Modular Tricks for Integer Sparse Polynomials

Dan Roche

Computer Science Department
United States Naval Academy

RTCA'23 Workshop: Fundamental Algorithms and Algorithmic Complexity
Institut Henri Poincaré, Paris
29 Sept 2023

Thank you!

Questions?

My Collaborators

Bruno Grenet
Université Grenoble Alpes

Pascal Giorgi
University of Montpellier

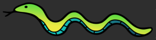
Armelle Perret du Cray
University of Waterloo

Plan

1. What are we doing?
2. How do we do it?
3. The modular tricks
4. Why did we do it?

Integer Sparse Polynomials

Setting

- Univariate polynomials in $\mathbb{Z}[X]$
- Multi-precision coefficients () AND exponents ()
- Necessarily "supersparse" / "lacunary"

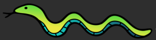
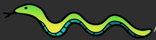
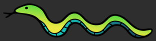
Example

$$f(X) = 123\,X^{987} + 346\,X^{765} + 567\,X^{543} + 789\,X^{321}$$

- $T = 4$ nonzero terms
- All coefficients and exponents at most $B = 1000$

Sparse Interpolation

Input

- Unknown $f(X) =$ $_1 X$$_1 + \cdots +$ $_T X$$_T$
- Way to evaluate at chosen points
- Bound T on sparsity (number of nonzero terms)
- Bound B on largest  or 

Output

- List of coefficient/exponent pairs $\in (\mathbb{Z} \times \mathbb{Z})^T$
- Total bit-length $O(T \log B)$

Optimal number of evaluations?

Quiz

Say f has $T \leq 4$ nonzero terms and coeffs, expons $\leq B = 1000$.
What is the fewest number of evaluations to recover f?

Optimal number of evaluations?

Quiz

Say f has $T \leq 4$ nonzero terms and coeffs, expons $\leq B = 1000$.
What is the fewest number of evaluations to recover f?

Answer

Just one evaluation over $\mathbb{Z}$ is enough!

Suppose $f(X) = 123\,X^{987} + 346\,X^{765} + 567\,X^{543} + 789\,X^{321}$

$f(1000) = 12300000 \cdots 0000034600000 \cdots 0000056700000 \cdots 0000078900000 \cdots 000$

But this has bit-length $O(TB)$, exponentially larger than output size $O(T \log B)$

Cost model

Evaluating f: "Modular Black Box"

Input: Modulus $m \in \mathbb{Z}$

Input: Point $\theta \in \mathbb{Z}/m\mathbb{Z}$

Output: $f(\theta) \bmod m$

Cost: evaluation bit-length $\log m$

Goal

- Total evaluation bit-length $O(T \log B)$
- Total computation bit-cost $\widetilde{O}(T \log B)$

where $\widetilde{O}(\blacksquare)$ is defined as $O(\blacksquare \cdot \text{polylog}(T + \log B))$

Plan

- What are we doing?
- How do we do it?
- The modular tricks
- Why did we do it?

Recovering tiny exponents from f_{tiny}

- Write $f_{\text{tiny}}(X) = \text{🐍}_1 X^{\text{🐘}_1}$
- Assume all $\text{🐍}_i \in \text{poly}(T + \log B)$ and all $\text{🐘}_i \in O(T \log B)$
- Let q a prime and $\omega \in \mathbb{F}_q$ where $q \in \text{poly}(T + \log B)$ and $\text{ord}_q(\omega) > \max \text{🐘}_i$

Recovering tiny exponents from f_{tiny}

- Write $f_{\text{tiny}}(X) = \text{🐍}_1 X^{\text{🐘}_1}$
- Assume all $\text{🐍}_i \in \text{poly}(T + \log B)$ and all $\text{🐘}_i \in O(T \log B)$
- Let q a prime and $\omega \in \mathbb{F}_q$ where $q \in \text{poly}(T + \log B)$ and $\text{ord}_q(\omega) > \max \text{🐘}_i$

Algorithm: Tiny Exponent Recovery

1. Evaluate $f_{\text{tiny}}(1), f_{\text{tiny}}(\omega), \cdots, f_{\text{tiny}}(\omega^{2T-1})$ modulo q
2. Fast Berlekamp-Massey to recover $\Lambda(Z) = \prod_i (Z - \omega^{\text{🐘}_i})$
3. Evaluate $\Lambda(1), \Lambda(\omega), \ldots, \Lambda(\omega^{\widetilde{O}(T \log B)})$
4. Roots of Λ reveal values of 🐘_i's

Recovering tiny exponents from f_{tiny}

- Write $f_{\text{tiny}}(X) =$  $_1 X$  $_1$
- Assume all  $_i \in$ poly$(T + \log B)$ and all  $_i \in O(T \log B)$
- Let q a prime and $\omega \in \mathbb{F}_q$ where $q \in$ poly$(T + \log B)$ and $\mathrm{ord}_q(\omega) > \max$  $_i$

Algorithm: Tiny Exponent Recovery

1. Evaluate $f_{\text{tiny}}(1), f_{\text{tiny}}(\omega), \cdots, f_{\text{tiny}}(\omega^{2T-1})$ modulo q
2. Fast Berlekamp-Massey to recover $\Lambda(Z) = \prod_i (Z - \omega$  $_i)$
3. Evaluate $\Lambda(1), \Lambda(\omega), \ldots, \Lambda(\omega^{\widetilde{O}(T \log B)})$
4. Roots of Λ reveal values of  $_i$'s

COST: $O(T \log B))$ evaluation bits and $\widetilde{O}(T \log B)$ computation

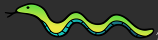
Recovering big coefficients from f_{wide}

- Write $f_{\text{wide}}(X) =$ $_1 X^{}$$^{1} + \cdots +$ $_T X^{}$^{T}
- Assume all $_i \leq B$ and all $_i \in O(T \log B)$

Recovering big coefficients from f_{wide}

- Write $f_{\text{wide}}(X) =$ $_1 X^{}$$^{}_1 + \cdots +$ $_T X^{}$$^{}_T$
- Assume all $_i \leq B$ and all $_i \in O(T \log B)$

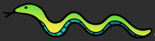
Algorithm: Big Coefficient Recovery

1. Choose small prime $q \in \text{poly}(T + \log B)$ and larger modulus $m \geq B$
2. Use Tiny Exponent Recovery mod q to obtain all $_i$'s
3. Evaluate $f_{\text{wide}}(1), f_{\text{wide}}(\omega), \ldots, f_{\text{wide}}(\omega^{T-1})$ mod m
4. Solve Transposed Vandermonde system (**next slide) to recover $_i$'s

Recovering big coefficients from f_{wide}

- Write $f_{\text{wide}}(X) =$  $_1 X$  $_1$ $+ \cdots +$  $_T X$  $_T$
- Assume all  $_i \leq B$ and all  $_i \in O(T \log B)$

Algorithm: Big Coefficient Recovery

1. Choose small prime $q \in \text{poly}(T + \log B)$ and larger modulus $m \geq B$
2. Use Tiny Exponent Recovery mod q to obtain all  $_i$'s
3. Evaluate $f_{\text{wide}}(1), f_{\text{wide}}(\omega), \ldots, f_{\text{wide}}(\omega^{T-1})$ mod m
4. Solve Transposed Vandermonde system (**next slide) to recover  $_i$'s

COST: $O(T \log B))$ evaluation bits and $\widetilde{O}(T \log B)$ computation

Aside: Transposed Vandermonde

Recall:

- $f_{\text{wide}}(X) = \text{🐍}_1 X^{\text{🐘}_1} + \cdots + \text{🐍}_T X^{\text{🐘}_T}$
- We know 🐘_i's and have $f_{\text{wide}}(\omega^i)$'s

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \omega^{\text{🐘}_1} & \omega^{\text{🐘}_2} & \cdots & \omega^{\text{🐘}_T} \\ \omega^{2\text{🐘}_1} & \omega^{2\text{🐘}_2} & \cdots & \omega^{2\text{🐘}_T} \\ \vdots & \vdots & \vdots & \vdots \\ \omega^{(T-1)\text{🐘}_1} & \omega^{(T-1)\text{🐘}_2} & \cdots & \omega^{(T-1)\text{🐘}_T} \end{bmatrix} \begin{bmatrix} \text{🐍}_1 \\ \text{🐍}_2 \\ \vdots \\ \text{🐍}_T \end{bmatrix} = \begin{bmatrix} f_{\text{wide}}(1) \\ f_{\text{wide}}(\omega) \\ f_{\text{wide}}(\omega^2) \\ \vdots \\ f_{\text{wide}}(\omega^{T-1}) \end{bmatrix}$$

Can solve using $\widetilde{O}(T)$ field operations using fast polynomial arithmetic

Recovering big exponents from f

- Write $f(X) =$  $_1 X$  $^{1} + \cdots +$  $_T X$  ^{T}
- Assume all  $_i,$  $_i \le B$

Recovering big exponents from f

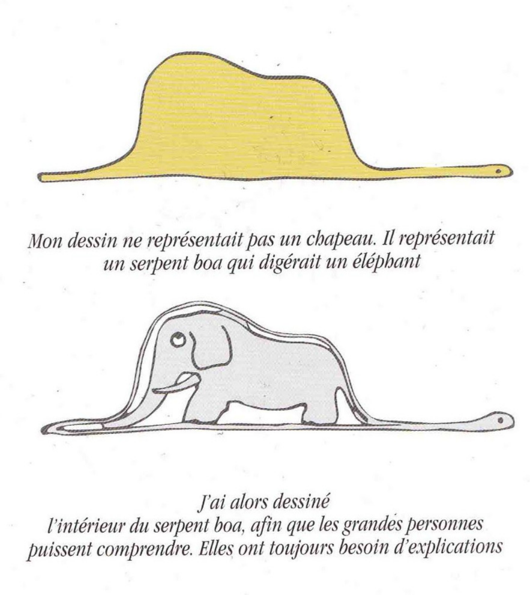
- Write $f(X) =$  $_1 X$  $^{1} + \cdots +$  $_T X$  ^{T}
- Assume all  $_i$,  $_i \leq B$

Algorithm: Big Exponent Recovery

1. Create implicit polynomial g with tiny exponents and full exponents embedded in the coefficients
2. Use Big Coefficient Recovery to obtain coefficients of g
3. Extract actual 's and 's from coefficients of g

Plan

- What are we doing?
- How do we do it?
- The modular tricks
- Why did we do it?



What is needed

1. Make exponents tiny: turn  $\mapsto$ 
2. Embed exponents in coefficients:  $+$  $\mapsto$ 
3. Make small q and $\omega \in \mathbb{F}_q$ "compatible with" large m and $\omega \in \mathbb{Z}/m\mathbb{Z}$

Trick 1: Linnick

 $\mapsto$ 

1. Choose tiny prime $p \in O(T \log B)$
2. Try to find a prime $q = pr + 1$

How large should we try for r?

How long will this take?

The dirty work

Sedunova 2018, Corollary 1.5

Let $\pi(x)$ denote the number of prime numbers $\leq x$, $\pi(x; m, a)$ the number of prime numbers $\leq x$ that are congruent to a modulo m, and $\ell(x)$ the smaller prime divisor of x. Then for any $\gamma \geq 4$ and $\lambda_1 \leq \lambda_2 \leq \gamma^{1/2}$,

$$\sum_{\substack{m \leq \lambda_2 \\ \ell(m) > \lambda_1}} \max_{2 \leq y \leq \gamma} \max_{a : \gcd(a,m)=1} \left| \pi(y; m, a) - \frac{\pi(y)}{\phi(m)} \right|$$
$$\leq 122.77 \left(14 \frac{\gamma}{\lambda_1} + 4\gamma^{1/2}\lambda_2 + 15\gamma^{2/3}\lambda_2^{1/2} + 4\gamma^{5/6} \ln(\frac{\lambda_2}{\lambda_1}) \right) (\ln \gamma)^{7/2}.$$

Our version

Giorgi, Grenet, Perret du Cray, R 2022

Given a bit-size $b \geq 60$, in worst-case poly(b) time, we can find a triple (p, q, ω) where w.h.p.

- p is a b-bit prime
- q is a prime with at most $6b$ bits
- $p \mid (q-1)$
- ω is a p-PRU in $\mathbb{F}_q$

Trick 2: Paillier

How to implicitly embed exponents in coefficients?

- Assume we can evaluate $\frac{d}{dX} f(X)$

  OR

- Use the fact that $(1+m)^e \bmod m^2 = 1 + em$

One-step coefficients and exponents embedding

Recall $f(X) =$  $_1 X$  $^{1} + \cdots +$  $_T X$  ^{T}

Fact

Let $g(X) = X \cdot f(X + m) + (1 - X) \cdot f(X) \mod m^2$.

Then the coefficient of X^i in g is  $_i \cdot (1 +$  $_i \cdot m) \mod m^2$

One-step coefficients and exponents embedding

Recall $f(X) =$  $_1 X$  $^{1} + \cdots +$  $_T X$  ^{T}

Fact

Let $g(X) = X \cdot f(X+m) + (1-X) \cdot f(X) \mod m^2$.

Then the coefficient of X^i in g is  $_i \cdot (1 +$  $_i \cdot m) \mod m^2$

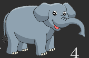
We can recover both  $_i$ and  $_i$ if m is large enough

Trick 3: Newton

Find large m, ω_m "consistent with" small q and ω

- Use $m = q^k$ for $k \geq \log_q B$
- Each p-PRU in $\mathbb{Z}/q^k\mathbb{Z}$ is 1-1 with the p-PRUs in $\mathbb{Z}/q^i\mathbb{Z}$ for $1 \leq i \leq k$
- We construct a Newton iteration to lift $\omega_m \mod q^k$ from $\omega \mod q$ in $O(\log k)$ steps.

Example: Setup

$f(X) =$  $_1 X$  $^1 +$  $_2 X$  $^2 + \cdots +$  $_4 X$  ^{4}

Bounds: $T = 4$, $B = 1000$

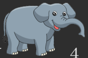
Example: Recover tiny exponents

$f(X) =$ 🐍$_1 X^{🐘_1} +$ 🐍$_2 X^{🐘_2} + \cdots +$ 🐍$_4 X^{🐘_4}$

Bounds: $T = 4,\ B = 1000$

1. Choose tiny $p = 11$, small $q = 23$, p-PRU $\omega = 6$
2. $2T$ evals $f(\omega_i) \mod q = 8, 22, 3, 5, 17, 11, 11, 8$
3. Berlekamp-Massey to find $\Lambda(Z) = x^4 + 18x^3 + 7x^2 + 3x + 16$
4. Multi-point evals to find roots: $\Lambda(\omega^2) = \Lambda(\omega^4) = \Lambda(\omega^6) = \Lambda(\omega^8) = 0$
5. Therefore 🐘's are $[2, 4, 6, 8]$

Example: Recover big coefficients

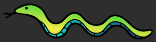
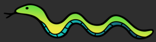
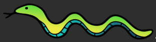
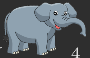
$f(X) =$  $_1 X$  $^1 +$  $_2 X$  $^2 + \cdots +$  $_4 X$  ^{4}

Bounds: $T = 4,\ B = 1000$

Recall: $p = 11$, small $q = 23$, p-PRU $\omega = 6$, and 's are $[2, 4, 6, 8]$

6. Set $m = q^3 = 12167$ and lift ω to p-PRU $\omega_m = 90603883 \bmod m^2$
7. Define $g(X) = X \cdot f(X + m) + (1 - X) \cdot f(X) \mod m^2$
8. T evals $g(\omega_m^i) \bmod m^2 = 126319619, 41994848, 22280517, 104183726$
9. Solve system to get s $= 120806932, 45091469, 111729907, 144763089$

Example: Recover f

$f(X) =$  $_1 X$  $^1 + $  $_2 X$  $^2 + \cdots +$  $_4 X$  ^{4}

Bounds: $T = 4$, $B = 1000$

Recall: $p = 11$, small $q = 23$, $m = 12167$,
and  s are $120806932, 45091469, 111729907, 144763089$

10 Unpack each  as in:

- $120806932 = 9929m + 789$
-  $_1 = 789$
-  $_1 = (9929/789) \bmod m = 321$

11 $f = 789X^{321} + 567X^{543} + 346X^{765} + 123X^{987}$

Plan

- What are we doing?
- How do we do it?
- The modular tricks
- Why did we do it?

What we have

Given a modular black box for an unknown polynomial $f \in \mathbb{Z}[x]$,
and bounds on f's sparsity and (uniform) term bit-length,
we can recover f in time proportional to the worst-case output size.

Easily extends to multivariate polynomials and rational coefficients

What we don't (yet) have

- Fast sparse interpolation using only low-precision evaluations
- Sensitivity to average term bit-length
- (Super)sparse rational function recovery
- Softly-optimal sparse interpolation over finite fields
- Numerical stability with (soft)-optimal complexity

Applications and Connections

- Sparse polynomial multiplication
- More generally: Avoid intermediate expression swell
- Related to Reed-Solomon decoding, exponential analysis, Hermite-Pade, ...

A brief history

- Ben-Or & Tiwari 1988
- Zippel 1990
- Kaltofen & Lakshman 1988
- Kaltofen, Lakshman, Wiley 1990
- Grigoriev, Karpinski, & Singer 1990
- Mansour 1995
- Huang & Rao 1996
- Murao & Fujise 1996
- Kaltofen & Lee 2003
- Avendaño, Krick, & Pacetti 2006
- Garg & Schost 2009
- Kaltofen 2010
- Javadi & Monagan 2010
- Giesbrecht & R 2011
- Cuyt & Lee 2011
- Arnold & R 2014
- van der Hoeven & Lecerf 2015
- Arnold, Giesbrecht & R 2016
- Huang & Gao 2017
- van der Hoven & Lecerf 2019
- Huang 2020
- Giorgi, Grenet, Perret du Cray, & R 2022

Merci encore !