Sparse Polynomial Interpolation

Michael Monagan

Department of Mathematics, Simon Fraser University
This is a joint work with Mahdi Javadi

JonFest
May 16-20, 2011
Canucks 7 Sharks 3

The CECM and my connection with JB

SFU computer algebra group photo – circa 2003.

The Black Box Model

Let $f \in K[x_1, x_2, ..., x_n]$.
Model: the target polynomial f is represented with a black box **B**.

- $f = \det A \in K[x_1, x_2, \ldots, x_n]$
- [Zippel, 1979] **B** computes

$$\gcd(f_1(x_1, \alpha_2, \ldots, \alpha_n), f_2(x_1, \alpha_2, \ldots, \alpha_n)) \in \mathbb{Z}_p[x_1]$$

Problem: how many "probes of **B**" do we need to interpolate f?
Assuming we know $d \geq \deg f$?

The Black Box Model

Let $f \in K[x_1, x_2, ..., x_n]$.
Model: the target polynomial f is represented with a black box **B**.

- $f = \det A \in K[x_1, x_2, \ldots, x_n]$
- [Zippel, 1979] **B** computes

$$\gcd(f_1(x_1, \alpha_2, \ldots, \alpha_n), f_2(x_1, \alpha_2, \ldots, \alpha_n)) \in \mathbb{Z}_p[x_1]$$

Problem: how many "probes of **B**" do we need to interpolate f?
Assuming we know $d \geq \deg f$?

Sparse Polynomial Interpolation

Let $f \in K[x_1, x_2, ..., x_n]$ have degree d and t non-zero terms.

If f is dense then we need $O((1+d)^n)$ points in K^n.

If f is sparse ($t \ll \binom{d+n}{n}$) e.g.

$$f = x_1^d + 2x_2^d + ... + nx_n^d$$

can we interpolate f using $O(dt)$ points? $O(t)$?

Zippel [1979] $(K = \mathbb{Z}_p)$ – $O(ndt)$ points.
Ben-Or and Tiwari [1988] $(char(K) = 0)$ – $O(t)$ points.
Javadi and MM [2010] $(K = \mathbb{Z}_p)$ – $O(nt)$ points.

Sparse Polynomial Interpolation

Let $f \in K[x_1, x_2, ..., x_n]$ have degree d and t non-zero terms.

If f is dense then we need $O((1+d)^n)$ points in K^n.

If f is sparse ($t \ll \binom{d+n}{n}$) e.g.

$$f = x_1^d + 2x_2^d + ... + nx_n^d$$

can we interpolate f using $O(dt)$ points? $O(t)$?

Zippel [1979] $(K = \mathbb{Z}_p)$ – $O(ndt)$ points.
Ben-Or and Tiwari [1988] $(char(K) = 0)$ – $O(t)$ points.
Javadi and MM [2010] $(K = \mathbb{Z}_p)$ – $O(nt)$ points.

The Ben-Or/Tiwari Algorithm for $\mathbb{Z}$

Let $f = \sum_{i=1}^{t} c_i M_i$ where $c_i \in \mathbb{Z}$ and $M_i = x_1^{d_{i1}} \cdot x_2^{d_{i2}} \cdots x_n^{d_{in}}$.

Let $T \geq t$ be a bound on the number of non-zero terms in f.

Step 1 For $i = 0 \ldots 2T-1$ compute $v_i = \mathbf{B}(2^i, 3^i, 5^i, \ldots, p_n^i)$.

Step 2 Input the v_i into the Berlekamp/Massey algorithm to compute $\Lambda(z) \in \mathbb{Z}[z]$ the linear generator for the sequence $v_0, v_1, \ldots, v_{2T-1}$.

Theorem: $\Lambda(z) = \prod_{i=1}^{t} (z - \underbrace{M_i(2, 3, 5, \ldots, p_n)}_{m_i})$.

Step 3 Compute $m_1, \ldots, m_t$ the integer roots of $\Lambda(z)$.

Step 4 Determine the degrees of each monomial by trial division in $\mathbb{Z}$.

Step 5 Solve for the coefficients c_i.

The integers $\mathbf{B}(2^i, 3^i, 5^i, \ldots)$ have size $O(dT \log n)$ bits !!

The Ben-Or/Tiwari Algorithm for $\mathbb{Z}$

Let $f = \sum_{i=1}^{t} c_i M_i$ where $c_i \in \mathbb{Z}$ and $M_i = x_1^{d_{i1}} \cdot x_2^{d_{i2}} \cdots x_n^{d_{in}}$.

Let $T \geq t$ be a bound on the number of non-zero terms in f.

Step 1 For $i = 0 \ldots 2T-1$ compute $v_i = \mathbf{B}(2^i, 3^i, 5^i, \ldots, p_n^i)$.

Step 2 Input the v_i into the Berlekamp/Massey algorithm to compute $\Lambda(z) \in \mathbb{Z}[z]$ the linear generator for the sequence $v_0, v_1, \ldots, v_{2T-1}$.

Theorem: $\Lambda(z) = \prod_{i=1}^{t} (z - \underbrace{M_i(2, 3, 5, \ldots, p_n)}_{m_i})$.

Step 3 Compute $m_1, \ldots, m_t$ the integer roots of $\Lambda(z)$.

Step 4 Determine the degrees of each monomial by trial division in $\mathbb{Z}$.

Step 5 Solve for the coefficients c_i.

The integers $\mathbf{B}(2^i, 3^i, 5^i, \ldots)$ have size $O(dT \log n)$ bits !!

The Ben-Or/Tiwari Algorithm for $\mathbb{Z}$

Let $f = \sum_{i=1}^{t} c_i M_i$ where $c_i \in \mathbb{Z}$ and $M_i = x_1^{d_{i1}} \cdot x_2^{d_{i2}} \cdots x_n^{d_{in}}$.

Let $T \geq t$ be a bound on the number of non-zero terms in f.

Step 1 For $i = 0 \ldots 2T-1$ compute $v_i = \mathbf{B}(2^i, 3^i, 5^i, \ldots, p_n^i)$.

Step 2 Input the v_i into the Berlekamp/Massey algorithm to compute $\Lambda(z) \in \mathbb{Z}[z]$ the linear generator for the sequence $v_0, v_1, \ldots, v_{2T-1}$.

Theorem: $\Lambda(z) = \prod_{i=1}^{t} (z - \underbrace{M_i(2, 3, 5, \ldots, p_n)}_{m_i})$.

Step 3 Compute $m_1, \ldots, m_t$ the integer roots of $\Lambda(z)$.

Step 4 Determine the degrees of each monomial by trial division in $\mathbb{Z}$.

Step 5 Solve for the coefficients c_i.

The integers $\mathbf{B}(2^i, 3^i, 5^i, \ldots)$ have size $O(dT \log n)$ bits !!

The Ben-Or/Tiwari Algorithm for $\mathbb{Z}$

Let $f = \sum_{i=1}^{t} c_i M_i$ where $c_i \in \mathbb{Z}$ and $M_i = x_1^{d_{i1}} \cdot x_2^{d_{i2}} \cdots x_n^{d_{in}}$.

Let $T \geq t$ be a bound on the number of non-zero terms in f.

Step 1 For $i = 0 \ldots 2T-1$ compute $v_i = \mathbf{B}(2^i, 3^i, 5^i, \ldots, p_n^i)$.

Step 2 Input the v_i into the Berlekamp/Massey algorithm to compute $\Lambda(z) \in \mathbb{Z}[z]$ the linear generator for the sequence $v_0, v_1, \ldots, v_{2T-1}$.

Theorem: $\Lambda(z) = \prod_{i=1}^{t} (z - \underbrace{M_i(2, 3, 5, \ldots, p_n)}_{m_i})$.

Step 3 Compute $m_1, \ldots, m_t$ the integer roots of $\Lambda(z)$.

Step 4 Determine the degrees of each monomial by trial division in $\mathbb{Z}$.

Step 5 Solve for the coefficients c_i.

The integers $\mathbf{B}(2^i, 3^i, 5^i, \ldots)$ have size $O(dT \log n)$ bits !!

Ben-Or/Tiwari Algorithm (contd.)

$$\Lambda(z) = \prod_{i=1}^{t} (z - M_i(2, 3, 5, \ldots, p_n)).$$

Do it modulo a prime p ! ($p > M_i(2, 3, 5, ..., p_n) < p_n^d$)

Huang and Rao [1990]
Replace $2, 3, 5, \ldots$ by irreducibles $y - a_1, y - a_2, \ldots$ in $GF(p)[y]$.

Compute $\mathbf{B}((y - a_1)^i, \ldots, (y - a_n)^i)$.
How?
By evaluation and interpolation of y !
Needs $p > 8d^2t^2$.
Does $O(ndt^2)$ probes!

Ben-Or/Tiwari Algorithm (contd.)

$$\Lambda(z) = \prod_{i=1}^{t} (z - M_i(2, 3, 5, \ldots, p_n)).$$

Do it modulo a prime p ! ($p > M_i(2, 3, 5, ..., p_n) < p_n^d$)

Huang and Rao [1990]
Replace $2, 3, 5, \ldots$ by irreducibles $y - a_1, y - a_2, \ldots$ in $GF(p)[y]$.

Compute $\mathbf{B}((y - a_1)^i, \ldots, (y - a_n)^i)$.
How?
By evaluation and interpolation of y !
Needs $p > 8d^2t^2$.
Does $O(ndt^2)$ probes!

Ben-Or/Tiwari Algorithm (contd.)

$$\Lambda(z) = \prod_{i=1}^{t} (z - M_i(2, 3, 5, \ldots, p_n)).$$

Do it modulo a prime p ! ($p > M_i(2, 3, 5, ..., p_n) < p_n^d$)

Huang and Rao [1990]
Replace $2, 3, 5, \ldots$ by irreducibles $y - a_1, y - a_2, \ldots$ in $GF(p)[y]$.

Compute $\mathbf{B}((y - a_1)^i, \ldots, (y - a_n)^i)$.
How?
By evaluation and interpolation of y !
Needs $p > 8d^2t^2$.
Does $O(ndt^2)$ probes!

Some Algorithms

If we can choose the prime p:

Alg.	# Probes	Deterministic?	Parallel?	Prime
Ben-Or/Tiwari 1988	$O(t)$	Las Vegas	Yes	$p > p_n^d$
Huang/Rao 1990	$O(dt^2)$	Las Vegas	Yes	$p > 8d^2t^2$
Zippel 1979	$O(ndt)$	Monte-Carlo	Some	$p \gg nt$
Kaltofen et. al. 2000	$O(nt)$	Monte-Carlo	None	$p \gg ndt$
Javadi/Monagan 2010	$O(nt)$	Monte-Carlo	Yes!	$p \gg nt^2$
Discrete Logs	$O(t)$	Las Vegas	Yes	$p > (d+1)^n$

Example

Three problems:

- Medium: $n = 10, d = 20, t = 10^2$.
- Big: $n = 15, d = 40, t = 10^4$.
- Very Big: $n = 20, d = 100, t = 10^6$.

Alg.	Prime	Medium	Big	Very Big
Ben-Or/Tiwari	$p > p_n^d$	2^{96}	2^{223}	2^{615}
Huang/Rao	$p > 8d^2t^2$	2^{27}	2^{41}	2^{56}
Zippel	$p \gg nt$	2^{10}	2^{17}	2^{24}
Kaltofen et. al.	$p \gg ndt$	2^{14}	2^{23}	2^{31}
Javadi/Monagan	$p \gg nt^2$	2^{17}	2^{31}	2^{44}
Discrete Logs	$p > d^n$	2^{44}	2^{81}	2^{133}

Our New Algorithm

Step 1 Choose evaluation points $\alpha_1, \ldots, \alpha_n$ at random from $\mathbb{Z}_p^*$.

Step 2 Evaluate $\mathbf{B}(\alpha_1^i, \ldots, \alpha_n^i)$ for $i = 0 \ldots 2T-1$ and compute $\Lambda_0(z) \in \mathbb{Z}_p[z]$.

Step 3 Find the roots $r_1, r_2, \ldots$ of $\Lambda_0(z)$ using Rabin's algorithm.
If lucky? then all $m_i = M_i(\alpha_1, \ldots, \alpha_n)$ are distinct
hence $\deg(\Lambda_0(z)) = t$ and $\{r_1, \ldots, r_t\} = \{m_1, \ldots, m_t\}$.

Step 4 For each x_j in parallel do

4.1 Choose $\beta_j \neq \alpha_j$ at random from $\mathbb{Z}_p$

4.2 Evaluate $\mathbf{B}(\alpha_1^i, \ldots, \beta_j^i, \ldots, \alpha_n^i)$ for $i = 0 \ldots 2t-1$ and compute $\Lambda_j(z)$.

Let $\bar{r}_1, \ldots, \bar{r}_t$ denote the roots of $\Lambda_j(z)$ and $\bar{m}_i = M_i(\alpha_1, \ldots, \beta_j, \ldots, \alpha_n)$.
We have $\{\bar{r}_1, \ldots, \bar{r}_t\} = \{\bar{m}_1, \ldots, \bar{m}_t\}$. Observe:

$$\frac{\bar{m}_i}{m_i} = \left(\frac{\beta_j}{\alpha_j}\right)^{d_{ij}} \Rightarrow \bar{m}_i = \left(\frac{\beta_j}{\alpha_j}\right)^{d_{ij}} m_i \Rightarrow \Lambda_j\left(\left(\frac{\beta_j}{\alpha_j}\right)^{d_{ij}} r_i\right) = 0.$$

4.3 For $i = 1 \ldots t$ do

For $s = 0 \ldots d$ do if $\Lambda_j\left(\left(\frac{\beta_j}{\alpha_j}\right)^s r_i\right) = 0$ then $d_{ij} := s$

Step 5 Solve for the coefficients c_i to determine $f(x_1, \ldots, x_n)$.

Step 6 Check: pick $\alpha \in \mathbb{Z}_p^n$ at random.
If $\mathbf{B}(\alpha) = f(\alpha)$ output $f(x_1, \ldots, x_n)$ else output FAIL.

Theorem 1 (Computational cost)
Our algorithm does $2T + 2nt + 1$ probes to **B** plus $O(T^2 + nt^2 \log p + ndt \log t)$ arithmetic operations in $\mathbb{Z}_p$ using classical algorithms for polynomial arithmetic.

Theorem 2 (Failure probability)
If $p - 2 > 2^{k-2}3(n+1)d(d+3)t^2$ then our algorithm outputs $f(x_1, \ldots, x_n)$ with probability at least $1 - 2^{-k}$.

Schwartz-Zippel lemma [1978]: Let $f \in K[x_1, \ldots, x_n]$ for K a field and $\alpha_1, \ldots, \alpha_n$ be chosen at random from any $S \subset K$.

$$\text{If } f \neq 0 \text{ then } \text{Prob}(f(\alpha_1, \ldots, \alpha_n) = 0) \leq \frac{\deg f}{|S|}.$$

Theorem 1 (Computational cost)
Our algorithm does $2T + 2nt + 1$ probes to **B** plus $O(T^2 + nt^2 \log p + ndt \log t)$ arithmetic operations in $\mathbb{Z}_p$ using classical algorithms for polynomial arithmetic.

Theorem 2 (Failure probability)
If $p - 2 > 2^{k-2} 3(n+1)d(d+3)t^2$ then our algorithm outputs $f(x_1, \ldots, x_n)$ with probability at least $1 - 2^{-k}$.

Schwartz-Zippel lemma [1978]: Let $f \in K[x_1, \ldots, x_n]$ for K a field and $\alpha_1, \ldots, \alpha_n$ be chosen at random from any $S \subset K$.

$$\text{If } f \neq 0 \text{ then } \text{Prob}(f(\alpha_1, \ldots, \alpha_n) = 0) \leq \frac{\deg f}{|S|}.$$

Theorem 1 (Computational cost)
Our algorithm does $2T + 2nt + 1$ probes to **B** plus $O(T^2 + nt^2 \log p + ndt \log t)$ arithmetic operations in $\mathbb{Z}_p$ using classical algorithms for polynomial arithmetic.

Theorem 2 (Failure probability)
If $p - 2 > 2^{k-2}3(n+1)d(d+3)t^2$ then our algorithm outputs $f(x_1, \ldots, x_n)$ with probability at least $1 - 2^{-k}$.

Schwartz-Zippel lemma [1978]: Let $f \in K[x_1, \ldots, x_n]$ for K a field and $\alpha_1, \ldots, \alpha_n$ be chosen at random from any $S \subset K$.

$$\text{If } f \neq 0 \text{ then } \text{Prob}(f(\alpha_1, \ldots, \alpha_n) = 0) \leq \frac{\deg f}{|S|}.$$

About the Proofs

Lemma 1: For random $\alpha_1, \ldots, \alpha_n \in \mathbb{Z}_p^*$, the probability that any two monomials evaluate to the same value is at most $\binom{t}{2}\frac{d}{p-1}$.

Lemma 2: The probability that we cannot uniquely determine the degree of all monomials in x_j is at most $\frac{d^2t^2}{4(p-1)}$.

Proof of Lemma 1: Consider

$$A = \prod_{1 \le i < j \le t} \left(M_i(x_1, \ldots, x_n) - M_j(x_1, \ldots, x_n)\right).$$

We have $A(\alpha_1, \ldots, \alpha_n) = 0$ iff two monomial evaluations collide. Now $d \ge \deg f$ thus $\deg(A) \le \binom{t}{2}d$. Lemma 1 follows from the Schwartz-Zippel lemma since $|S| = p - 1$.

Benchmarks

- For polynomials in $n = 12$ variables with total degree $d = 30$.
- Choose t monomials and coefficients randomly modulo p a 31 bit prime.

t	New Algorithm (in Cilk)				Zippel (in C)	
	Time (seconds)			Probes	Time	Probes
	1 core	4 cores	12 cores			
255	0.70	0.19	0.11	6121	22.17	134664
507	2.33	0.60	0.25	12169	83.44	259594
1019	8.20	2.10	0.75	24457	316.23	498945
2041	30.2	7.62	2.61	48985	1195.13	952351
4074	113.1	28.6	9.90	97777	4575.83	1841795
8139	435.4	110.5	36.5	195337	>10000	-

Amdahl's law: Speedup for N cores is $\leq \dfrac{Tot}{\frac{Tot-Seq}{N} + Seq}$.

	1 core				4 cores	
t	time	roots	solve	probes	time	speedup
255	0.54	0.05	0.00	0.41	0.18	(3x)
507	2.02	0.18	0.02	1.48	0.67	(3.02x)
1019	7.94	0.65	0.08	5.76	2.58	(3.08x)
2041	31.3	2.47	0.32	22.7	9.94	(3.15x)
4074	122.3	9.24	1.26	90.0	38.9	(3.14x)
8139	484.6	34.7	5.02	357.3	152.5	(3.17x)

For $i = 13$ speedup < 3.21 on 4 cores and < 6.31 on 12 cores.

Second Attempt:

	1 core				4 cores		12 cores	
t	time	roots	solve	probe	time	(speedup)	time	(speedup)
255	0.688	0.01	0.01	0.40	0.186	(3.70x)	0.106	(6.5x)
507	2.33	0.05	0.02	1.53	0.603	(3.86x)	0.250	(9.3x)
1019	8.20	0.14	0.07	5.97	2.10	(3.90x)	0.748	(10.96x)
2041	30.17	0.34	0.26	23.6	7.62	(3.96x)	2.61	(11.56x)
4074	113.1	0.87	1.06	93.5	28.6	(3.96x)	9.90	(11.78x)
8139	435.3	2.25	4.20	371.7	110.5	(3.94x)	36.46	(11.95x)

Amdahl's law: Speedup for N cores is $\leq \dfrac{Tot}{\frac{Tot-Seq}{N} + Seq}$.

	1 core				4 cores	
t	time	roots	solve	probes	time	speedup
255	0.54	0.05	0.00	0.41	0.18	(3x)
507	2.02	0.18	0.02	1.48	0.67	(3.02x)
1019	7.94	0.65	0.08	5.76	2.58	(3.08x)
2041	31.3	2.47	0.32	22.7	9.94	(3.15x)
4074	122.3	9.24	1.26	90.0	38.9	(3.14x)
8139	484.6	34.7	5.02	357.3	152.5	(3.17x)

For $i = 13$ speedup < 3.21 on 4 cores and < 6.31 on 12 cores.

Second Attempt:

	1 core				4 cores		12 cores	
t	time	roots	solve	probe	time	(speedup)	time	(speedup)
255	0.688	0.01	0.01	0.40	0.186	(3.70x)	0.106	(6.5x)
507	2.33	0.05	0.02	1.53	0.603	(3.86x)	0.250	(9.3x)
1019	8.20	0.14	0.07	5.97	2.10	(3.90x)	0.748	(10.96x)
2041	30.17	0.34	0.26	23.6	7.62	(3.96x)	2.61	(11.56x)
4074	113.1	0.87	1.06	93.5	28.6	(3.96x)	9.90	(11.78x)
8139	435.3	2.25	4.20	371.7	110.5	(3.94x)	36.46	(11.95x)

The Discrete Logs Method

Giesbrecht, Labahn and Lee [2006] presented a variation of Ben-Or/Tiwari for numerical coefficients. They evaluate at powers of primitive elements in $\mathbb{C}$ of relatively prime order. We adapt this approach for $\mathbb{Z}_p$ as follows.

Pick the prime $p = q_1 \times q_2 \times \cdots \times q_n + 1$ where $q_i > d$ and $\gcd(q_i, q_j) = 1$.

Pick a generator α of $\mathbb{Z}_p^*$ and set $w_j = \alpha^{\frac{p-1}{q_j}}$ [w_j have relatively prime order].

Evaluate at $f(w_1^i, w_2^i, ..., w_n^i)$ for $0 \leq i < 2T - 1$. Hence

$$m_i = M_i(w_1, \ldots, w_n) = w_1^{d_{i1}} \times \cdots \times w_n^{d_{in}} = \alpha^{\frac{p-1}{q_1} d_{i1} + \cdots + \frac{p-1}{q_n} d_{in}}$$

$$\Rightarrow \quad \log_\alpha m_i = \tfrac{p-1}{q_1} d_{i1} + \cdots + \tfrac{p-1}{q_j} d_{ij} + \cdots + \tfrac{p-1}{q_n} d_{in}$$

$$\Rightarrow \quad \log_\alpha m_i \equiv 0 + \cdots + \tfrac{p-1}{q_j} d_{ij} + \cdots + 0 \mod q_j$$

- The discrete logarithm is efficient if $p - 1$ has no large prime factors.
- Requires $p > (d+1)^n$ which may force multi-precision arithmetic.

Thank you.


M. Ben-Or and P. Tiwari.
A deterministic algorithm for sparse multivariate polynomial interpolation.
In *Proc. of STOC '88*, pages 301–309. ACM, 1988.


M. Giesbrecht, G. Labahn and W. Lee.
Symbolic-numeric sparse interpolation of multivariate polynomials.
J. Symb. Comput., 44:943–959, 2009.


M. A. Huang and A. J. Rao.
Interpolation of sparse multivariate polynomials over large finite fields with applications.
Journal of Algorithms, 33:204–228, 1999.


Mahdi Javadi and Michael Monagan.
Parallel Sparse Polynomial Interpolation over Finite Fields.
In *Proceedings of PASCO '2010*, ACM Press, pp. 160–168, 2010.


Jack Schwartz.
Fast probabilistic algorithms for verification of polynomial identities.
J. ACM, 27:701−717, 1980.


Richard Zippel.
Probabilistic algorithms for sparse polynomials.
In *Proc. of EUROSAM '79*, pages 216–226. Springer-Verlag, 1979.