / large field $\rightarrow$ blackbox? circuit?
- small field $\rightarrow$ only circuit make sense
- over field of large characteristic: computational equivalence with root finding?

Truly quasi-linear algorithm for circuit interpolation

- input size is $s \log H$ where H bounds the constants
- algorithms in $\tilde{O}(sT(\log D + \log H))$
- Easier problem: given a circuit C and a sparse polynomial f, does C compute f?
  - (Deterministic) polynomial time algorithm [Bläser-Hardt-Lipton-Vishnoi (2009)]
  - Randomized: $O(sT \log(DH) + T \log(D) \log(DH))$ [Giorgi-G.-Perret du Cray-Roche (2022)]

Many open problems on sparse polynomials

- GCD, Euclidean division, divisibility testing, factorization, …

Open problems

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

- large characteristic / large field $\rightarrow$ blackbox? circuit?
- small field $\rightarrow$ only circuit make sense
- over field of large characteristic: computational equivalence with root finding?

Truly quasi-linear algorithm for circuit interpolation

- input size is $s \log H$ where H bounds the constants
- algorithms in $\tilde{O}(sT(\log D + \log H))$
- Easier problem: given a circuit C and a sparse polynomial f, does C compute f?
  - (Deterministic) polynomial time algorithm [Bläser-Hardt-Lipton-Vishnoi (2009)]
  - Randomized: $O(sT\log(DH) + T\log(D)\log(DH))$ [Giorgi-G.-Perret du Cray-Roche (2022)]

Many open problems on sparse polynomials

- GCD, Euclidean division, divisibility testing, factorization, …

Thank you!