Faster Sparse Interpolation of Straight-Line Programs

Andrew Arnold*, Mark Giesbrecht*, Dan Roche†

*University of Waterloo
†United States Naval Academy

CASC 2013
ZIB, Berlin

Outline

Outline

An overview of straight-line programs

- Straight-line programs
- Statement of the problem
- Summary of results

A new, recursive sparse interpolation algorithm

- "ok" primes - primes that give us information about *some* of the terms of f
- Building an approximation with possible errors
- Recursively interpolating the error

Straight-line programs

A straight-line program (SLP) is a model of algebraic computation. An SLP can represent a polynomial (a **functional representation**).

Definition

Let R be a ring, and let $b_0 \in R$ be a user-specified input. A **straight-line program (SLP)** *is a series instructions $(\Gamma_1, \ldots, \Gamma_\ell)$, where Γ_i assigns a value to $b_i \in R$, and Γ_i is of the form*

$$\Gamma_i : b_i \leftarrow \alpha \ [+ - \times] \ \beta, \qquad \alpha, \beta \in R \cup \{b_0, \ldots, b_{i-1}\}.$$

The vector b is the output corresponding to input b_0. We say ℓ is the **length** *of our program.*

Note

This definition is a restriction to **single-input**, **division-free** SLPs.

Straight-line programs

Example: a SLP for $f(z) = (z^3 + z^2 + z + 1)(z - 1)$

$b_0 = z$ is our input.

$b_1 \leftarrow b_0 \times b_0 \quad [= z^2]$, $b_2 \leftarrow b_0 \times b_1 \quad [= z^3]$,

$b_3 \leftarrow b_2 + b_1 \quad [= z^3 + z^2]$, $b_4 \leftarrow b_3 + b_0 \quad [= z^3 + z^2 + z]$,

$b_4 \leftarrow b_3 + 1 \quad [= z^3 + z^2 + z + 1]$, $b_5 \leftarrow b_0 - 1 \quad [= z - 1]$,

$b_6 \leftarrow b_4 \times b_5 \quad [= (z^3 + z^2 + z + 1)(z - 1)]$.

We say this SLP **computes** f. Interpolating this SLP means expanding

$$(z^3 + z^2 + z + 1)(z - 1) = z^4 - 1.$$

Probing a straight-line program

The interpolation problem

We are given:

1. A SLP that computes $f(z) = \sum_{i=1}^{t} c_i z^{e_i} \in R[z]$, where
   - $e_1 < e_2 < \cdots < e_t = \deg(f)$, and
   - $c_i \neq 0$ for $1 \leq i \leq s$.
2. Upper bounds $D \geq \deg(f)$ and $T \geq t$.

Aim: Construct the terms $c_i z^{e_i}$ of f.

- One can **probe** the SLP, i.e., execute the program on chosen inputs.
- Input: ζ, a symbolic n-th root of 1, for select n.
  - This gives $f(\zeta) \bmod (\zeta^n - 1)$.
  - $n =$ "the **probe degree**".

Note

In the black-box polynomial model, one instead inputs *all* n n-th roots of 1 in order to construct $f(\zeta) \bmod (\zeta^n - 1)$.

Probing a straight-line program

Example: Computing $f(\zeta) \bmod (\zeta^3 - 1)$

$b_0 \leftarrow \zeta,$

$b_1 \leftarrow b_0 \times b_0 \quad [= \zeta^2],$

$b_2 \leftarrow b_0 \times b_1 \quad [= \zeta^3 = 1],$

$b_3 \leftarrow b_2 + b_1 \quad [= 1 + \zeta^2],$

$b_4 \leftarrow b_3 + b_0 \quad [= 1 + \zeta + \zeta^2],$

$b_4 \leftarrow b_3 + 1 \quad [= 2 + \zeta + \zeta^2],$

$b_5 \leftarrow b_0 - 1 \quad [= \zeta - 1],$

$b_6 \leftarrow b_4 \times b_5 \quad [= (2 + \zeta + \zeta^2)(\zeta - 1) = \zeta - 1].$

Each SLP instruction entails adding/multiplying two polynomials modulo $(\zeta^n - 1)$

Cost of one SLP instruction: $\widetilde{\mathcal{O}}(n)$

Cost of probe: $\widetilde{\mathcal{O}}(\ell n)$

Probing a straight-line program

Example: Computing $f(\zeta) \bmod (\zeta^3 - 1)$

$b_0 \leftarrow \zeta$,

$b_1 \leftarrow b_0 \times b_0 \quad [= \zeta^2]$, $\qquad b_2 \leftarrow b_0 \times b_1 \quad [= \zeta^3 = 1]$,

$b_3 \leftarrow b_2 + b_1 \quad [= 1 + \zeta^2]$, $\qquad b_4 \leftarrow b_3 + b_0 \quad [= 1 + \zeta + \zeta^2]$,

$b_4 \leftarrow b_3 + 1 \quad [= 2 + \zeta + \zeta^2]$, $\qquad b_5 \leftarrow b_0 - 1 \quad [= \zeta - 1]$,

$b_6 \leftarrow b_4 \times b_5 \quad [= (2 + \zeta + \zeta^2)(\zeta - 1) = \zeta - 1]$.

Each SLP instruction entails adding/multiplying two polynomials modulo $(\zeta^n - 1)$
Cost of one SLP instruction: $\widetilde{\mathcal{O}}(n)$
Cost of probe: $\widetilde{\mathcal{O}}(\ell n)$

Measure of cost of interpolation

cost $\approx$ (# of probes)$*$(probe degree)

Results of Monte Carlo interpolation algorithms

algorithm	# of probes	probe degree	"cost"
Garg and Schost[1]	$\widetilde{\mathcal{O}}(T \log D)$	$\mathcal{O}(T^2 \log D)$	$\widetilde{\mathcal{O}}(T^3 \log^2 D)$
Giesbrecht and Roche[2]	$\widetilde{\mathcal{O}}(\log D)$	$\mathcal{O}(T^2 \log D)$	$\widetilde{\mathcal{O}}(T^2 \log^2 D)$
AA, Giesbrecht, Roche	$\widetilde{\mathcal{O}}(\log T \log D)$	$\mathcal{O}(T \log^2 D)$	$\widetilde{\mathcal{O}}(T \log^3 D)$

These algorithms admit a failure probability of $\leq \epsilon$, for a fixed parameter ϵ.

Recall

$f(z)$ is in $\widetilde{\mathcal{O}}(g(z))$ if $f \in \mathcal{O}\left(g(z) \log(g(z))^c\right)$ for some constant c.

1. *Interpolation of polynomials given by straight-line programs. Theoretical Comp. Sci. 2009.*

2. *Diversification improves interpolation. ISSAC 2011.*

From Monte Carlo to Las Vegas

These algorithms are Monte Carlo (probably correct, deterministic run-time). To verify an output f^* we can use the following:

Lemma (Blaser et al., 2009)

Let $g = f - f^*$ be a polynomial over an integral domain R. Suppose

1. *We know g has at most T terms and degree at most D.*
2. $g \bmod (z^p - 1) = 0$ *for some* $(T-1)\log_2 D$ *primes p.*

Then $g = 0$.

- $\#$ of probes: $\mathcal{O}(T \log D)$
- probe degree: $\widetilde{\mathcal{O}}(T \log D)$
- **cost**: $\widetilde{\mathcal{O}}(T^2 \log^2 D)$

Note

This test is at least as fast as the Garg-Schost and Giesbrecht-Roche algorithms; however, the test is costlier than the new algorithm when $T \in o(\log D)$.

Outline

An overview of straight-line programs
- Straight-line programs
- Statement of the problem
- Summary of results

A new, recursive sparse interpolation algorithm
- "ok" primes - primes that give us information about *some* of the terms of f
- Building an approximation with possible errors
- Recursively interpolating the error

A new, recursive sparse interpolation algorithm

Throughout the algorithm, we build a sparse polynomial f^* (initially zero) approximating the f given by our input SLP. In each recursive step of the algorithm we will refine f^*, until $f - f^* = 0$.

Refining our approximation f^*

We interpolate the difference $g = f - f^*$ as follows:

1. With high probability, find an "ok" prime, a prime p for which at most a small, specified proportion of the terms of g collide modulo $(z^p - 1)$.
2. Given p, look at images of the form $g(z) \bmod (z^{pq} - 1)$, in order to construct a sparse polynomial f^{**} such that $g - f^{**}$ has at most $T/2$ terms.
3. Set $f^* \leftarrow f^* + f^{**}$ and $T \leftarrow \lfloor T/2 \rfloor$ and repeat. If $T = 0$, return f^*.

good primes

- We say two terms $c_1 z^{e_1}$ and $c_2 z^{e_2}$ **collide** modulo $(z^p - 1)$ if $p \mid (e_1 - e_2)$.
- Both the Garg-Schost and Giesbrecht-Roche algorithms require a **good prime** p for which no terms of f collide modulo $(z^p - 1)$.

Example of a good prime

Let $f(z) = 1 + z + 3z^{10}$. Then

$$f(z) \bmod (z^5 - 1) = 4 + z,$$
$$f(z) \bmod (z^7 - 1) = 1 + z + 3z^3.$$

- 5 is **not** a good prime as 1 collides with $3z^{10}$ modulo $(z^5 - 1)$.
- $f(z) \bmod (z^7 - 1)$ has three terms (like f), so 7 is a good prime.

ok primes

- An ok prime is a weaker notion of a good prime, that allows for a small number of terms to collide.
- Ok primes will allow us to use primes of size $\mathcal{O}(T \log D)$ instead of $\mathcal{O}(T^2 \log D)$.

Definition

- *Given a polynomial g with $\leq T$ terms, and a prime p, we let $\mathcal{C}_g(p)$ denote the number of terms of f that are involved in collisions modulo $(z^p - 1)$.*
- *We call p an* **ok prime** *if $\mathcal{C}_g(p) < \frac{3}{8}T$.*
- *p is a* **good prime** *if $\mathcal{C}_g(p) = 0$.*

ok primes - an example

Let $g = 1 + z + z^4 - 2z^{13}$.

$p = 2$

$$1 + z + z^4 - 2z^{13} \bmod (z^2 - 1) = 1 + z + 1 - 2z,$$
$$= 2 - z.$$

z^4 collides with 1 and $-2z^{13}$ collides with z, so $\mathcal{C}_g(2) = 4$.

$p = 3$

$$1 + z + z^4 - 2z^{13} \bmod (z^3 - 1) = 1 + z + z - 2z,$$
$$= 1.$$

z, z^4, and $-2z^{13}$ collide, so $\mathcal{C}_g(3) = 3$.

"ok" primes

Lemma

Let $g(z)$ have degree $\leq D$ and $\leq T$ terms, and let

$$\lambda = \max\left(21, \left\lceil \frac{160}{9}(T-1)\ln D \right\rceil\right) \in \mathcal{O}(T \log D).$$

Then p, a prime chosen at random in the range $[\lambda, 2\lambda]$ satisfies $\mathcal{C}_g(p) < \mathbf{\frac{3}{16}T}$ with probability at least $\frac{1}{2}$.

Thus, if we look at $f(z) \bmod (z^p - 1)$ for some $\lceil \log 1/\epsilon \rceil$ primes $p \in [\lambda, 2\lambda]$, we will have come across an ok prime p with probablity $\geq 1 - \epsilon$.

Cost to search for an ok prime

- **Probe degree**: $p \in \mathcal{O}(T \log D)$.
- **# of probes**: $\mathcal{O}(\log 1/\epsilon)$.
- **Cost**: $\mathcal{O}(T \log D \log 1/\epsilon)$.

"ok" primes

Problem

How do we know the p which minimizes $\mathcal{C}_g(p)$?

"ok" primes

Problem

How do we know the p which minimizes $\mathcal{C}_g(p)$?

Lemma

Suppose $g \bmod (z^q - 1)$ *has sparsity* s_q *and* $g \bmod (z^p - 1)$ *has sparsity* $s_p \geq s_q$. *Then* $\mathcal{C}_g(p) \leq 2\mathcal{C}_g(q)$.

Thus choosing the p for which the image $g(z) \bmod (z^p - 1)$ has the **most terms** gives us $\mathcal{C}_g(p) < 2\frac{3}{16}T = \frac{3}{8}T$ (with probability $1 - \epsilon$).

Constructing images of $g = f - f^*$

Given an ok prime p, we then construct images $g(z) \bmod (z^{pq_i} - 1)$ for a set of coprime $q_i > 1$ chosen as follows. Let

$$x = \max(2\ln(D), 17), \text{ and}$$
$$k = \lceil x/\ln(x) \rceil \in \mathcal{O}\left(\frac{\log D}{\log\log D}\right).$$

Then, for $1 \leq i \leq k$, we set

$$q_i \longleftarrow \text{“greatest power of the } i\text{-th prime not exceeding } x\text{”}.$$

Note that $q_i \in \mathcal{O}(\log D)$ and $\prod_{i=1}^{k} q_i > D$.

Cost of probes used to build f^{**}

- **Probe degree**: $pq_i \in \mathcal{O}(T \log^2 D)$.
- **# of probes**: $k \in \widetilde{\mathcal{O}}(\log D)$.
- **Cost**: $\widetilde{\mathcal{O}}(T \log^3 D)$.

Building f^{**} approximating $g = f - f^*$

- If a term cz^e of g avoids collision modulo $(z^p - 1)$, then in any image $g \bmod (z^{pq} - 1)$ the term cz^e will appear as a unique term of degree congruent to $e \bmod p$.

- Conversely, suppose there exists $c_0 z^{e_0}$, where $e_0 \leq D$, satisfying
  1. There is a unique term in the reduced image $g \bmod (z^p - 1)$ that *may* be an image of $c_0 z^{e_0}$.
  2. There is a unique term in each image $g \bmod (z^{pq_i} - 1)$ that *may* be an image of $c_0 z^{e_0}$,

  ...then $c_0 z^{e_0}$ may be a term of g, and we add it to f^{**}.

- $c_0 z^{e_0}$ is not always a term of g. If not we call $c_0 z^{e_0}$ a **deceptive term.**

How to build f^{**} approximating g

Example of a deceptive term

Let

$$g(z) = z^2 + z^{40} + z^{60} + \mathbf{z}^{\mathbf{11+4}} - \mathbf{z}^{\mathbf{14\cdot 11+4}} - \mathbf{z}^{\mathbf{15\cdot 11+4}},$$

and let $p = 11$ and consider $q = 2, 3, 5$. We have

$$g(z) \bmod (z^{11} - 1) = z^2 + z^7 + z^5 - \mathbf{z}^{\mathbf{4}},$$
$$g(z) \bmod (z^{22} - 1) = z^2 + z^{18} + z^{16} - \mathbf{z}^{\mathbf{15}},$$
$$g(z) \bmod (z^{33} - 1) = z^2 + z^7 + z^{27} - \mathbf{z}^{\mathbf{26}},$$
$$g(z) \bmod (z^{55} - 1) = z^2 + z^{40} + z^5 - \mathbf{z}^{\mathbf{48}}.$$

- The first three terms of each image correspond to the terms z^2, z^{40}, z^{60} appearing in g. Note there is only one term of degree $\{2, 40, 60\}$ mod 11 in each image.
- The remaining term in each of the 4 images has degree congruent to 4 mod 11. By Chinese remaindering on the exponents, this gives a deceptive term $-\mathbf{z}^{\mathbf{323}}$ not appearing in g.

Detecting collisions

- A deceptive term can only result from a collision of **three or more** terms modulo $(z^p - 1)$.
- If only two terms collide modulo $z^p - 1$, there will be a q_i such that $g \bmod (z^{pq_i} - 1)$ separates those terms.

Example: $g(z) = 1 + z^{59} + z^{11+2} + z^{11\cdot 12+11+2}$

Using $p = 11$ and $(q_1, q_2, q_3) = (4, 3, 5)$, we have

$$g(z) \bmod (z^{11} - 1) = 1 + 2z^2 + z^4$$
$$g(z) \bmod (z^{44} - 1) = 1 + z^{15} + 2z^{13}$$
$$g(z) \bmod (z^{33} - 1) = 1 + z^{26} + 2z^{13}$$
$$g(z) \bmod (z^{55} - 1) = 1 + z^4 + z^{13} + z^{35}$$

We recognize that a collision occurred at degree 2 modulo $(z^{11} - 1)$ and ignore those terms. Here f^{**} would be $1 + z^{59}$.

Building an approximation f^{**} of $g = f - f^{**}$

- The polynomial f^{**} we construct will contain the $T - \mathcal{C}_g(p)$ non-colliding terms of g, plus potentially some $\lfloor \mathcal{C}_g(p)/3 \rfloor$ deceptive terms.
- $g - f^{**}$ will have at most $\mathcal{C}_g(p) + \frac{1}{3}\mathcal{C}_g(p) = \frac{4}{3}\mathcal{C}_g(p)$ terms.
- If $\mathcal{C}_g(p) < \frac{3}{8}T$, then $g - f^{**}$ will have sparsity less than $(\frac{4}{3})(\frac{3}{8})T = T/2$.

Recursively interpolating $g - f^{**}$

- Given a sparsity bound T for $g = f - f^*$, the approximation f^{**} gives us a new difference $f - f^* - f^{**}$ with smaller sparsity bound $T/2$.

- We set

$$f^* \leftarrow f^* + f^{**}, \text{ and}$$
$$T \leftarrow \lfloor T/2 \rfloor,$$

  then recursively interpolate our now updated difference $g = f - f^*$.

- We continue in this fashion some $\lfloor 1 + \log T \rfloor$ times until we have $f - f^* = 0$.

Problem:

At the start of our algorithm, we look at $\lceil \log 1/\epsilon \rceil$ primes p_i in an attempt to find an ok prime with probability $1 - \epsilon$. We now need this to succeed at each of the $\lfloor 1 + \log T \rfloor$ recursive calls now.

If we want to correctly interpolate f with probability μ, it suffices to instead bound the failure probability at each recursive step by $\epsilon = \mu/(1 + \log T)$. This does not affect the "soft-Oh" cost of the algorithm.

Cost with probability of success at least $1 - \mu$, for fixed μ

Interpolating f entails $\widetilde{\mathcal{O}}(\log T \log D)$ probes of degree $\mathcal{O}(T \log^2 D)$.
Cost: $\widetilde{\mathcal{O}}(T \log^3 D)$

- Final thoughts:
  - Difficult to guarantee zero collisions for small probe degree - birthday paradox.
  - ok primes: better performance by tolerating *some* errors.
  - An advantage of having a recursive algorithm decrementing T is that we can call the Giesbrecht-Roche $T^2 \log^2 D$ algorithm once $\log D \gg T$.
- Future work:
  - Investigate the numerical stability of a black-box variant of the algorithm.
  - Las Vegas algorithm: faster polynomial identity testing of SLPs.
  - $\mathcal{O}(T \log D)$ interpolation?

Thank you for your attention