

Faster sparse interpolation of straight-line programs*

Andrew Arnold¹, Mark Giesbrecht¹, and Daniel S. Roche^{†2}

¹Cheriton School of Computer Science, University of Waterloo

²Computer Science Department, United States Naval Academy

June 29, 2018

Abstract

We give a new probabilistic algorithm for interpolating a “sparse” polynomial f given by a straight-line program. Our algorithm constructs an approximation f^* of f , such that $f - f^*$ probably has at most half the number of terms of f , then recurses on the difference $f - f^*$. Our approach builds on previous work by Garg and Schost (2009), and Giesbrecht and Roche (2011), and is asymptotically more efficient in terms of the total cost of the probes required than previous methods, in many cases.

1 Introduction

We consider the problem of interpolating a sparse, univariate polynomial

$$f = c_1 z^{e_1} + c_2 z^{e_2} + \cdots + c_t z^{e_t} \in \mathcal{R}[z]$$

of degree d with t non-zero coefficients c_1, \dots, c_t (where t is called the *sparsity* of f) over a ring \mathcal{R} . More formally, we are given a *straight-line program* that evaluates f at any point, as well as bounds $D \geq d$ and $T \geq t$. The straight-line program is a simple but useful abstraction of a computer program without branches, but our interpolation algorithm will work in more common settings of “black box” sampling of f .

We summarize our final result as follows.

Theorem 1. *Let $f \in \mathcal{R}[z]$, where \mathcal{R} is any ring. Given any straight-line program of length L that computes f , and bounds T and D for the sparsity and degree of f , one can find all coefficients and exponents of f using $\mathcal{O}^{\sim}(LT \log^3 D + LT \log D \log(1/\mu))^{\S}$ ring operations in \mathcal{R} , plus a similar number of bit operations. The algorithm is probabilistic of the Monte Carlo type: it can generate*

*A version of this paper appeared at CASC 2013, doi:10.1007/978-3-319-02297-0_5

[†]Supported by NSF Award #1319994

random bits at unit cost and on any invocation returns the correct answer with probability greater than $1 - \mu$, for a user-supplied tolerance $\mu > 0$.

1.1 The straight-line program model and interpolation

Straight-line programs are a useful model of computation, both as a theoretical construct and from a more practical point of view; see, e.g., (Bürgisser et al., 1997, Chapter 4). Our interpolation algorithms work more generally for N -variate sparse polynomials $f \in \mathcal{R}[z_1, \dots, z_N]$ given by a straight-line program \mathcal{S}_f defined as follows. \mathcal{S}_f takes an input $(a_1, \dots, a_N) \in \mathcal{R}^N$ of length N , and produces a vector $b \in \mathcal{R}^L$ via a series of L instructions $\Gamma_i : 1 \leq i \leq L$ of the form

$$\Gamma_i = \begin{cases} \gamma_i \leftarrow \alpha_1 \star \alpha_2, & \text{or} \\ \gamma_i \leftarrow \delta \in \mathcal{R} & (\text{i.e., a constant from } \mathcal{R}), \end{cases}$$

where \star is a ring operation ‘+’, ‘−’, or ‘ \times ’, and either $\alpha_\ell \in \{a_j\}_{1 \leq j \leq n}$ or $\alpha_\ell \in \{\gamma_k\}_{1 \leq k < i}$ for $\ell = 1, 2$. When we say \mathcal{S}_f computes f , we mean \mathcal{S}_f sets γ_L to $f(a_1, \dots, a_N) \in \mathcal{R}$.

To interpolate an N -variate polynomial $f \in \mathcal{R}[z_1, \dots, z_N]$, we apply a Kronecker substitution, and interpolate

$$\hat{f}(z) = f\left(z, z^{(D+1)}, z^{(D+1)^2}, \dots, z^{(D+1)^{N-1}}\right) \in \mathcal{R}[z].$$

While this certainly increases the degree, f and \hat{f} have the same number of non-zero terms, and f can be easily recovered from \hat{f} . This reduces the problem of interpolating the N -variate polynomial f of partial degree at most D to interpolating a univariate polynomial \hat{f} of degree at most $(D+1)^N$. For the remainder of this paper we thus assume f is univariate.

It will also be necessary to evaluate our polynomial $f \in \mathcal{R}[z]$, or rather our straight-line program \mathcal{S}_f for f , in an extension ring of \mathcal{R} . Precisely, we want to evaluate f at symbolic ℓ th roots of unity for various choices of ℓ , or algebraically, in $\mathcal{R}[z]/(z^\ell - 1)$. This may be regarded as transforming our straight-line program by substituting operations in \mathcal{R} with operations in $\mathcal{R}[z]/(z^\ell - 1)$, where each element is represented by a polynomial in $\mathcal{R}[z]$ of degree less than ℓ . Each instruction Υ_i in the transformed branching program now potentially requires $M(\ell)$ operations in \mathcal{R} , where $M(\ell)$ is the number of operations in \mathcal{R} and bit operations needed to multiply two degree- ℓ polynomials over the base ring \mathcal{R} . By Cantor and Kaltofen (1991), we may assume $M(\ell) = \mathcal{O}(\ell \log \ell \log \log \ell)$.

Each evaluation of our straight-line program for f in $\mathcal{R}[z]/(z^\ell - 1)$ is called a *probe of degree ℓ* . Thus, the cost of a degree- ℓ probe to \mathcal{S}_f is $\mathcal{O}(L\ell)$ operations in \mathcal{R} , and similarly many bit operations.

This is easily connected to the more “classical” view of sparse interpolation, in which probes are simply evaluations of a “black-box” polynomial at a single

[§]For summary convenience we use soft-Oh notation: for functions $\phi, \psi \in \mathbb{R}_{>0} \rightarrow \mathbb{R}_{>0}$ we say $\phi \in \mathcal{O}^\sim(\psi)$ if and only if $\phi \in \mathcal{O}(\psi(\log \psi)^c)$ for some constant $c \geq 0$.

point (and we do not have any representation for how f is calculated). Each probe in the straight-line program model can be thought of as evaluating f at *all* ℓ th roots of unity in the classical model. Since we charge $M(\ell) = \mathcal{O}(\ell)$ operations in \mathcal{R} for a degree ℓ probe in the straight-line program model, i.e., about ℓ times as much as a single black-box probe, this is consistent with the costs in a classical model. We note that algorithms for sparse interpolation presented below could be stated in this classical model, though we find the straight-line program model convenient and will continue with it throughout this paper.

1.2 Previous work

Straight-line programs, or equivalently algebraic circuits, are important both as a computational model and as a data structure for polynomial computation. Their rich history includes both algorithmic advances and practical implementations (Kaltofen, 1989; Sturttivant and Zhang, 1990; Bruno et al., 2002).

One can naively interpolate a polynomial $f \in \mathcal{R}[z]$ given by a straight-line program using a dense method, with D probes of degree 1. Prony’s (1795) interpolation algorithm — see (Ben-Or and Tiwari, 1988; Kaltofen et al., 1990; Giesbrecht et al., 2009) — is a sparse interpolation method that uses evaluations at only $2T$ powers of a root of unity whose order is greater than D . However, in the straight-line program model for a general ring, this would require evaluating at a symbolic D th root of unity, which would use at least $\Omega(D)$ ring operations and defeat the benefit of sparsity. Problems with Prony’s algorithm are also seen in the classical model in that the underlying base ring \mathcal{R} must also support an efficient discrete logarithm algorithm on entries of high multiplicative order (which, for example, is not feasible over large finite fields).

We mention two algorithms specifically intended for straight-line programs.

1.2.1 The Garg-Schost deterministic algorithm.

Garg and Schost (2009) describe a novel deterministic algorithm for interpolating a multivariate polynomial f given by a straight-line program. Their algorithm entails constructing an integer symmetric polynomial with roots at the exponents of f :

$$\chi = \prod_{i=1}^t (y - e_i) \in \mathbb{Z}[y],$$

which is then factored to obtain the exponents e_i .

Their algorithm first finds a *good* prime: a prime p for which the terms of f remain distinct when reduced modulo $z^p - 1$. We call such an image $f \bmod (z^p - 1)$ a *good image*. Such an image gives us the values $e_i \bmod p$ and hence $\chi(y) \bmod p$.

Example 2. For $f = z^{33} + z^3$, 5 is not a good prime because $f \bmod (z^5 - 1) = 2z^3$. We say z^{33} and z^3 collide modulo $z^5 - 1$. 7 is a good prime, as the image $f(z) \bmod (z^7 - 1) = z^5 + z^3$ has as many terms as $f(z)$ does.

In order to guarantee that we have a good prime, the algorithm requires that we construct the images $f \bmod (z^p - 1)$ for the first N primes, where N is roughly $\mathcal{O}^\sim(T^2 \log D)$. A good prime will be a prime p for which the image $f \bmod (z^p - 1)$ has maximally many terms, which will be exactly t . Once we know we have a good image we can discard the images $f \bmod (z^q - 1)$ for *bad* primes q , i.e. images with fewer than t terms. We use the remaining images to construct $\chi(y) = \prod_{i=1}^t (y - e_i) \in \mathbb{Z}[y]$ by way of Chinese remaindering on the images $\chi(y) \bmod p$.

We factor $\chi(y)$ to obtain the exponents e_i , after which we directly obtain the corresponding coefficients c_i directly from a good image.

The algorithm of [Garg and Schost \(2009\)](#) can be made faster, albeit Monte Carlo, using the following number-theoretic fact.

Fact 3 ([Giesbrecht and Roche, 2011](#)). *Let $f \in \mathcal{R}[z]$ be a polynomial with at most T terms and degree at most D . Let $\lambda = \max(21, \lceil \frac{5}{3}T(T-1) \log D \rceil)$. A prime p chosen at random in the range $[\lambda, 2\lambda]$ is a good prime for $f(z)$ with probability at least $\frac{1}{2}$.*

Thus, in order to find a good image with probability at least $1 - \varepsilon$, we can inspect images $f \bmod (z^p - 1)$ for $\lceil \log 1/\varepsilon \rceil$ primes p chosen at random in $[\lambda, 2\lambda]$. As the height of $\chi(y)$ can be roughly as large as D^T , we still require some $\mathcal{O}^\sim(T \log D)$ probes to construct $\chi(y)$.

1.2.2 The “diversified” interpolation algorithm.

[Giesbrecht and Roche \(2011\)](#) obtain better performance by way of *diversification*. A polynomial f is said to be *diverse* if its coefficients c_i are pairwise distinct. The authors show that, for f over a finite field or \mathbb{C} and for appropriate random choices of α , $f(\alpha z)$ is diverse with probability at least $\frac{1}{2}$. They then try to interpolate the diversified polynomial $f(\alpha z)$.

Once we have t with high probability, we look at images $f(\alpha z) \bmod (z^p - 1)$ for primes p in $[\lambda, 2\lambda]$, discarding bad images. As $f(\alpha z)$ is diverse, we can recognize which terms in different good images are images of the same term. Thus, as all the e_i are at most D , we can get all the exponents e_i by looking at some $\mathcal{O}(\log D)$ good images of f .

1.3 Deterministic zero testing

Both the Monte Carlo algorithms of [Garg and Schost \(2009\)](#) and [Giesbrecht and Roche \(2011\)](#) can be made Las Vegas (i.e., no possibility of erroneous output, but unbounded worst-case running time) by way of deterministic zero-testing. Given a polynomial f represented by a straight-line program, each of these algorithms produces a polynomial f^* that is probably f .

Fact 4 ([Bläser et al. \(2009\)](#); Lemma 13). *Let \mathcal{R} be an integral domain, and suppose $f = f^* \bmod (z^p - 1)$ for $T \log D$ primes. Then $f = f^*$.*

Table 1: A “soft-Oh” comparison of interpolation algorithms for straight-line programs

	Probes	Probe degree	Cost of probes	Type
Dense	D	1	LD	deterministic
Garg & Schost	$T^2 \log D$	$T^2 \log D$	$LT^3 \log^2 D$	deterministic
*Las Vegas G & S	$T \log D$	$T^2 \log D$	$LT^3 \log^2 D$	Las Vegas
*Diversified	$\log D$	$T^2 \log D$	$LT^2 \log^2 D$	Las Vegas
†Recursive	$\log T \log D$	$T \log^2 D$	$LT \log^3 D$	Monte Carlo

*Average # of probes given; † for a fixed probability of failure μ

Thus, testing the correctness of the output of a Monte Carlo algorithm requires some $\mathcal{O}(T \log D)$ probes of degree at most $\mathcal{O}(T \log D)$. This cost does not dominate the cost of either Monte Carlo algorithm. We note that this deterministic zero test can dominate the cost of the interpolation algorithm presented in this paper if T is asymptotically dominated by $\log D$.

1.4 Summary of results

We state as a theorem the number and degree of probes required by our new algorithm presented in this paper.

Theorem 5. *Let $f \in \mathcal{R}[z]$, where \mathcal{R} is a ring. Given a straight-line program for f , one can find all coefficients and exponents of f with probability at least $1 - \mu$ using $\mathcal{O}(\log T(\log D + \log \frac{1}{\mu}))$ probes of degree at most $\mathcal{O}(T \log^2 D)$.*

Table 1 gives a rough comparison of known algorithms. Our recursive algorithm improves by a factor of $T/\log D$ over the Giesbrecht-Roche diversification algorithm — ignoring “soft” multiplicative factors of $(\log(T/\log D))^{O(1)}$ — and as such is better suited for moderate values of T . Our algorithm recursively interpolates a series of polynomials of decreasing sparsity. An advantage of this method is that, when we cross a threshold where $\log D$ begins to dominate T , we can merely call the Monte Carlo diversification algorithm instead.

2 A recursive algorithm for interpolating f

Entering each recursive step in our algorithm we have our polynomial f represented by a straight-line program, and an explicit sparse polynomial f^* “approximating” f , that is, whose terms mostly appear in the sparse representation of f . At each recursive step we try to interpolate the difference $g = f - f^*$. To begin with, f^* is initialized to zero.

We first find an “ok” prime p which separates most of the terms of g . We then use that prime p to build a approximation f^{**} , containing most of the terms of g , plus possibly some additional “deceptive” terms. The polynomial

f^{**} is constructed such that $g = f - f^*$ has, with high probability, at most $T/2$ terms. We then recursively interpolate the difference $g - f^{**}$.

Producing images $f^* \bmod (z^\ell - 1)$ is straightforward, we merely reduce the exponents of terms of f^* modulo ℓ . We assume g has a sparsity bound $T_g \leq T$.

2.1 A weaker notion of “good” primes

To interpolate a polynomial g , the sparse interpolation algorithm described by [Giesbrecht and Roche \(2011\)](#) requires a *good* prime p which keeps the exponents of g distinct modulo p . That is, $g \bmod (z^p - 1)$ has the same number of terms as g . We define a weaker notion of a good prime, an *ok prime*, which separates most of the terms of g . To that end we measure, for fixed g and prime p , how well p separates the terms of g .

Definition 6. Fix a polynomial $g = \sum_{i=1}^t c_i z^{e_i} \in \mathcal{R}[z]$ with non-zero $c_1, \dots, c_t \in \mathcal{R}$, where $e_i < e_j$ for $i < j$, we say $c_i z^{e_i}$ and $c_j z^{e_j}$, $i \neq j$, collide modulo $z^p - 1$ if $e_i \equiv e_j \pmod{p}$. We call any term $c_i z^{e_i}$ of f which collides with any other term of f a colliding term of f modulo $z^p - 1$. We let $\mathcal{C}_g(p) \in [0, t]$ denote the number of colliding terms of g modulo $z^p - 1$.

Example 7. For the polynomial $g = 1 + z^5 + z^7 + z^{10}$, $\mathcal{C}_g(2) = 4$, since 1 collides with z^{10} and z^5 collides with z^7 modulo $z^2 - 1$. Similarly, $\mathcal{C}_g(5) = 2$, since z^5 collides with z^{10} modulo $z^5 - 1$.

We say $c_i z^{e_i}$ and $c_j z^{e_j}$ collide modulo $z^p - 1$ because both terms have the same exponent once reduced modulo $z^p - 1$. All other terms of g we will call *non-colliding* terms modulo $z^p - 1$.

In the sparse interpolation algorithm of [Giesbrecht and Roche \(2011\)](#), one chooses a $\lambda \in \mathbb{Z}_{>0}$ such that the probability of a prime $p \in [\lambda, 2\lambda]$, chosen at random and having $\mathcal{C}_g(p) = 0$, is at least $\frac{1}{2}$. However, in order to guarantee that we find such a prime with high probability, we need to choose $\lambda \in \mathcal{O}(T^2 \log D)$.

In this paper we will search over a range of smaller primes, while allowing for a reasonable number of collisions. We try to pick λ such that

$$\Pr(\mathcal{C}_g(p) \geq \gamma \text{ for a random prime } p \in [\lambda, 2\lambda]) < 1/2,$$

for a parameter γ to be determined.

Lemma 8. Let $g \in \mathcal{R}[z]$ be a polynomial with $t \leq T$ terms and degree at most $d \leq D$. Suppose we are given T and D , and let $\lambda = \max\left(21, \left\lceil \frac{10T(T-1)\ln(D)}{3\gamma} \right\rceil\right)$. Let p be a prime chosen at random in the range $\lambda, \dots, 2\lambda$. Then $\mathcal{C}_g(p) \geq \gamma$ with probability less than $\frac{1}{2}$.

Proof. The proof follows similarly to the proof of Lemma 2.1 in [\(Giesbrecht and Roche, 2011\)](#).

Let B be the set of unfavourable primes for which $\mathcal{C}_g(p) \geq \gamma$ terms collide modulo $z^p - 1$, and denote the size of B by $\#B$. As every colliding term

collides with at least one other term modulo $z^p - 1$, we know $p^{C_g(p)}$ divides $\prod_{1 \leq i \neq j \leq t} (e_i - e_j)$. Thus, as $C_g(p) \geq \gamma$ for $p \in B$,

$$\lambda^{\#B\gamma} \leq \prod_{p \in B} p^\gamma \leq \prod_{1 \leq i \neq j \leq t} (e_i - e_j) \leq d^{t(t-1)} \leq D^{T(T-1)}.$$

Solving the inequality for $\#B$ gives us

$$\#B \leq \frac{T(T-1)\ln(D)}{\ln(\lambda)\gamma}.$$

The total number of primes in $[\lambda, 2\lambda]$ is greater than $3\lambda/(5\ln(\lambda))$ for $\lambda \geq 21$ by Corollary 3 to Theorem 2 of (Rosser and Schoenfeld, 1962). From our definition of λ we have

$$\frac{3\lambda}{5\ln(\lambda)} > \frac{2T(T-1)\ln(D)}{\ln(\lambda)\gamma} \geq 2\#B,$$

completing the proof. \square

\square

2.1.1 Relating the sparsity of $g \bmod (z^p - 1)$ with $C_g(p)$

Suppose we choose λ according to Lemma 8, and make k probes to compute $g \bmod (z^{p_1} - 1), \dots, g \bmod (z^{p_k} - 1)$. One of the primes p_i will yield an image with fewer than γ colliding terms (i.e. $C_g(p_i) < \gamma$) with probability at least $1 - 2^{-k}$. Unfortunately, we do not know which prime p maximizes $C_g(p)$. A good heuristic might be to select the prime p for which $g \bmod (z^p - 1)$ has maximally many terms. However, this does not necessarily minimize $C_g(p)$. Consider the following example.

Example 9. *Let*

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

We have

$$g \bmod (z^2 - 1) = 2 - z, \quad \text{and} \quad g \bmod (z^3 - 1) = 1.$$

While $g \bmod (z^2 - 1)$ has more terms than $g \bmod (z^3 - 1)$, we see that $C_g(2) = 4$ is larger than $C_g(3) = 3$.

While we cannot determine the prime p for which $g \bmod (z^p - 1)$ has maximally many non-colliding terms, we show that choosing the prime p which maximizes the number of terms in $g \bmod (z^p - 1)$ is, in fact, a reasonable strategy.

We would like to find a precise relationship between $C_g(p)$, the number of terms of g that collide in the image $g \bmod (z^p - 1)$, and the sparsity s of $g \bmod (z^p - 1)$.

Lemma 10. *Suppose that g has t terms, and $g \bmod (z^p - 1)$ has $s \leq t$ terms. Then $t - s \leq C_g(p) \leq 2(t - s)$.*

Proof. To prove the lower bound, note that $t - \mathcal{C}_g(p)$ terms of g will not collide modulo $z^p - 1$, and so $g \bmod (z^p - 1)$ has sparsity s at least $t - \mathcal{C}_g(p)$.

We now prove the upper bound. Towards a contradiction, suppose that $\mathcal{C}_g(p) > 2(t - s)$. There are $\mathcal{C}_g(p)$ terms of g that collide modulo $z^p - 1$. Let h be the $\mathcal{C}_g(p)$ -sparse polynomial comprised of those terms of g . As each term of h collides with at least one other term of h , $h \bmod (z^p - 1)$ has sparsity at most $\mathcal{C}_g(p)/2$. Since none of the terms of $g - h$ collide modulo $z^p - 1$, $(g - h) \bmod (z^p - 1)$ has sparsity exactly $t - \mathcal{C}_g(p)$. It follows that $g \bmod (z^p - 1)$ has sparsity at most $t - \mathcal{C}_g(p) + \mathcal{C}_g(p)/2 = t - \mathcal{C}_g(p)/2$. That is, $s \leq t - \mathcal{C}_g(p)/2$, and so $\mathcal{C}_g(p) \leq 2(t - s)$. \square

Corollary 11. *Suppose g has sparsity t , $g \bmod (z^q - 1)$ has sparsity s_q , and $g \bmod (z^p - 1)$ has sparsity $s_p \geq s_q$. Then $\mathcal{C}_g(p) \leq 2\mathcal{C}_g(q)$.*

Proof.

$$\begin{aligned} \mathcal{C}_g(p) &\leq 2(t - s_p) && \text{by the second inequality of Lemma 10,} \\ &\leq 2(t - s_q) && \text{since } s_p \geq s_q, \\ &\leq 2\mathcal{C}_g(q) && \text{by the first inequality of Lemma 10.} \end{aligned} \quad \square$$

\square

Suppose then that we have computed $g \bmod (z^p - 1)$, for p belonging to some set of primes S , and the minimum value of $\mathcal{C}_g(p)$, $p \in S$, is less than γ . Then a prime $p^* \in S$ for which $g \bmod (z^{p^*} - 1)$ has maximally many terms satisfies $\mathcal{C}_g(p^*) < 2\gamma$. We will call such a prime p^* an *ok prime*.

We then choose $\gamma = wT$ for an appropriate proportion $w \in (0, 1)$. We show that setting $w = 3/16$ allows that each recursive call reduces the sparsity of the subsequent polynomial by at least half. This would make $\lambda = \lceil \frac{10}{3w}(T - 1) \ln(D) \rceil = \lceil \frac{160}{9}(T - 1) \ln(D) \rceil$. As per Lemma 8, in order to guarantee with probability $1 - \varepsilon$ that we have come across a prime p such that $\mathcal{C}_g(p) \leq \gamma$, we will need to perform $\lceil \log 1/\varepsilon \rceil$ probes of degree $\mathcal{O}(T \log D)$. Procedure [FindOkPrime](#) summarizes how we find an ok prime.

A practical application would probably choose random primes by selecting random integer values in $[\lambda, 2\lambda]$ and then applying probabilistic primality testing. In order to ensure deterministic worst-case run-time, we could pick random primes in the range $[\lambda, 2\lambda]$ by using a sieve method to pre-compute all the primes up to 2λ .

2.2 Generating an approximation f^{**} of g

We suppose now that we have, with probability at least $1 - \varepsilon$, an ok prime p ; i.e., a prime p such that $\mathcal{C}_g(p) \leq 2wT$ for a suitable proportion w . We now use this ok prime p to construct a polynomial f^{**} containing *most* of the terms of $g = f - f^*$.

For a set of coprime moduli $\mathcal{Q} = \{q_1, \dots, q_k\}$ satisfying $\prod_{i=1}^k q_i > D$, we will compute $g \bmod (z^{p_{q_i}} - 1)$ for $1 \leq i \leq k$. Here we make no requirement that the q_i be prime. We merely require that the q_i are pairwise co-prime.

Procedure FindOkPrime($\mathcal{S}_f, f^*, T_g, D, \varepsilon$)

Input:

- \mathcal{S}_f , a straight-line program that computes a polynomial f
- f^* , a current approximation to f
- T_g and D , bounds on the sparsity and degree of $g = f - f^*$ respectively
- ε , a bound on the probability of failure

Output: With probability at least $1 - \varepsilon$, we return an “ok prime” for $g = f - f^*$

```

 $\lambda \leftarrow \max(21, \lceil \frac{160}{9}(T_g - 1) \ln D \rceil)$ 
 $(\text{max\_sparsity}, p) \leftarrow (0, 0)$ 
for  $i \leftarrow 1$  to  $\lceil \log 1/\varepsilon \rceil$  do
     $p' \leftarrow$  a random prime in  $[\lambda, 2\lambda]$ 
    if # of terms of  $(f - f^*) \bmod (z^{p'} - 1) \geq \text{max\_sparsity}$  then
         $\text{max\_sparsity} \leftarrow$  # of terms of  $(f - f^*) \bmod (z^{p'} - 1)$ 
         $p \leftarrow p'$ 
return  $p$ 

```

We choose the q_i as follows: denoting the i^{th} prime by p_i , we set $q_i = p_i^{\lfloor \log_{p_i} x \rfloor}$, for an appropriate choice of x . That is, we let q_i be the greatest power of the i^{th} prime that is no more than x . For $p_i \leq x$, we have $q_i \geq x/p_i$ and $q_i \geq p_i$. Either x/p_i or p_i is at least \sqrt{x} , and so $q_i \geq \sqrt{x}$ as well.

By Corollary 1 of Theorem 2 in [Rosser and Schoenfeld \(1962\)](#), there are more than $x/\ln x$ primes less than or equal to x for $x \geq 17$. Therefore

$$\prod_{p_i \leq x} q_i \geq (\sqrt{x})^{x/\ln x}.$$

As we want this product to exceed D , it suffices that

$$\ln D < \ln \left((\sqrt{x})^{x/\ln x} \right) = x/2.$$

Thus, if we choose $x \geq \max(2 \ln(D), 17)$ and $k = \lceil x/\ln x \rceil$, then $\prod_{i=1}^k q_i$ will exceed D . This means $q_i \in \mathcal{O}(\log D)$ and $p_{q_i} \in \mathcal{O}(T \log^2 D)$. The number of probes in this step is $k \in \mathcal{O}(\log(D)/\log \log(D))$. Since we will use the same set of moduli $\mathcal{Q} = \{q_1, \dots, q_k\}$ in every recursive call, we can pre-compute \mathcal{Q} prior to the first recursive call.

We now describe how to use the images $g \bmod (z^{p_{q_i}} - 1)$ to construct a polynomial f^{**} such that $g - f^{**}$ is at most $T/2$ -sparse.

If cz^e is a term of g that does not collide with any other terms modulo $z^p - 1$, then it certainly will not collide with other terms modulo $z^{p_q} - 1$ for any natural number q . Similarly, if $c^* z^{e^* \bmod p}$ appears in $g \bmod (z^p - 1)$ and there exists a unique term $c^* z^{e^* \bmod p_{q_i}}$ appearing in $g \bmod (z^{p_{q_i}} - 1)$ for $i = 1, 2, \dots, k$, then $c^* z^{e^*}$ is potentially a term of g . Note that $c^* z^{e^*}$ is not *necessarily* a term of g : consider the following example.

Example 12. *Let*

$$g(z) = 1 + z + z^2 + z^3 + z^{11+4} - z^{14 \cdot 11 + 4} - z^{15 \cdot 11 + 4},$$

with hard sparsity bound $T_g = 7$ and degree bound $D = 170$ and let $p = 11$. We have

$$g(z) \bmod (z^{11} - 1) = 1 + z + z^2 + z^3 - z^4.$$

As $\deg(g) = 170 < 2 \cdot 3 \cdot 5 \cdot 7 = 210$, it suffices to make the probes $g \bmod z^{11q} - 1$ for $q = 2, 3, 5, 7$. Probing our remainder black-box polynomial, we have

$$g \bmod (z^{22} - 1) = 1 + z + z^2 + z^3 - z^{15},$$

$$g \bmod (z^{33} - 1) = 1 + z + z^2 + z^3 - z^{26},$$

$$g \bmod (z^{55} - 1) = 1 + z + z^2 + z^3 - z^{48},$$

$$g \bmod (z^{77} - 1) = 1 + z + z^2 + z^3 - z^{15}.$$

In each of the images $g \bmod z^{pq} - 1$, there is a unique term whose degree is congruent to one of $e = 0, 1, 2, 3, 4$ modulo p . Four of these terms correspond to the terms $1, z, z^2, z^3$ appearing in g . Whereas the remaining term has degree e satisfying $e \equiv 1 \pmod{2}$, $e \equiv 2 \pmod{3}$, $e \equiv 3 \pmod{5}$, and $e \equiv 1 \pmod{7}$. By Chinese remaindering on the exponents, this gives a term $-z^{113}$ not appearing in g .

Definition 13. *Let $c^* z^{e^*}$, $e^* \leq D$ be a monomial such that $c^* z^{e^* \bmod p}$ appears in $g \bmod z^p - 1$, and $c^* z^{e^* \bmod pq_i}$ is the unique term of degree congruent to e^* modulo p appearing in $g \bmod (z^{pq_i} - 1)$ for each modulus q_i . If $c^* z^{e^*}$ is not a term of g we call it a deceptive term.*

Fortunately, we can detect a collision comprised of only two terms. Namely, if $c_1 z^{e_1} + c_2 z^{e_2}$ collide, there will exist a q_i such that $q_i \nmid (e_1 - e_2)$. That is, $g \bmod (z^{pq_i} - 1)$ will have two terms whose degree is congruent to $e_1 \bmod p$. Once we observe that, we know the term $(c_1 + c_2) z^{e_1 \bmod p}$ appearing in $g \bmod (z^p - 1)$ was not a distinct term, and we can ignore exponents of the congruence class $e_1 \bmod p$ in subsequent images $g \bmod (z^{pq_i} - 1)$.

Thus, supposing $g \bmod (z^p - 1)$ has at most 2γ colliding terms and at least $t - 2\gamma$ non-colliding terms, f^{**} will have the $t - 2\gamma$ non-colliding terms of g , plus potentially an additional $\frac{2}{3}\gamma$ deceptive terms produced by the colliding terms of g . In any case, $g - f^{**}$ has sparsity at most $\frac{8}{3}\gamma$. Choosing $\gamma = \frac{3}{16}T_g$ guarantees that $g - f^{**}$ has sparsity at most $T_g/2$. This would make $\lambda = \lceil \frac{160}{9}(T_g - 1) \ln(D) \rceil$.

Procedure [ConstructApproximation](#) gives a pseudocode description of how we construct f^{**} .

If we find a prospective term in our new approximation f^{**} has degree greater than D , then we know that term must have been a deceptive term and discard it. There are other obvious things we can do to recognize deceptive terms which we exclude here. For instance, we should check that all terms from images modulo $z^{pq} - 1$ whose degrees agree modulo p share the same coefficient.

Procedure ConstructApproximation($\mathcal{S}_f, f^*, D, p, \mathcal{Q}$)

Input:

- \mathcal{S}_f , a straight-line program that computes a polynomial f
- f^* , a current approximation to f
- D a bound on the degree of $g = f - f^*$
- p , an ok prime for g (with high probability)
- \mathcal{Q} , a set of co-prime moduli whose product exceeds D

Output: A polynomial f^{**} such that, if p is an ok prime, $g - f^{**}$ has sparsity at most $\lfloor T_g/2 \rfloor$, where g has at most T_g terms.

```

// Collect images of g
 $\mathcal{E} \leftarrow$  set of exponents of terms in  $(f - f^*) \bmod (z^p - 1)$ 
for  $q \in \mathcal{Q}$  do
     $h \leftarrow (f - f^*) \bmod (z^{pq} - 1)$ 
    for each term  $cz^e$  in  $h$  do
        if  $E_{(e \bmod p), q}$  is already initialized then  $\mathcal{E} \leftarrow \mathcal{E} / \{e \bmod p\}$  else
             $E_{(e \bmod p), q} \leftarrow e \bmod q$ 

// Construct terms of new approximation of  $g$ ,  $f^{**}$ 
 $f^{**} \leftarrow 0$ 
for  $e_p \in \mathcal{E}$  do
     $e \leftarrow$  least nonnegative solution to  $\{e = E_{e_p, q} \bmod q \mid q \in \mathcal{Q}\}$ 
     $c \leftarrow$  coefficient of  $z^{e_p}$  term in  $(f - f^*) \bmod (z^p - 1)$ 
    if  $e \leq D$  then  $f^{**} \leftarrow f^{**} + cz^e$ 
return  $f^{**}$ 

```

Procedure Interpolate(\mathcal{S}_f, T, D, μ)

Input:

- \mathcal{S}_f , a straight-line program that computes a polynomial f
- T and D , bounds on the sparsity and degree of f , respectively
- μ , an upper bound on the probability of failure

Output: With probability at least $1 - \mu$, we return f

```

 $x \leftarrow \max(2 \ln(D), 17)$ 
 $\mathcal{Q} \leftarrow \{p^{\lfloor \log_p x \rfloor} : p \text{ is prime}, p \leq x\}$ 
return InterpolateRecurse( $\mathcal{S}_f, 0, T, D, \mathcal{Q}, \mu/(\log T + 1)$ )

```

Procedure `InterpolateRecurse`($\mathcal{S}_f, f^*, T_g, D, \mathcal{Q}, \varepsilon$)

Input:

- \mathcal{S}_f , a straight-line program that computes a polynomial f
- f^* , a current approximation to f
- T_g and D , bounds on the sparsity and degree of $g = f - f^*$, respectively
- \mathcal{Q} , a set of coprime moduli whose product is at least D
- ε , a bound on the probability of failure at one recursive step

Output: With probability at least $1 - \mu$, the algorithm outputs f

if $T_g = 0$ **then return** f^*

$p \leftarrow \text{FindOkPrime}(\mathcal{S}_f, f^*, T_g, D, \varepsilon)$

$f^{**} \leftarrow \text{ConstructApproximation}(\mathcal{S}_f, f^*, D, p, \mathcal{Q})$

return `InterpolateRecurse`($\mathcal{S}_f, f^* + f^{**}, \lfloor T_g/2 \rfloor, D, \mathcal{Q}, \varepsilon$)

2.3 Recursively interpolating $f - f^*$

Once we have constructed f^{**} , we refine our approximation f^* by adding f^{**} to it, giving us a new difference $g = f - f^*$ containing at most half the terms of the previous polynomial g . We recursively interpolate our new polynomial g . With an updated sparsity bound $\lfloor T_g/2 \rfloor$, we update the values of γ and λ and perform the steps of Sections 2.1 and 2.2. We recurse in this fashion $\log T$ times. Thus, the total number of probes becomes

$$\mathcal{O}\left(\log T \left(\frac{\log D}{\log \log D} + \log(1/\varepsilon)\right)\right),$$

of degree at most $\mathcal{O}(T \log^2 D)$.

Note now that in order for this method to work we need that, at every recursive call, we in fact get a good prime, otherwise our sparsity bound on the subsequent difference of polynomials could be incorrect. At every stage we succeed with probability $1 - \varepsilon$, thus the probability of failure is $1 - (1 - \varepsilon)^{\lceil \log T \rceil}$. This is less than $\lceil \log T \rceil \varepsilon$. If we want to succeed with probability μ , then we can choose $\varepsilon = \frac{\mu}{\log T + 1} \in \mathcal{O}(\frac{\mu}{\log T})$.

`Interpolate` pre-computes our set of moduli \mathcal{Q} , then makes the first recursive call to `InterpolateRecurse`, which subsequently calls itself.

2.4 A cost analysis

We analyse the cost of our algorithm, thereby proving Theorems 1 and 5.

2.4.1 Pre-computation.

Using the wheel sieve (Pritchard, 1982), we can compute the set of primes up to $x \in \mathcal{O}(\log D)$ in $\mathcal{O}(\log D)$ bit operations. From this set of primes we obtain \mathcal{Q} by computing $p^{\lfloor \log_p x \rfloor}$ for $p \leq \sqrt{x}$ by way of squaring-and-multiplying. For each

such prime, this costs $\mathcal{O}(\log x)$ bit operations, so the total cost of computing Q is $\mathcal{O}(\log D)$.

2.4.2 Finding ok primes.

In one recursive call, we will look at some $\log 1/\varepsilon = \mathcal{O}(\log 1/\mu \log \log T)$ primes in the range $[\lambda, 2\lambda]$ in order to find an ok prime. Any practical implementation would select such primes by using probabilistic primality testing on random integer values in the range $[\lambda, 2\lambda]$; however, the probabilistic analysis of such an approach, in the context of our interpolation algorithm, becomes somewhat ungainly. We merely note here that we could instead pre-compute primes up to our initial value of $\lambda \in \mathcal{O}(T \log D)$ in $\mathcal{O}(T \log D)$ bit operations by way of the wheel sieve.

Each prime p is of order $T \log D$, and so, per our discussion in Section 1, each probe costs $\mathcal{O}(LT \log D)$ ring operations and similarly many bit operations. Considering the $\mathcal{O}(\log T)$ recursive calls, this totals $\mathcal{O}(LT \log D \log 1/\mu)$ ring and bit operations.

2.4.3 Constructing the new approximation f^{**} .

Constructing f^{**} requires $\mathcal{O}(\log D)$ probes of degree $\mathcal{O}(T \log^2 D)$. This costs $\mathcal{O}(LT \log^3 D)$ ring and bit operations. Performing these probes at each $\mathcal{O}(\log T)$ recursive call introduces an additional factor of $\log T$, which does not affect the “soft-Oh” complexity. This step dominates the cost of the algorithm.

Building a term cz^e of f^{**} amounts to solving a set of congruences. By Theorem 5.8 of [Gathen and Gerhard \(2003\)](#), this requires some $\mathcal{O}(\log^2 D)$ word operations. Thus the total cost of Chinese remaindering to construct f^{**} becomes $\mathcal{O}(T \log^2 D)$. Again, the additional $\log T$ factor due to the recursive calls does not affect the stated complexity.

3 Conclusions

We have presented a recursive algorithm for interpolating a polynomial f given by a straight-line program, using probes of smaller degree than in previously known methods. We achieve this by looking for “ok” primes which separate most of the terms of f , as opposed to “good” primes which separate all of the terms of f . As is seen in Table 1, our algorithm is an improvement over previous algorithms for moderate values of T .

This work suggests a number of problems for future work. We believe our algorithms have the potential for good numerical stability, and could improve on [Giesbrecht and Roche’s \(2011\)](#) work on numerical interpolation of sparse complex polynomials, hopefully capitalizing on the lower degree probes. Our Monte Carlo algorithms are now more efficient than the best known algorithms for polynomial identity testing, and hence these cannot be used to make them error free. We would ideally like to expedite polynomial identity testing of straight-line programs, the best known methods currently due to [Bläser et al. \(2009\)](#).

Finally, we believe there is still room for improvement in sparse interpolation algorithms. The vector of exponents of f comprises some $T \log D$ bits. Assuming no collisions, a degree- ℓ probe gives us some $t \log \ell$ bits of information about these exponents. One might hope, aside from some seemingly rare degenerate cases, that $\log D$ probes of degree $T \log D$ should be sufficient to interpolate f .

4 Acknowledgements

We would like to thank Reinhold Burger and Colton Pauderis for their feedback on a draft of this paper.

References

- Michael Ben-Or and Prasoona Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 301–309. ACM, 1988.
- Markus Bläser, Moritz Hardt, Richard J. Lipton, and Nisheeth K. Vishnoi. Deterministically testing sparse polynomial identities of unbounded degree. *Information Processing Letters*, 109(3):187–192, 2009.
- Nicolas Bruno, Joos Heintz, Guillermo Matera, and Rosita Wachenchauzer. Functional programming concepts and straight-line programs in computer algebra. *Mathematics and Computers in Simulation*, 60(6):423–473, 2002. doi: 10.1016/S0378-4754(02)00035-6.
- Peter Bürgisser, Michael Clausen, and M. Amin Shokrollahi. *Algebraic Complexity Theory*, volume 315 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, 1997.
- David G. Cantor and Erich Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, 28:693–701, 1991.
- R. de Prony. Essai expérimental et analytique sur les lois de la dilatabilité et sur celles de la force expansive de la vapeur de l’eau et de la vapeur de l’alkool, à différentes températures. *J. de l’École Polytechnique*, 1:24–76, 1795.
- Sanchit Garg and Éric Schost. Interpolation of polynomials given by straight-line programs. *Theor. Comput. Sci.*, 410(27-29):2659–2662, June 2009. ISSN 0304-3975. doi: 10.1016/j.tcs.2009.03.030. URL <http://dx.doi.org/10.1016/j.tcs.2009.03.030>.
- Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, New York, NY, USA, 2nd edition, 2003. ISBN 0521826462.

- Mark Giesbrecht and Daniel S. Roche. Diversification improves interpolation. *ISSAC '11*, pages 123–130, 2011. doi: 10.1145/1993886.1993909. URL <http://doi.acm.org/10.1145/1993886.1993909>.
- Mark Giesbrecht, George Labahn, and Wen-shin Lee. Symbolic–numeric sparse interpolation of multivariate polynomials. *Journal of Symbolic Computation*, 44(8):943–959, 2009.
- Erich Kaltofen. Factorization of polynomials given by straight-line programs. In *Randomness and Computation*, pages 375–412. JAI Press, 1989.
- Erich Kaltofen, Y. N. Lakshman, and John-Michael Wiley. Modular rational sparse multivariate polynomial interpolation. In *Proceedings of the international symposium on Symbolic and algebraic computation*, ISSAC '90, pages 135–139, New York, NY, USA, 1990. ACM. doi: 10.1145/96877.96912.
- Paul Pritchard. Explaining the wheel sieve. *Acta Informatica*, 17(4):477–485, 1982.
- J. Barkley Rosser and Lowell Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois J. Math.*, 6:64–94, 1962. ISSN 0019-2082.
- Carl Sturtevant and Zhi-Li Zhang. Efficiently inverting bijections given by straight line programs. In *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on*, pages 327–334. IEEE, Oct 1990. doi: 10.1109/FSCS.1990.89551.