Parallel sparse interpolation using small primes

Mohamed Khochtali
Daniel S. Roche*
Xisen Tian

United States Naval Academy
Annapolis, Maryland, USA

PASCO 2015
Bath, UK, July 10, 2015
The Problem

Sample point → Unknown Function → Evaluation

What sparse polynomial is in here?

**Algorithm input:** Black box for a sparse polynomial
Bounds on the size of the polynomial

**Algorithm output:** List of nonzero coefficients and exponents
Black Box Model (univariate version)

\[ f \in \mathbb{Z}[x] \]
\[ f = c_1 x^{e_1} + c_2 x^{e_2} + \cdots + c_t x^{e_t} \]

**Black box input**

\[ q \in \mathbb{Z} \]
\[ h \in (\mathbb{Z}/q\mathbb{Z})[z] \]
\[ g \in (\mathbb{Z}/q\mathbb{Z})[z] \]

**Black box output**

\[ f(h) \mod g \text{ over } (\mathbb{Z}/q\mathbb{Z})[z] \]
Black Box Model (multivariate version)

\[ f \in \mathbb{Z}[x_1, x_2 \ldots, x_n] \]
\[ f = c_1 x_1^{e_{11}} x_2^{e_{12}} \cdots x_n^{e_{1n}} + \cdots + c_t x_1^{e_{t1}} x_2^{e_{t2}} \cdots x_n^{e_{tn}} \]

**Black box input**

\[ q \in \mathbb{Z} \]
\[ h_1, h_2, \ldots, h_n \in (\mathbb{Z}/q\mathbb{Z})[z] \]
\[ g \in (\mathbb{Z}/q\mathbb{Z})[z] \]

**Black box output**

\[ f(h_1, h_2, \ldots, h_n) \mod g \text{ over } (\mathbb{Z}/q\mathbb{Z})[z] \]
Brief History

“Big prime” methods

- Prony (1795)
- Blahut (1979)
- Zippel (1979)
- Ben-Or & Tiwari (1989)
- Kaltofen & Lakshman (1989)
- Javadi & Monagan (2010)
- van der Hoeven & Lecerf (2014)

“Small prime” methods

- Grigoriev & Karpinsky (1987)
- Garg & Schost (2009)
- R. & Giesbrecht (2011)
- Arnold, Giesbrecht, R. (2014)
Big Prime Interpolation (Kaltofen 2010)

1. Choose $q \gg \deg f$
2. Find PRU $\omega \in \mathbb{Z}/q\mathbb{Z}$
3. Evaluate $f(1), f(\omega), \ldots, f(\omega^{2T-1})$
4. Berlekamp-Massey to find $\Gamma(z)$
5. Compute roots $\zeta_1, \ldots, \zeta_t$ of $\Gamma$
6. Compute discrete logs of $\zeta_i$'s
7. Solve transposed Vandermonde system
Big Prime Interpolation (Kaltofen 2010)

1. Choose $q \gg \deg f$
2. Find PRU $\omega \in \mathbb{Z}/q\mathbb{Z}$
3. Evaluate $f(1), f(\omega), \ldots, f(\omega^{2T-1})$
4. Berlekamp-Massey to find $\Gamma(z)$
5. Compute roots $\zeta_1, \ldots, \zeta_t$ of $\Gamma$
6. Compute discrete logs of $\zeta_i$'s
7. Solve transposed Vandermonde system

The expensive parts
Big Prime Interpolation (Kaltofen 2010)

1. Choose $q \gg \deg f$
2. Find PRU $\omega \in \mathbb{Z}/q\mathbb{Z}$
3. Evaluate $f(1), f(\omega), \ldots, f(\omega^{2T-1})$ Easy
4. Berlekamp-Massey to find $\Gamma(z)$ Hard
5. Compute roots $\zeta_1, \ldots, \zeta_t$ of $\Gamma$ Hard
6. Compute discrete logs of $\zeta_i$'s Easy
7. Solve transposed Vandermonde system Hard

The expensive parts

How easy to parallelize?
### Small Primes Interpolation

1. Repeat $O(\log D)$ times:
2. Choose $q \gg \max c_i$, $p \gg T$
3. Evaluate $f(z) \mod (z^p - 1)$
4. Save nonzero coefficients and exponents
5. Correlate exponents between the images
6. CRT to find actual exponents
## Small Primes Interpolation

1. Repeat $O(\log D)$ times:
2. Choose $q \gg \max c_i, \quad p \gg T$
3. Evaluate $f(z) \mod (z^p - 1)$
4. Save nonzero coefficients and exponents
5. Correlate exponents between the images
6. CRT to find actual exponents

### The expensive parts
# Small Primes Interpolation

1. **Repeat** $O(\log D)$ times: \textbf{Easy!}
2. **Choose** $q \gg \max c_i, \quad p \gg T$
3. **Evaluate** $f(z) \mod (z^p - 1)$ \textbf{Hard}
4. **Save nonzero coefficients and exponents**
5. **Correlate exponents between the images**
6. **CRT to find actual exponents** \textbf{Easy!}

## The expensive parts
How easy to parallelize?
What about multivariate?

Option 1: Kronecker

\[ f(x_1, x_2, \ldots, x_n) \leftrightarrow f(z, z^D, z^{D^2}, \ldots, z^{D^{n-1}}) \]

Performed implicitly *in the evaluations.*
What about multivariate?

**Option 1: Kronecker**

\[
f(x_1, x_2, \ldots, x_n) \leftrightarrow f(z, z^D, z^{D^2}, \ldots, z^{D^{n-1}})
\]

Performed implicitly *in the evaluations.*

**Option 2: Variable by variable**

(Zippel ’79): Iterative
(Javadi & Monagan ’10): *In parallel*
Small Primes Algorithm

1. Repeat \( \gg \log D \) times in parallel:
2. Choose \( q \gg \max c_i, \quad p \gg T \)
3. Evaluate \( f(z) \mod (z^p - 1) \)
4. Save nonzero coefficients and exponents
5. Correlate exponents between the images
6. CRT to find actual exponents
### Parallel Small Primes Algorithm

1. **Repeat** $\gg \log D$ times **in parallel**:  
2. Choose $q \gg \max c_i$, $p \gg T$  
3. Evaluate $f(z) \mod (z^p - 1)$  
4. Save nonzero coefficients and exponents  
5. Correlate exponents between the images  
6. CRT to find actual exponents

---

We only needed to parallelize the main loop.
# Parallel Small Primes Algorithm

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.</td>
<td>Choose random prime $q$, random $\alpha \in \mathbb{Z}/q\mathbb{Z}$</td>
</tr>
<tr>
<td>1.</td>
<td>Repeat $\gg \log D$ times in parallel:</td>
</tr>
<tr>
<td>2.</td>
<td>Choose $p \gg T$</td>
</tr>
<tr>
<td>3.</td>
<td>Evaluate $f(\alpha z) \mod (z^p - 1)$</td>
</tr>
<tr>
<td>4.</td>
<td>Save nonzero coefficients and exponents</td>
</tr>
<tr>
<td>5.</td>
<td>Correlate exponents between the images</td>
</tr>
<tr>
<td>6.</td>
<td>CRT to find actual exponents</td>
</tr>
</tbody>
</table>

Diversification trick (Giesbrecht & R. 2011)
Parallel Small Primes Algorithm

0. Choose random prime $q$, random $\alpha \in \mathbb{Z}/q\mathbb{Z}$

1. Repeat $\gg \log D$ times in parallel:
   
   2. Choose $p \in O(T \log D)$
   
   3. Evaluate $f(\alpha z) \mod (z^p - 1)$

   4. Save nonzero coefficients and exponents

   5. Correlate exponents between the images

   6. CRT to find actual exponents

   “OK primes” trick (Arnold, Giesbrecht, R. 2013)
## Parallel Small Primes Heuristic

0. Choose random prime $q$, random $\alpha \in \mathbb{Z}/q\mathbb{Z}$

1. Repeat $\lceil \ell \log D \rceil$ times in parallel:
   2. Choose $p \approx kT$
   3. Evaluate $f(\alpha z) \mod (z^p - 1)$
   4. Save nonzero coefficients and exponents
   5. Correlate exponents between the images
   6. CRT to find actual exponents

---

Throw caution to the wind; $k, \ell$ determined experimentally
What is the communication?

**Sent TO each process:**

\( p, \alpha, q, \) and access to the black box

**Received FROM each process:**

A \((\text{coeff, expon, prime})\) triple for each nonzero term

\[
\begin{array}{ccc}
& c \mod q & c \mod q \\
& e \mod p & e \mod p \\
& p & p \\
\end{array}
\]
What is the communication?

**Sent TO each process:**
\( p, \alpha, q, \) and access to the black box

**Received FROM each process:**
A \((\text{coeff, expon, prime})\) triple for each nonzero term

\[
\begin{array}{ccc}
  c \mod q & c \mod q & \cdots \\
  e \mod p & e \mod p & \cdots \\
  p & p & \cdots \\
\end{array}
\]

**Gathering the images:**
List comes in sorted by the primes \( p_i \)
We then **sort by coefficients** to gather exponent images.
An example
0. Initial set-up

**Given**

Unknown $f$ has $n = 2$ variables, max degree $< 10$, and $\leq 3$ nonzero terms.

- Choose $q = 11$ (for coefficient field)
- Choose $\alpha = 5$ (for diversification)
- Choose small primes $p_1 = 7, p_2 = 13, p_3 = 17$
An example

1. Parallel evaluation

Process 1 receives:

\[ n = 2, \quad D = 10, \quad q = 11, \quad \alpha = 5, \quad p = 7 \]
An example
1. Parallel evaluation

Process 1 receives:
\[ n = 2, \quad D = 10, \quad q = 11, \quad \alpha = 5, \quad p = 7 \]

- Evaluate \( f(5z, (5z)^{10}) \mod (z^7 - 1) \)
- \[ = 3z^6 + 7z^3 + 2z \]
An example

1. Parallel evaluation

Process 1 receives:

\[ n = 2, \quad D = 10, \quad q = 11, \quad \alpha = 5, \quad p = 7 \]

- Evaluate \( f(5z, (5z)^{10}) \mod (z^7 - 1) \)
- \( = 3z^6 + 7z^3 + 2z \)
- Add nonzero terms to the list

<table>
<thead>
<tr>
<th>coefficient</th>
<th>3</th>
<th>7</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>exponent</td>
<td>6</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>prime</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>
An example
1. Parallel evaluation

Process 2 receives:

\[ n = 2, \quad D = 10, \quad q = 11, \quad \alpha = 5, \quad p = 13 \]

- Evaluate \( f(5z, (5z)^{10}) \mod (z^{13} - 1) \)
- \( = 10z^7 + 2z^4 \)
- Add nonzero terms to the list

<table>
<thead>
<tr>
<th>coefficient</th>
<th>3</th>
<th>7</th>
<th>2</th>
<th>10</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>exponent</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>prime</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>13</td>
<td>13</td>
</tr>
</tbody>
</table>
An example

1. Parallel evaluation

**Process 3 receives:**

\[ n = 2, \quad D = 10, \quad q = 11, \quad \alpha = 5, \quad p = 17 \]

- Evaluate \( f(5z, (5z)^{10}) \) mod \((z^{17} - 1)\)
- \[ = 2z^9 + 7z^8 + 3z^3 \]
- Add nonzero terms to the list

<table>
<thead>
<tr>
<th>coefficient</th>
<th>3</th>
<th>7</th>
<th>2</th>
<th>10</th>
<th>2</th>
<th>2</th>
<th>7</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>exponent</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>7</td>
<td>4</td>
<td>9</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>prime</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>13</td>
<td>13</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
</tbody>
</table>
An example

2. Recovering $f$

Main process knows $n = 2$, $q = 11$, $\alpha = 5$, and receives

<table>
<thead>
<tr>
<th>coefficient</th>
<th>3</th>
<th>7</th>
<th>2</th>
<th>10</th>
<th>2</th>
<th>2</th>
<th>7</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>exponent</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>7</td>
<td>4</td>
<td>9</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>prime</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>13</td>
<td>13</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
</tbody>
</table>
An example

2. Recovering $f$

**Main process** knows $n = 2$, $q = 11$, $\alpha = 5$, and receives

<table>
<thead>
<tr>
<th>coefficient</th>
<th>10</th>
<th>7</th>
<th>7</th>
<th>3</th>
<th>3</th>
<th>2</th>
<th>2</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>exponent</td>
<td>7</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>prime</td>
<td>13</td>
<td>17</td>
<td>7</td>
<td>17</td>
<td>7</td>
<td>17</td>
<td>13</td>
<td>7</td>
</tr>
</tbody>
</table>

- Sort the table by coefficients
An example

2. Recovering $f$

**Main process** knows $n = 2$, $q = 11$, $\alpha = 5$, and receives

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>10</th>
<th>7</th>
<th>7</th>
<th>3</th>
<th>3</th>
<th>2</th>
<th>2</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponent</td>
<td>7</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Prime</td>
<td>13</td>
<td>17</td>
<td>7</td>
<td>17</td>
<td>7</td>
<td>17</td>
<td>13</td>
<td>7</td>
</tr>
</tbody>
</table>

- Sort the table by coefficients
- Perform CRT on each sufficiently-large group

$$e_1 = 59 \quad e_2 = 20 \quad e_3 = 43$$

$$f(5z, (5z)^{10}) = 7z^{59} + 3z^{20} + 2z^{43}$$
An example

2. Recovering $f$

Main process knows $n = 2, \quad q = 11, \quad \alpha = 5$, and receives

<table>
<thead>
<tr>
<th>coefficient</th>
<th>10</th>
<th>7</th>
<th>7</th>
<th>3</th>
<th>3</th>
<th>2</th>
<th>2</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>exponent</td>
<td>7</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>prime</td>
<td>13</td>
<td>17</td>
<td>7</td>
<td>17</td>
<td>7</td>
<td>17</td>
<td>13</td>
<td>7</td>
</tr>
</tbody>
</table>

- Sort the table by coefficients
- Perform CRT on each sufficiently-large group

\[
e_1 = 59 \quad e_2 = 20 \quad e_3 = 43
\]

\[
f(5z, (5z)^{10}) = 7z^{59} + 3z^{20} + 2z^{43}
\]

- Undo diversification

\[
f(z, z^{10}) = 9z^{59} + z^{20} + 4z^{43}
\]

- Undo Kronecker

\[
f(x, y) = 9x^9y^5 + y^2 + 4x^3y^4
\]
Complexity analysis

Each of $\lceil \ell n \lg D \rceil$ processes evaluates modulo $(z^{p_i} - 1)$, where $p_i \approx kT$.

In theory, we need $\ell \in O(1)$ and $k \in O(n \log D)$, resulting in $O(n^2 \log^2 D)$ potential speedup of a $O(n^2 T \log^2 D)$ algorithm.

Heuristically, $k \in O(1)$, resulting in $O(n \log D)$ parallel speedup of a $O(nT \log D)$ algorithm.
Libraries

- FLINT used for dense arithmetic (evaluations)
- We made a small `fmpz_sparse` type for FLINT
- Used Open MPI for parallelism (more scalable than threads, but must be careful with communication)
Experiment 1

Benchmark copied from (van der Hoeven & Lecerf 2014): Interpolating the product of $m$ random 3-sparse polynomials, each 20 variables, degree 40, single-precision coefficients.

We first compared our heuristic algorithm to Mathemagix with and without parallelization.
Experiment 2

Same benchmark with # of polynomials fixed at $m = 6$, varying # of processes and the degree.

**Hardware was limited:**
Using Core i7, 6 cores, each hyper-threaded.

Measuring parallel speedup over the single-threaded version of our code.
To-Do List

- Explore constants $k, \ell$ more deeply.
  
  Try to balance larger $\ell$, smaller $k$

- Incorporate randomized Kronecker substitution
  (Arnold & R. ’14)

- Run on more impressive hardware

- Incorporate with new sparse multiplication algorithm?

- Use for signal processing in exponential analysis?
## Timings

<table>
<thead>
<tr>
<th>vars</th>
<th>terms</th>
<th>maxdeg</th>
<th>$\mu$</th>
<th>$\lambda$</th>
<th>Mathemagix</th>
<th>Ours (single)</th>
<th>Ours (multi)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>3</td>
<td>40</td>
<td>14</td>
<td>50000</td>
<td>0.078</td>
<td>0.035</td>
<td>0.029</td>
</tr>
<tr>
<td>20</td>
<td>9</td>
<td>80</td>
<td>18</td>
<td>5000</td>
<td>0.155</td>
<td>0.151</td>
<td>0.048</td>
</tr>
<tr>
<td>20</td>
<td>27</td>
<td>120</td>
<td>18</td>
<td>6900</td>
<td>0.305</td>
<td>0.329</td>
<td>0.116</td>
</tr>
<tr>
<td>20</td>
<td>81</td>
<td>160</td>
<td>18</td>
<td>4900</td>
<td>0.598</td>
<td>0.323</td>
<td>0.085</td>
</tr>
<tr>
<td>20</td>
<td>243</td>
<td>200</td>
<td>17</td>
<td>9900</td>
<td>2.156</td>
<td>0.785</td>
<td>0.175</td>
</tr>
<tr>
<td>20</td>
<td>729</td>
<td>240</td>
<td>15</td>
<td>39900</td>
<td>5.053</td>
<td>3.084</td>
<td>0.814</td>
</tr>
<tr>
<td>20</td>
<td>2187</td>
<td>280</td>
<td>14</td>
<td>80050</td>
<td>13.333</td>
<td>8.714</td>
<td>2.225</td>
</tr>
<tr>
<td>20</td>
<td>6561</td>
<td>320</td>
<td>13</td>
<td>321300</td>
<td>41.070</td>
<td>43.911</td>
<td>10.605</td>
</tr>
</tbody>
</table>

Thank you!