Parallel Sparse Polynomial Interpolation over Finite Fields

Michael Monagan

Department of Mathematics,
Simon Fraser University

PASCO, July 21–23, 2010

This is a joint work with Mahdi Javadi

The Problem

- We want to compute $g = \gcd(f_1, f_2) \in \mathbb{Z}_p[x_1, \ldots, x_n]$ for some prime p.
- We choose p to be a 31.5 bit prime on a 64 bit machine (a C limitation, one could use a 63 bit prime).
- For a sparse g, we need to do sparse interpolation.
- In general, the target polynomial f is represented with a black box.

- Zippel in 1979 presented a probabilistic method to determine g given the black box B which computes:

$$\gcd(f1(x_1, \alpha_2, \ldots, \alpha_n), f_2(x_1, \alpha_2, \ldots, \alpha_n)) \in \mathbb{Z}_p[x_1]$$

The Ben-Or/Tiwari Algorithm (1988) for $\mathbb{Z}$

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

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

Let $d \geq \deg(f)$ be a bound on degree of f.

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

Step 2 Compute the linear generator $\Lambda(z)$ for the sequence $v_0, v_1, \ldots, v_{2T-1}$ using the Berlekamp/Massey algorithm. Theorem:

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

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

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

Step 5 Solve for the coefficients c_i.

Ben-Or/Tiwari Algorithm (contd.)

Ben-Or/Tiwari algorithm is deterministic and does $2T$ probes to the black box.

For characteristic p: requires $p > \max_i M_i(2, 3, 5, ..., p_n) < p_n^d$.

In 1990 Huang and Rao replaced the primes $2, 3, 5, \ldots$ by irreducible polynomials $y - a_i$ in $GF(q)[y]$.

- Does $O(ndt^2)$ probes $\Rightarrow$ worse than Zippel's algorithm.
- Also, need to factor a bivariate polynomial $GF(q)[x, y]$.

In 2000 Kaltofen, Lee and Lobo presented a hybrid of Zippel's algorithm with Ben-Or/Tiwari algorithm.

Their algorithm is a modification of Zippel's algorithm:

- For univariate interpolation they race Newton's interpolation algorithm with univariate Ben-Or/Tiwari algorithm.
- They use early termination and hence their algorithm is Monte Carlo.

The Discrete Logs Method

In 2006, Giesbrecht, Labahn and Lee presented a variation of Ben-Or/Tiwari for numerical coefficients. They evalaluate at powers of primitive elements in $\mathbb{C}$ of relatively prime order. We observe that this approach can also work in $\mathbb{Z}_p$ 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 w of $\mathbb{Z}_p^*$ and set $w_j = w^{\frac{p-1}{q_j}}$.
Evaluate at $f(w_1^i, w_2^i, ..., w_n^i)$ for $0 \leq i < 2T - 1$. Hence

$$m_i = M_i(w_1, \ldots, w_n) = w_1^{d_{i1}} \times w_2^{d_{i2}} \times \cdots \times w_n^{d_{in}} = w^{\frac{p-1}{q_1} d_{i1} + \cdots + \frac{p-1}{q_n} d_{in}}$$

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

To compute d_{ij}, solve this modulo q_j.

- The discrete log is efficient; we choose $p - 1$ with no large prime factors.
- Requires $p > (d+1)^n$ which may force multi-precision arithmetic.

Comparison Chart

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	$p > 8d^2t^2$
Discrete Logs	$O(t)$	Las Vegas	Yes	$p > (d+1)^n$
Zippel 1979	$O(ndt)$	Monte-Carlo	Some	$p \gg nt$
Kaltofen et. al. 2000	$O(nt)$	Monte-Carlo	Less	$p \gg ndt$
Javadi/Monagan 2010	$O(nt)$	Monte-Carlo	Yes!	$p \gg nt^2$

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}
Discrete Logs	$p > d^n$	2^{44}	2^{81}	2^{133}
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}

Our New Algorithm

The Algorithm:

1. Choose evaluation points $\alpha_1, \ldots, \alpha_n$ at random from $\mathbb{Z}_p^*$.
2. Evaluate $f(\alpha_1^i, \ldots, \alpha_n^i)$ for $i = 0 \ldots 2T-1$ and compute $\Lambda_0(z) \in \mathbb{Z}_p[z]$.
3. Find the roots of $\Lambda_0(z)$: $r_1, \ldots, r_t$ using Rabin's algorithm. If $\deg(\Lambda_0(z)) = t$ we have $\{r_1, \ldots, r_t\} = \{m_1, \ldots, m_t\}$ where $m_i = M_i(\alpha_1, \ldots, \alpha_n)$.
4. For each x_j do the following in parallel:
   - 4.1 Choose β_j at random from $\mathbb{Z}_p^*$ such that (β_j/α_j) has order $> d$.
   - 4.2 Evaluate $f(\alpha_1^i, \ldots, \beta_j^i, \ldots, \alpha_n^i)$ 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:

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

   - 4.3 For $k = 1 \ldots t$ do
     - 4.3.1 For $s = 0 \ldots d$ do if $\Lambda_j\left(\left(\frac{\beta_j}{\alpha_j}\right)^s r_k\right) = 0$ then $d_{kj} = s$ w.h.p.

Our New Algorithm (contd.)

Our algorithm can only work if all monomial evaluations $(M_i(\alpha_1, \ldots, \alpha_n))$ are distinct.

Theorem 1: For random evaluations $\alpha_1, \ldots, \alpha_n$, the probability that two or more monomials evaluate to the same value is at most: $\binom{t}{2}\frac{d}{p-1}$.

Proof: 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.

Schwartz-Zippel lemma: If $f \in K[x_1, \ldots, x_n]$ is non-zero $r_1, \ldots, r_n$ are chosen at random from any subset S of a field K then

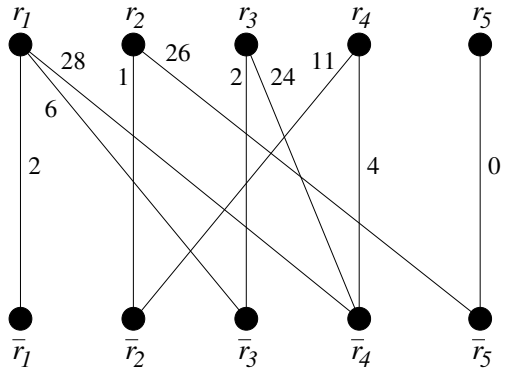
$$\mathrm{Prob}(f(r_1, \ldots, r_n) = 0) \le \frac{\deg f}{|S|}.$$

We have $d \ge \deg f$ and thus $\deg(A) \le \binom{t}{2} d$ and $|S| = p - 1$.

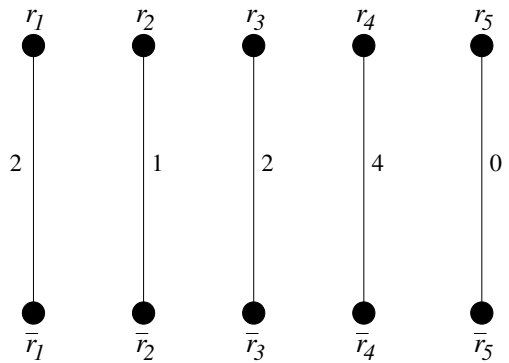
Theorem 2: If $\deg(\Lambda_0) = \deg(\Lambda_j) = t$, then the probability that we will not be able to uniquely compute the degrees in x_j is at most $\frac{d^2 t^2}{4(p-1)}$.

Our New Algorithm (contd.)

We obtain the following bipartite graph. r_i is connected to $\bar{r}_j$ with the weight e iff $\bar{r}_j = r_i(\frac{\beta_j}{\alpha_j})^e$.



This graph has a unique perfect matching.



Our New Algorithm (contd.)

Theorem: If the bipartite graph G does not have a unique perfect matching, with one more set of evaluations ($2t$ more probes) we can uniquely compute d_{ij}, the degrees of all monomials in x_j.

Remarks:

- To compute the degrees of the monomials in the last variable x_n, we do not need to do any more probes to the black box. We have

$$m_i = \alpha_1^{d_{i1}} \times \cdots \times \alpha_{n-1}^{d_{i(n-1)}} \times \alpha_n^{d_{in}}.$$

- Number of Probes: Between $2nt$ and $4nt$.
- One can compute the degrees of the monomials in $x_1, \ldots, x_{n-1}$ in Parallel.
- If the number of terms t is known, our algorithm is Las Vegas.
  If a bound $T \geq t$ is given, the algorithm is Monte Carlo.
- Like the racing algorithm, our algorithm is not sensitive to a bad degree bound (unlike Zippel's algorithm).

Benchmarks

- Random polynomials with approximately 2^i terms, $n = 12$ variables and total degree 30.
- Degree bound: $d = 30$.

i	t	New Algorithm		Zippel		ProtoBox
		Time (4 cores)	Probes	Time	Probes	Probes
1	2	0.00 (0.00)	44	0.03	1736	67
2	4	0.00 (0.00)	96	0.04	3038	121
3	8	0.00 (0.00)	192	0.08	5053	250
4	15	0.00 (0.00)	360	0.20	10230	470
5	32	0.02 (0.01)	768	0.54	18879	962
6	63	0.04 (0.02)	1512	1.79	36735	1856
7	127	0.15 (0.05)	3048	6.10	69595	3647
8	255	0.54 (0.17)	6120	22.17	134664	7055
9	507	2.01 (0.60)	12168	83.44	259594	13440
10	1019	7.87 (2.33)	24456	316.23	498945	26077
11	2041	31.0 (9.16)	48984	1195.13	952351	DNF
12	4074	122.3 (35.9)	97776	4575.83	1841795	DNF
13	8139	484.6 (141.)	195336	>10000	-	DNF

Benchmarks (contd.)

i	t	1 core				4 cores		
		time	roots	solve	probes	time 1	time 2	speedup
6	63	0.04	0.01	0.00	0.04	0.02	0.02	
7	127	0.15	0.02	0.00	0.15	0.06	0.05	(2.5x)
8	255	0.54	0.05	0.00	0.41	0.18	0.17	(3x)
9	507	2.02	0.18	0.02	1.48	0.67	0.60	(3.02x)
10	1019	7.94	0.65	0.08	5.76	2.58	2.33	(3.08x)
11	2041	31.3	2.47	0.32	22.7	9.94	9.16	(3.15x)
12	4074	122.3	9.24	1.26	90.0	38.9	35.9	(3.14x)
13	8139	484.6	34.7	5.02	357.3	152.5	141.5	(3.17x)

Amdahl's law: for $i = 13$, the maximum speedup on 4 cores is

$$\frac{Tot}{\frac{Tot - Seq}{\#cores} + Seq} = 3.21.$$

Thank you!

We are currently coding arithmetic and root finding in $\mathbb{Z}_p[x]$ for 63 bit primes for our algorithm for large t and so we can implement the Discrete Logarithm method for larger n.

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	$p > 8d^2t^2$
Discrete Logs	$O(t)$	Las Vegas	Yes	$p > (d+1)^n$
Zippel 1979	$O(ndt)$	Monte-Carlo	Some	$p \gg nt$
Kaltofen et. al. 2000	$O(nt)$	Monte-Carlo	Less	$p \gg ndt$
Javadi/Monagan 2010	$O(nt)$	Monte-Carlo	Yes	$p \gg nt^2$