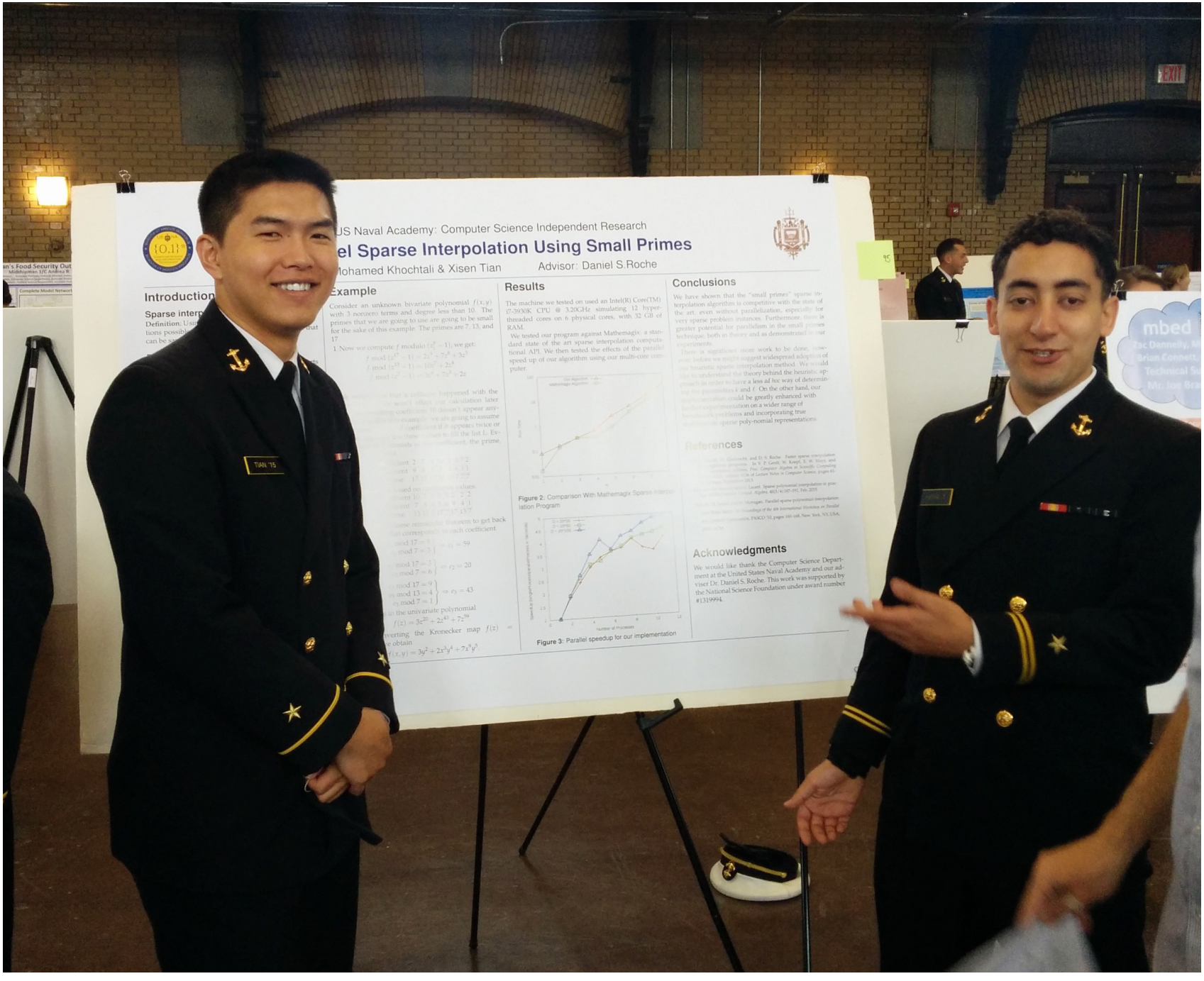
Parallel sparse interpolation using small primes

Mohamed Khochtali
Daniel S. Roche*
Xisen Tian

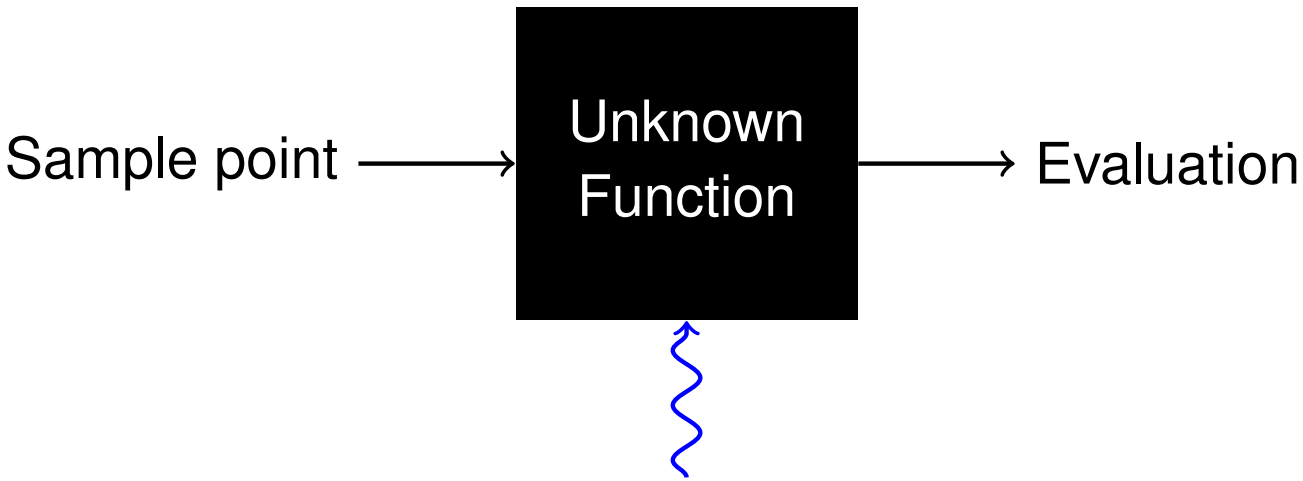
United States Naval Academy
Annapolis, Maryland, USA

PASCO 2015

Bath, UK, July 10, 2015


US Naval Academy: Computer Science Independent Research
el Sparse Interpolation Using Small Primes
Mohamed Khochtali & Xisen Tian
Advisor: Daniel S.Roche
Introduction
Example
Results
Conclusions
References
Acknowledgments
Figure 2: Comparison With Mathemagix Sparse Interpolation Program
Figure 3: Parallel speedup for our implementation
TIAN '15
EXIT

The Problem



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

$f(h) \bmod g$ 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

$f(h_1, h_2, \ldots, h_n) \bmod g$ 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 Γ
6. Compute discrete logs of ζ_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 Γ
6. Compute discrete logs of ζ_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 Γ	**Hard**
6.	Compute discrete logs of ζ_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 $q \gg \max c_i, \quad p \gg T$
3. Evaluate $f(z) \bmod (z^p - 1)$
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 $f(z) \bmod (z^p - 1)$
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:	**Easy!**
2.	Choose $q \gg \max c_i, \quad p \gg T$	
3.	Evaluate $f(z) \bmod (z^p - 1)$	**Hard**
4.	Save nonzero coefficients and exponents	
5.	Correlate exponents between the images	
6.	CRT to find actual exponents	**Easy!**

The expensive parts

How easy to parallelize?

What about multivariate?

Option 1: Kronecker

$$f(x_1, x_2, \ldots, x_n) \longleftrightarrow f(z, z^D, z^{D^2}, \ldots, z^{D^{n-1}})$$

Performed implicitly *in the evaluations*.

What about multivariate?

Option 1: Kronecker

$$f(x_1, x_2, \ldots, x_n) \longleftrightarrow f(z, z^D, z^{D^2}, \ldots, z^{D^{n-1}})$$

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 $f(z) \bmod (z^p - 1)$
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 $q \gg \max c_i, \quad p \gg T$
3. Evaluate $f(z) \bmod (z^p - 1)$
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

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

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 $f(\alpha z) \bmod (z^p - 1)$
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 $f(\alpha z) \bmod (z^p - 1)$
4. Save nonzero coefficients and exponents
5. Correlate exponents between the images
6. CRT to find actual exponents

Throw caution to the wind; k, ℓ determined experimentally

What is the communication?

Sent TO each process:

p, α, q, and access to the black box

Received FROM each process:

A (coeff, expon, prime) triple for each nonzero term

$c_? \bmod q$	$c_? \bmod q$	$\cdots$
$e_? \bmod p$	$e_? \bmod p$	$\cdots$
p	p	$\cdots$

What is the communication?

Sent TO each process:
p, α, q, and access to the black box

Received FROM each process:
A (coeff, expon, prime) triple for each nonzero term

$c_? \bmod q$	$c_? \bmod q$	$\cdots$
$e_? \bmod p$	$e_? \bmod p$	$\cdots$
p	p	$\cdots$

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 ≤ 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 $f(5z, (5z)^{10}) \bmod (z^7 - 1)$
- $= 3z^6 + 7z^3 + 2z$

An example

1. Parallel evaluation

Process 1 receives:

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

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

coefficient	3	7	2
exponent	6	3	1
prime	7	7	7

An example

1. Parallel evaluation

Process 2 receives:

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

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

coefficient	3	7	2	10	2
exponent	6	3	1	7	4
prime	7	7	7	13	13

An example

1. Parallel evaluation

Process 3 receives:

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

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

coefficient	3	7	2	10	2	2	7	3
exponent	6	3	1	7	4	9	8	3
prime	7	7	7	13	13	17	17	17

An example

2. Recovering f

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

coefficient	3	7	2	10	2	2	7	3
exponent	6	3	1	7	4	9	8	3
prime	7	7	7	13	13	17	17	17

An example

2. Recovering f

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

coefficient	10	7	7	3	3	2	2	2
exponent	7	8	3	3	6	9	4	1
prime	13	17	7	17	7	17	13	7

- Sort the table by coefficients

An example

2. Recovering f

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

coefficient	10	7	7	3	3	2	2	2
exponent	7	8	3	3	6	9	4	1
prime	13	17	7	17	7	17	13	7

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

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

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

An example

2. Recovering f

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

coefficient	10	7	7	3	3	2	2	2
exponent	7	8	3	3	6	9	4	1
prime	13	17	7	17	7	17	13	7

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

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

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

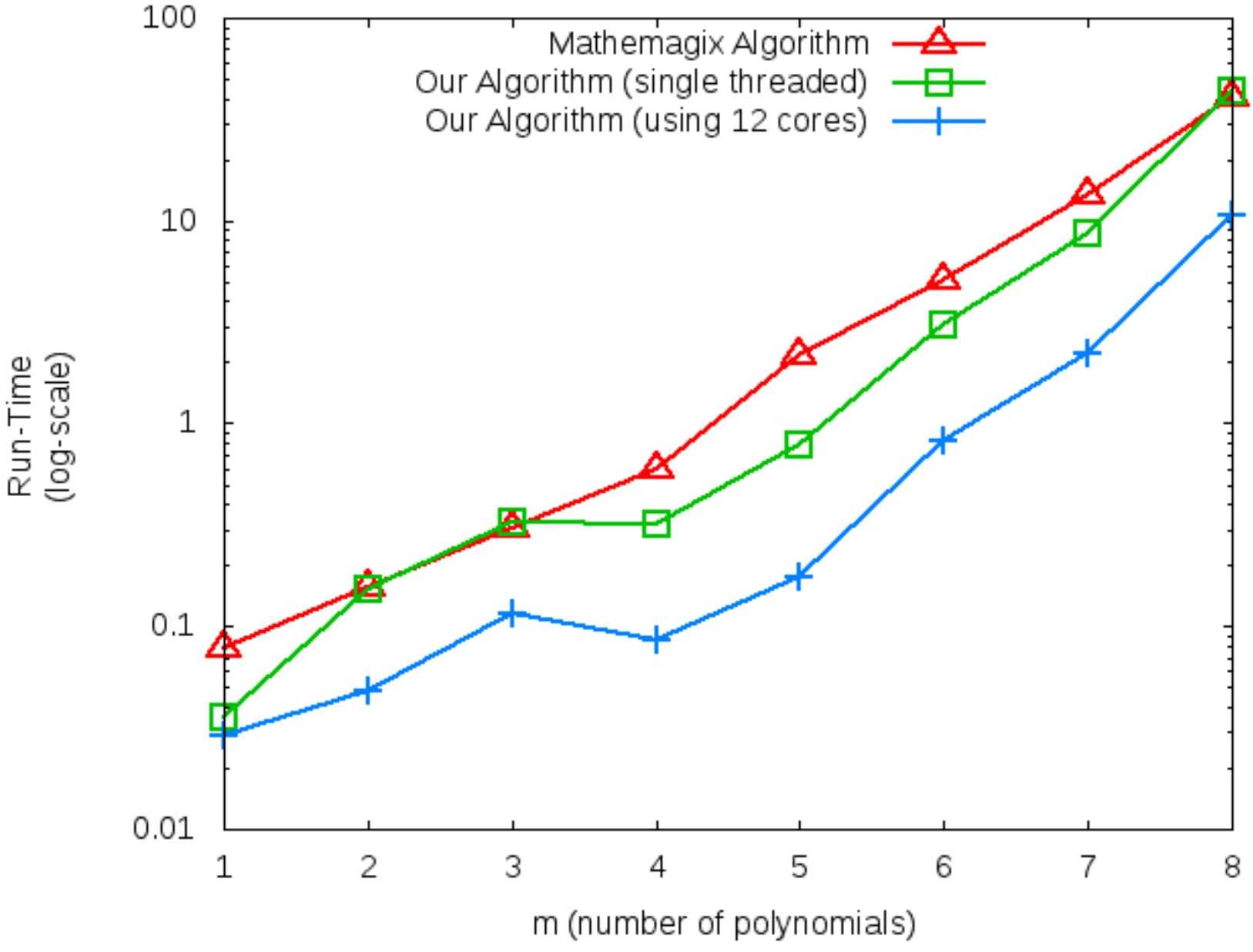
Libraries

- 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**.

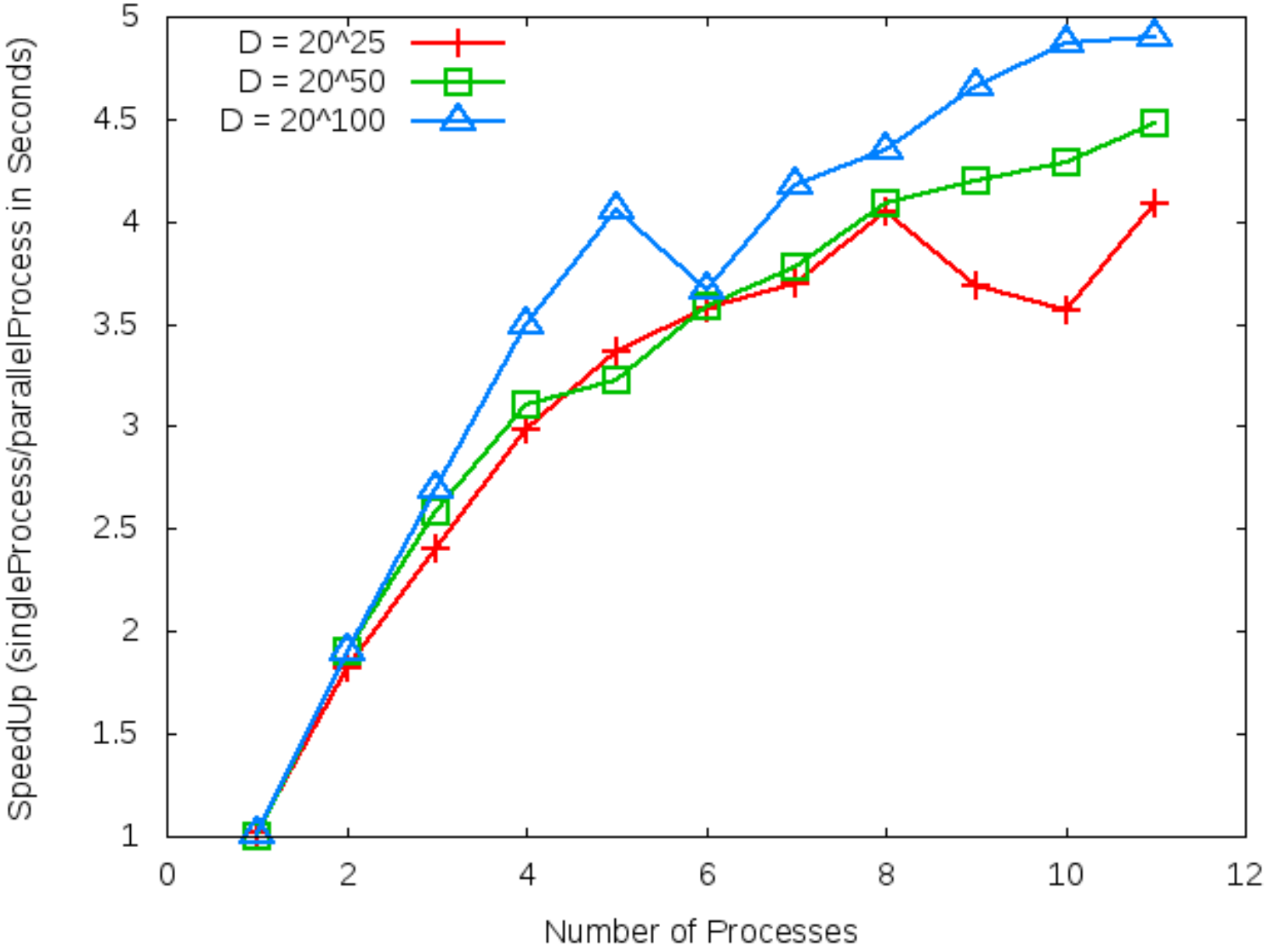

Mathemagix Algorithm
Our Algorithm (single threaded)
Our Algorithm (using 12 cores)
Run-Time (log-scale)
100
10
1
0.1
0.01
1
2
3
4
5
6
7
8
m (number of polynomials)

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.


D = 20^25
D = 20^50
D = 20^100
SpeedUp (singleProcess/parallelProcess in Seconds)
Number of Processes
1
1.5
2
2.5
3
3.5
4
4.5
5
0
2
4
6
8
10
12

To-Do List

- Explore constants k, ℓ more deeply.
  Try to balance larger ℓ, 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

vars	terms	maxdeg	μ	λ	Mathemagix	Ours (single)	Ours (multi)
20	3	40	14	50 000	0.078	0.035	0.029
20	9	80	18	5 000	0.155	0.151	0.048
20	27	120	18	6 900	0.305	0.329	0.116
20	81	160	18	4 900	0.598	0.323	0.085
20	243	200	17	9 900	2.156	0.785	0.175
20	729	240	15	39 900	5.053	3.084	0.814
20	2187	280	14	80 050	13.333	8.714	2.225
20	6561	320	13	321 300	41.070	43.911	10.605

Thank you!