comput. complex. *Online First*
DOI 10.1007/s00037-010-0294-0

computational complexity

INTERPOLATION OF SHIFTED-LACUNARY POLYNOMIALS

Mark Giesbrecht and Daniel S. Roche

Abstract. Given a "black box" function to evaluate an unknown rational polynomial $f \in \mathbb{Q}[x]$ at points modulo a prime p, we exhibit algorithms to compute the representation of the polynomial in the sparsest shifted power basis. That is, we determine the sparsity $t \in \mathbb{Z}_{>0}$, the shift $\alpha \in \mathbb{Q}$, the exponents $0 \le e_1 < e_2 < \cdots < e_t$, and the coefficients $c_1, \ldots, c_t \in \mathbb{Q} \setminus \{0\}$ such that

$$f(x) = c_1(x-\alpha)^{e_1} + c_2(x-\alpha)^{e_2} + \cdots + c_t(x-\alpha)^{e_t}\,.$$

The computed sparsity t is *absolutely* minimal over any shifted power basis. The novelty of our algorithm is that the complexity is polynomial in the (sparse) representation size, which may be logarithmic in the degree of f. Our method combines previous celebrated results on sparse interpolation and computing sparsest shifts, and provides a way to handle polynomials with extremely high degree which are, in some sense, sparse in information.

Keywords. Sparse interpolation, sparsest shift, lacunary polynomials.

Subject classification. Primary 68W30; Secondary 12Y05.

1. Introduction

Interpolating an unknown polynomial from a set of evaluations is a problem which has interested mathematicians for hundreds of years, and which is now implemented as a standard function in most computer algebra systems. To illustrate some different kinds of interpolation problems, consider the following three representations for a polynomial f of degree n:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n\,, \tag{1.1}$$

$$f(x) = b_0 + b_1 x^{d_1} + b_2 x^{d_2} + \cdots + b_s x^{d_s}\,, \tag{1.2}$$

$$f(x) = c_0 + c_1(x-\alpha)^{e_1} + c_2(x-\alpha)^{e_2} + \cdots + c_t(x-\alpha)^{e_t}\,. \tag{1.3}$$

In (1.1) we see the dense representation of the polynomial, where all coefficients (even zeroes) are represented. Newton and Waring discovered methods to interpolate f in time proportional to the size of this representation in the 18^{th} century. The sparse or lacunary representation is shown in (1.2), wherein only the terms with non-zero coefficients are written (with the possible exception of the constant coefficient b_0). Here we say that f is *s-sparse* because it has exactly s non-zero and non-constant terms; the constant coefficient requires special treatment in our algorithms regardless of whether or not it is zero, and so we do not count it towards the total number of terms. Ben-Or & Tiwari (1988) discovered a method to interpolate in time polynomial in the size of this representation. Kaltofen & Lee (2003) present and analyze very efficient algorithms for this problem, in theory and practice. The Ben-Or & Tiwari method has also been examined in the context of approximate (floating point) polynomials by Giesbrecht *et al.* (2006), where the similarity to the 1795 method of de Prony is also pointed out. Bläser *et al.* (2009) consider the more basic problem of identity testing of sparse polynomials over $\mathbb{Q}$, and present a deterministic polynomial-time algorithm.

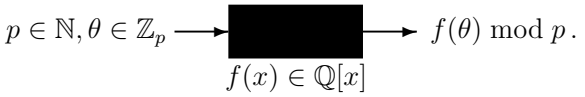
In (1.3), f is written in the shifted power basis $1, (x-\alpha), (x-\alpha)^2, \ldots$, and we say that α is a *t-sparse shift* of f because the representation has exactly t non-zero and non-constant terms in this basis. When α is chosen so that t is absolutely minimal in (1.3), we call this the *sparsest shift* of f. We present new algorithms to interpolate $f \in \mathbb{Q}[x]$, given a black box for evaluation, in time proportional to the size of the shifted-lacunary representation corresponding to (1.3). It is easy to see that t could be exponentially smaller than both n and s, for example when $f = (x+1)^n$, demonstrating that our algorithms are providing a significant improvement in complexity over those previously known, whose running times are polynomial in n and s.

The main applications of all these methods for polynomial interpolation are signal processing and reducing intermediate expression swell. Dense and sparse interpolation have been applied successfully to both these ends, and our new algorithms effectively extend the class of polynomials for which such applications can be made.

The most significant challenge here is computing the sparsest shift $\alpha \in \mathbb{Q}$. Computing this value from a set of evaluation points was stated as an open problem by Borodin & Tiwari (1991). An algorithm for a generalization of our problem in the dense representation was given by Grigoriev & Karpinski (1993), though its cost is exponential in the size of the output; they admit that the dependency on the degree of the polynomial is probably not optimal. Our algorithm achieves deterministic polynomial-time complexity for polynomials over the rational numbers. We are always careful to count the *bit complexity* –

the number of fixed-precision machine operations – and hence account for any coefficient growth in the solution or intermediate expressions.

The black box model we use is slightly modified from the traditional one:



Given a prime p and an element θ in $\mathbb{Z}_p$, the black box computes the value of the unknown polynomial evaluated at θ over the field $\mathbb{Z}_p$. (An error is produced exactly in those unfortunate circumstances that p divides the denominator of $f(\theta)$.) We generally refer to this as a *modular black box*. To account for the reasonable possibility that the cost of black box calls depends on the size of p, we define κ_f to be an upper bound on the number of field operations in $\mathbb{Z}_p$ used in black box evaluation, for a given polynomial $f \in \mathbb{Q}[x]$.

Some kind of extension to the standard black box, such as the modular black box proposed here, is in fact necessary, since the value of a polynomial of degree n at any point other than $0, \pm 1$ will typically have n bits or more. Thus, any algorithm whose complexity is proportional to $\log n$ cannot perform such an evaluation over $\mathbb{Q}$ or $\mathbb{Z}$. Other possibilities might include allowing for evaluations on the unit circle in some representation of a subfield of $\mathbb{C}$, or returning only a limited number of bits of precision for an evaluation.

To be precise about our notion of size, first define $\text{size}(q)$ for $q \in \mathbb{Q}$ to be the number of bits needed to represent q. So if we write $q = \frac{a}{b}$ with $a \in \mathbb{Z}$, $b \in \mathbb{N}$, and $\gcd(a, b) = 1$, then $\text{size}(q) = \lceil \log_2(|a| + 1) \rceil + \lceil \log_2(b + 1) \rceil + 1$. For a rational polynomial f as in (1.3), define:

$$\text{size}(f) = \text{size}(\alpha) + \sum_{i=0}^{t} \text{size}(c_i) + \sum_{i=1}^{t} \text{size}(e_i)\,. \tag{1.4}$$

We will often employ the following upper bound for simplicity:

$$\text{size}(f) \le \text{size}(\alpha) + t\big(H(f) + \log_2 n\big)\,, \tag{1.5}$$

where $H(f)$ is defined as $\max_{0 \le i \le t} \text{size}(c_i)$.

Our algorithms will have polynomial complexity in the smallest possible $\text{size}(f)$. For the complexity analysis, we use the normal notion of a "multiplication time" function $\mathsf{M}(n)$, which is the number of field operations required to compute the product of polynomials with degrees less than n, or integers with sizes at most n. We always assume that $\mathsf{M}(n) \in \Omega(n)$ and

$\mathsf{M}(n) \in O(n^2)$. Using the results from Cantor & Kaltofen (1991), we can write $\mathsf{M}(n) \in O(n \log n \log\log n)$.

The remainder of the paper is structured as follows. In Section 2 we show how to find the sparsest shift from evaluation points in $\mathbb{Z}_p$, where p is a prime with some special properties provided by some "oracle". In Section 3 we show how to perform sparse interpolation given a modular black box for a polynomial. In Section 4 we show how to generate primes such that a sufficient number satisfy the conditions of our oracle. Section 5 provides the complexity analysis of our algorithms. We conclude in Section 6, and introduce some open questions.

2. Computing the Sparsest Shift

For a polynomial $f \in \mathbb{Q}[x]$, we first focus on computing the sparsest shift $\alpha \in \mathbb{Q}$ so that $f(x+\alpha)$ has a minimal number of non-zero and non-constant terms. This information will later be used to recover a representation of the unknown polynomial.

2.1. The polynomial $f^{(p)}$. Here, and for the remainder of this paper, for a prime p and $f \in \mathbb{Q}[x]$, define $f^{(p)} \in \mathbb{Z}_p[x]$ to be the unique polynomial with degree less than p which is equivalent to f modulo $x^p - x$ and with all coefficients reduced modulo p. From Fermat's Little Theorem, we then see immediately that $f^{(p)}(\alpha) \equiv f(\alpha) \bmod p$ for all $\alpha \in \mathbb{Z}_p$. Hence $f^{(p)}$ can be found by evaluating f at each point $0, 1, \ldots, p-1$ modulo p and using dense interpolation over $\mathbb{Z}_p[x]$.

Notice that, over $\mathbb{Z}_p[x]$, $(x-\alpha)^p \equiv x - \alpha \mod x^p - x$, and therefore $(x-\alpha)^{e_i} \equiv (x-\alpha)^k$ for any $k \neq 0$ such that $e_i \equiv k \bmod (p-1)$. The smallest such k is in the range $\{1, 2, \ldots, p\}$; we now define this with some more notation. For $a \in \mathbb{Z}$ and positive integer m, define $a \operatorname{rem}_1 m$ to be the unique integer in the range $\{1, 2, \ldots, m\}$ which is congruent to a modulo m. As usual, $a \operatorname{rem} m$ denotes the unique congruent integer in the range $\{0, 1, \ldots, m-1\}$.

If f is as in (1.3), then by reducing term-by-term we can write

$$f^{(p)}(x) = (c_0 \operatorname{rem} p) + \sum_{i=1}^{t} (c_i \operatorname{rem} p)(x - \alpha_p)^{e_i \operatorname{rem}_1 (p-1)}, \tag{2.1}$$

where α_p is defined as $\alpha \operatorname{rem} p$. Hence, for some $k \leq t$, α_p is a k-sparse shift for $f^{(p)}$. That is, the polynomial $f^{(p)}(x+\alpha_p)$ over $\mathbb{Z}_p[x]$ has at most t non-zero and non-constant terms.

Computing $f^{(p)}$ from a modular black box for f is straightforward. First, use p black-box calls to determine $f(i) \operatorname{rem} p$ for $i = 0, 1, \ldots, p-1$. Recalling that κ_f is the number of field operations in $\mathbb{Z}_p$ for each black-box call, the cost of this step is $O(p\kappa_f \mathsf{M}(\log p))$ bit operations. Second, we use the well-known divide-and-conquer method to interpolate $f^{(p)}$ into the dense representation (see, e.g., Borodin & Munro (1975, Section 4.5)). Since $\deg f^{(p)} < p$, this step has bit complexity $O(\mathsf{M}(p)\mathsf{M}(\log p) \log p)$.

Furthermore, for any $\alpha \in \mathbb{Z}_p$, the dense representation of $f^{(p)}(x+\alpha)$ can be computed in exactly the same way as the second step above, simply by shifting the indices of the already-evaluated points by α. This immediately gives a naïve algorithm for computing the sparsest shift of $f^{(p)}$: compute $f^{(p)}(x+\gamma)$ for $\gamma = 0, 1, \ldots, p-1$, and return the γ that minimizes the number of non-zero, non-constant terms. The bit complexity of this approach is $O(p \log p\, \mathsf{M}(p)\mathsf{M}(\log p))$, which for our applications will often be less costly than the more sophisticated approaches of, e.g., Lakshman & Saunders (1996) or Giesbrecht *et al.* (2003), precisely because p will not be very much larger than $\deg f^{(p)}$.

2.2. Overview of Approach. We will make repeated use of the following fundamental theorem from Lakshman & Saunders (1996):

FACT 2.2. *Let* F *be an arbitrary field and* $f \in \mathsf{F}[x]$*, and suppose* $\alpha \in \mathsf{F}$ *is such that* $f(x+\alpha)$ *has* t *non-zero and non-constant terms. If* $\deg f \geq 2t+1$ *then* α *is the unique sparsest shift of* f*.*

From this we can see that, if α is the unique sparsest shift of f, then $\alpha_p = \alpha \operatorname{rem} p$ is the unique sparsest shift of $f^{(p)}$ *provided that* $\deg f^{(p)} \geq 2t+1$. This observation provides the basis for our algorithm.

The input to the algorithms will be a modular black box for evaluating a rational polynomial, as described above, and bounds on the maximal size of the unknown polynomial. Note that such bounds are a necessity in any type of black-box interpolation algorithm, since otherwise we could never be sure that the computed polynomial is really equal to the black-box function at *every* point. Specifically, we require $B_A, B_T, B_H, B_N \in \mathbb{N}$ such that

$$\begin{aligned} \operatorname{size}(\alpha) &\leq B_A, \\ t &\leq B_T, \\ \operatorname{size}(c_i) &\leq B_H, \quad \text{for } 0 \leq i \leq t, \\ \log_2 n &\leq B_N. \end{aligned}$$

By considering the following polynomial:

$$c(x-\alpha)^n + (x-\alpha)^{n-1} + \cdots + (x-\alpha)^{n-t+1},$$

we see that these bounds are independent – that is, none is polynomially-bounded by the others – and therefore are all necessary.

We are now ready to present the algorithm for computing the sparsest shift α almost in its entirety. The only part of the algorithm left unspecified is an *oracle* which, based on the values of the bounds, produces primes to use. We want primes p such that $\deg f^{(p)} \geq 2t+1$, which allows us to recover one modular image of the sparsest shift α. But since we do not know the exact value of t or the degree n of f over $\mathbb{Q}[x]$, we define some prime p to be a *good prime for sparsest shift computation* if and only if $\deg f^{(p)} \geq \min\{2B_T+1, n\}$. For the remainder of this section, "good prime" means "good prime for sparsest shift computation." Our oracle indicates when enough primes have been produced so that at least one of them is guaranteed to have been a good prime, which is necessary for the procedure to terminate. The details of how to construct such an oracle will be considered in Section 4.

Algorithm 2.3. Computing the sparsest shift.

Input:
- A modular black box for an unknown polynomial $f \in \mathbb{Q}[x]$
- Bounds $B_A, B_T, B_H, B_N \in \mathbb{N}$ as described above
- An oracle which produces primes and indicates when at least one good prime must have been produced

Output: A sparsest shift α of f.

1. $P \leftarrow 1, \qquad \mathcal{G} \leftarrow \emptyset$
2. While $\log_2 P < 2B_A + 1$ do 3–14
3. $\quad p \leftarrow$ new prime from the oracle
4. $\quad$ Evaluate $f(i) \operatorname{rem} p$ for $i = 0, 1, \ldots, p-1$
5. $\quad$ Use dense interpolation to compute $f^{(p)}$
6. $\quad$ If $\deg f^{(p)} \geq 2B_T + 1$ then
7. $\quad\quad$ Use dense interpolation to compute $f^{(p)}(x+\gamma)$ for $\gamma = 1, 2, \ldots, p-1$
8. $\quad\quad \alpha_p \leftarrow$ the unique sparsest shift of $f^{(p)}$
9. $\quad\quad P \leftarrow P \cdot p, \qquad \mathcal{G} \leftarrow \mathcal{G} \bigcup \{p\}$
10. $\quad$ Else if $P = 1$ and oracle indicates ≥ 1 good prime has been produced then
11. $\quad\quad q \leftarrow$ least prime such that $\log_2 q > 2B_T B_A + B_H$ (computed directly)
12. $\quad\quad$ Evaluate $f(i) \operatorname{rem} q$ for $i = 0, 1, \ldots, 2B_T$
13. $\quad\quad$ Compute $f \in \mathbb{Q}[x]$ with $\deg f \leq 2B_T$ by dense interpolation in $\mathbb{Z}_q[x]$ followed by rational reconstruction on the coefficients

14. **Return** A sparsest shift α computed by a univariate algorithm from Giesbrecht *et al.* (2003) on input f
15. **Return** The unique $\alpha = a/b \in \mathbb{Q}$ such that $|a|, b \leq 2^{B_A}$ and $a \equiv b\alpha_p \bmod p$ for each $p \in \mathcal{G}$, using Chinese remaindering and rational reconstruction

THEOREM 2.4. *With inputs as specified, Algorithm 2.3 correctly returns a sparsest shift α of f.*

PROOF. Let f, B_A, B_T, B_H, B_N be the inputs to the algorithm, and suppose t, α are as specified in (1.3).

First, consider the degenerate case where $n \leq 2B_T$, i.e., the bound on the sparsity of the sparsest shift is at least half the actual degree of f. Then, since each $f^{(p)}$ can have degree at most n (regardless of the choice of p), the condition of Step 6 will never be true. Hence Steps 10–14 will eventually be executed. The size of coefficients over the standard power basis is bounded by $2B_TB_A + B_H$ since $\deg f \leq 2B_T$, and therefore f will be correctly computed on Step 5. In this case, Fact 2.2 may not apply, i.e. the sparsest shift may not be unique, but the algorithms from Giesbrecht *et al.* (2003) will still produce a sparsest shift of f.

Now suppose instead that $n \geq 2B_T + 1$. The oracle eventually produces a good prime p, so that $\deg f^{(p)} \geq 2B_T + 1$. Since $t \leq B_T$ and $f^{(p)}$ has at most t non-zero and non-constant terms in the $(\alpha \operatorname{rem} p)$-shifted power basis, the value computed as α_p on Step 8 is exactly $\alpha \operatorname{rem} p$, by Fact 2.2. The value of P will also be set to $p > 1$ here, and can only increase. So the condition of Step 10 is never true. Since the numerator and denominator of α are both bounded above by 2^{B_A}, we can use rational reconstruction to compute α once we have the image modulo P for $P \geq 2^{2B_A+1}$. Therefore, when we reach Step 15, we have enough images α_p to recover and return the correct value of α. □

We still need to specify which algorithm to use to compute the sparsest shift of a densely-represented $f \in \mathbb{Q}[x]$ on Step 14. To make Algorithm 2.3 completely deterministic, we should use the univariate symbolic algorithm from Giesbrecht *et al.* (2003, Section 3.1), although this will have very high complexity. Using a probabilistic algorithm instead gives the following, which follows directly from the referenced work.

THEOREM 2.5. *If the "two projections" algorithm of Giesbrecht* et al. *(2003, Section 3.3) is used on Step 14, then Steps 10–14 of Algorithm 2.3 can be performed with $O(B_T^2\mathsf{M}(B_T^4B_A + B_T^3B_H))$ bit operations, plus $O(\kappa_f B_T\mathsf{M}(B_TB_A + B_H))$ bit operations for the black-box evaluations.*

The precise complexity analysis proving that the entire Algorithm 2.3 has bit complexity polynomial in the bounds given depends heavily on the size and number of primes p that are used, and so must be postponed until Section 5.1, after our discussion on choosing primes.

EXAMPLE 2.6. Suppose we are given a modular black box for the following unknown polynomial:

$$\begin{aligned} f(x) = x^{15} &- 45x^{14} + 945x^{13} - 12285x^{12} + 110565x^{11} - 729729x^{10} \\ &+ 3648645x^9 - 14073345x^8 + 42220035x^7 - 98513415x^6 \\ &+ 177324145x^5 - 241805625x^4 + 241805475x^3 - 167403375x^2 \\ &+ 71743725x - 14348421, \end{aligned}$$

along with the bounds $B_A = 4$, $B_T = 2$, $B_H = 4$, and $B_N = 4$. One may easily confirm that $f(x) = (x-3)^{15} - 2(x-3)^5$, and hence these bounds are actually tight.

Now suppose the oracle produces $p = 7$ in Step 3. We use the black box to find $f(0), f(1), \ldots, f(6)$ in $\mathbb{Z}_7$, and dense interpolation to compute

$$f^{(7)}(x) = 5x^5 + 2x^4 + 3x^3 + 6x^2 + x + 4.$$

Since $\deg f^{(7)} = 5 \geq 2B_T + 1$, we move on to Step 8 and compute each $f^{(7)}(x+\gamma)$ with $\gamma = 1, 2, \ldots, 6$. Examining these, we see that $f^{(7)}(x+3) = 5x^5 + x^3$ has the fewest non-zero and non-constant terms, and so set α_7 to 3 on Step 8. This means the sparsest shift must be congruent to 3 modulo 7. This provides a single modular image for use in Chinese remaindering and rational reconstruction on Step 15, after enough successful iterations for different primes p. ◊

2.3. Conditions for Success. We have seen that, provided $\deg f > 2B_T$, a good prime p is one such that $\deg f^{(p)} > 2B_T$. The following theorem provides (quite loose) sufficient conditions on p to satisfy this requirement.

THEOREM 2.7. *Let $f \in \mathbb{Q}[x]$ as in (1.3) and $B_T \in \mathbb{N}$ such that $t \leq B_T$. Then, for some prime p, the degree of $f^{(p)}$ is greater than $2B_T$ whenever the following hold:*

- $c_t \not\equiv 0 \mod p$;
- $\forall i \in \{1, 2, \ldots, t-1\},\ e_t \not\equiv e_i \mod (p-1)$;
- $\forall i \in \{1, \ldots, 2B_T\},\ e_t \not\equiv i \mod (p-1)$.

PROOF. The first condition guarantees that the last term of $f^{(p)}(x)$ as in (2.1) does not vanish. We also know that there is no other term with the same degree from the second condition. Finally, the third condition tells us that the degree of the last term will be greater than $2B_T$. Hence the degree of $f^{(p)}$ is greater than $2B_T$. □

For purposes of computation it will be convenient to simplify the above conditions to two non-divisibility requirements, on p and $p-1$ respectively:

COROLLARY 2.8. *Let f, B_T, B_H, B_N be as in the input to Algorithm 2.3 with $\deg f > 2B_T$. Then there exist $C_1, C_2 \in \mathbb{N}$ with $\log_2 C_1 \leq 2B_H$ and $\log_2 C_2 \leq B_N(3B_T - 1)$ such that $\deg f^{(p)} > 2B_T$ whenever $p \nmid C_1$ and $(p-1) \nmid C_2$.*

PROOF. Write f as in (1.3). We will use the sufficient conditions given in Theorem 2.7. Write $|c_t| = a/b$ for $a, b \in \mathbb{N}$ relatively prime. In order for $c_t \operatorname{rem} p$ to be well-defined and not zero, neither a nor b can vanish modulo p. This is true whenever $p \nmid ab$. Set $C_1 = ab$. Since $a, b \leq 2^{B_H}$, $\log_2 C_1 = \log_2(ab) \leq 2B_H$.

Now write

$$C_2 = \prod_{i=1}^{t-1}(e_t - e_i) \cdot \prod_{i=1}^{2B_T}(e_t - i).$$

We can see that the second and third conditions of Theorem 2.7 are satisfied whenever $(p-1) \nmid C_2$. Now, since each integer e_i is distinct and positive, and e_t is the greatest of these, each $(e_t - e_i)$ is a positive integer less than e_t. Similarly, since $e_t = \deg f > 2B_T$, each $(e_t - i)$ in the second product is also a positive integer less than e_t. Therefore, using the fact that $t \leq B_T$, we see $C_2 \leq e_t^{3B_T - 1}$. Furthermore, $e_t \leq 2^{B_N}$, so we know that $\log_2 C_2 \leq B_N(3B_T - 1)$. □

A similar criteria for success is required in Bläser *et al.* (2009), and they employ Linnik's theorem to obtain a polynomial-time algorithm for polynomial identity testing. Linnik's theorem was also employed in Giesbrecht & Roche (2007) to yield a much more expensive deterministic polynomial-time algorithm for finding sparse shifts than the one presented here.

3. Interpolation

Once we know the value of the sparsest shift α of f, we can trivially construct a modular black box for the t-sparse polynomial $f(x + \alpha)$ using the modular black box for f. Therefore, for the purposes of interpolation, we can assume $\alpha = 0$, and focus only on interpolating a t-sparse polynomial $f \in \mathbb{Q}[x]$ given a modular black box for its evaluation. The basic techniques of this section are,

for the most part, known in the literature. However, a unified presentation in terms of bit complexity for our model of modular black boxes will be helpful.

For convenience, we restate the notation for f and $f^{(p)}$, given a prime p:

$$f = c_0 + c_1 x^{e_1} + c_2 x^{e_2} + \cdots + c_t x^{e_t}\,, \tag{3.1}$$

$$f^{(p)} = (c_0 \operatorname{rem} p) + (c_1 \operatorname{rem} p) x^{e_1 \operatorname{rem}_1 (p-1)} + \cdots + (c_t \operatorname{rem} p) x^{e_t \operatorname{rem}_1 (p-1)}\,. \tag{3.2}$$

Again, we assume that we are given bounds B_H, B_T, and B_N on $\max_i \operatorname{size}(c_i)$, t, and $\log_2 \deg f$, respectively. We also introduce the notation $\tau(f)$, which is defined to be the number of distinct non-zero, non-constant terms in the univariate polynomial f.

This algorithm will again use the polynomials $f^{(p)}$ for primes p, but now rather than a degree condition, we need $f^{(p)}$ to have the maximal number of non-constant terms. So we define a prime p to be a *good prime for interpolation* if and only if $\tau(f^{(p)}) = t$. Again, the term "good prime" refers to this kind of prime for the remainder of this section.

Now suppose we have used modular evaluation and dense interpolation (as in Algorithm 2.3) to recover the polynomials $f^{(p)}$ for k distinct good primes $p_1, \ldots, p_k$. We therefore have k images of each exponent e_i modulo $(p_1 - 1), \ldots, (p_k - 1)$. Write each of these polynomials as:

$$f^{(p_i)} = c_0^{(i)} + c_1^{(i)} x^{e_1^{(i)}} + \cdots + c_t^{(i)} x^{e_t^{(i)}}\,. \tag{3.3}$$

Note that it is *not* generally the case that $e_j^{(i)} = e_j \operatorname{rem}_1 (p_i - 1)$. Because we don't know how to associate the exponents in each polynomial $f^{(p_i)}$ with their pre-image in $\mathbb{Z}$, a simple Chinese remaindering on the exponents will not work. Possible approaches are provided by Kaltofen (1988), Kaltofen *et al.* (1990) or Avendaño *et al.* (2006). However, the most suitable approach for our purposes is the clever technique of Garg & Schost (2009), based on ideas of Grigoriev & Karpinski (1987). We interpolate the polynomial

$$g(z) = (z - e_1)(z - e_2) \cdots (z - e_t)\,, \tag{3.4}$$

whose coefficients are symmetric functions in the e_i's. Given $f^{(p_i)}$, we have all the values of $e_j^{(i)} \operatorname{rem}_1 (p_i - 1)$ for $j = 1, \ldots, t$; we just don't know the order. But since g is not dependent on the order, we can compute $g \bmod (p_i - 1)$ for $i = 1, \ldots, k$, and then find the roots of $g \in \mathbb{Z}[x]$ to determine the exponents $e_1, \ldots, e_t$. Once we know the exponents, we recover the coefficients from their images modulo each prime. The correct coefficient in each $f^{(p)}$ can be identified because the residues of the exponents modulo $p-1$ are unique, for each chosen prime p. This approach is made explicit in the following algorithm.

ALGORITHM 3.5. Sparse Polynomial Interpolation over $\mathbb{Q}[x]$.

Input:
- A modular black box for unknown $f \in \mathbb{Q}[x]$
- Bounds B_H and B_N as described above
- An oracle which produces primes and indicates when at least one good prime must have been returned

Output: $f \in \mathbb{Q}[x]$ as in (3.1)

1. $Q \leftarrow 1, \quad P \leftarrow 1, \quad k \leftarrow 1, \quad t \leftarrow 0$
2. While $\log_2 P < 2B_H + 1$ or $\log_2 Q < B_N$ or the oracle does not guarantee a good prime has been produced do 3–8
3. $\quad p_k \leftarrow$ new prime from the oracle
4. $\quad$ Compute $f^{(p_k)}$ by black box calls and dense interpolation
5. $\quad$ If $\tau(f^{(p_k)}) > t$ then
6. $\quad\quad Q \leftarrow p_k - 1$, $P \leftarrow p_k$, $t \leftarrow \tau(f^{(p_k)})$, $p_1 \leftarrow p_k$, $f^{(p_1)} \leftarrow f^{(p_k)}$, $k \leftarrow 2$
7. $\quad$ Else if $\tau(f^{(p_k)}) = t$ then
8. $\quad\quad Q \leftarrow \operatorname{lcm}(Q, p_k - 1), \quad P \leftarrow P \cdot p_k, \quad k \leftarrow k + 1$
9. For $i \in \{1, \ldots, k-1\}$ do
10. $\quad g^{(p_i)} \leftarrow \prod_{1 \le j \le t} (z - e_j^{(i)}) \mod p_i - 1$
11. Construct $g = a_0 + a_1 z + a_2 z^2 + \cdots + a_t z^t \in \mathbb{Z}[x]$ such that $g \equiv g^{(p_i)} \bmod p_i - 1$ for $1 \le i < k$, by Chinese remaindering
12. Factor g as $(z - e_1)(z - e_2) \cdots (z - e_t)$ to determine $e_1, \ldots, e_t \in \mathbb{Z}$
13. For $1 \le i \le t$ do
14. $\quad$ For $1 \le j \le k$ do
15. $\quad\quad$ Find the exponent $e_{\ell_j}^{(j)}$ of $f^{(p_j)}$ such that $e_{\ell_j}^{(j)} \equiv e_i \bmod p_j - 1$
16. $\quad$ Reconstruct $c_i \in \mathbb{Q}$ by Chinese remaindering from residues $c_{\ell_1}^{(1)}, \ldots, c_{\ell_k}^{(k)}$
17. Reconstruct $c_0 \in \mathbb{Q}$ by Chinese remaindering from residues $c_0^{(1)}, \ldots, c_0^{(k)}$

The following theorem follows from the above discussion.

THEOREM 3.6. *Algorithm 3.5 works correctly as stated.*

Again, this algorithm runs in polynomial time in the bounds given, but we postpone the detailed complexity analysis until Section 5.2, after we discuss how to choose primes from the "oracle". Some small practical improvements may be gained if we use Algorithm 3.5 to interpolate $f(x + \alpha)$ after running Algorithm 2.3 to determine the sparsest shift α, since in this case we will have a few previously-computed polynomials $f^{(p)}$. However, we do not explicitly

consider this savings in our analysis, as there is not necessarily any asymptotic gain.

Now we just need to analyze the conditions for primes p to be good. This is quite similar to the analysis of the sparsest shift algorithm above, so we omit many of the details here.

THEOREM 3.7. *Let f, B_T, B_H, B_N be as above. There exist $C_1, C_2 \in \mathbb{N}$ with $\log_2 C_1 \leq 2B_H B_T$ and $\log_2 C_2 \leq \frac{1}{2} B_N B_T (B_T - 1)$ such that $\tau(f^{(p)})$ is maximal whenever $p \nmid C_1$ and $(p-1) \nmid C_2$.*

PROOF. Let f be as in (3.1), write $|c_i| = a_i/b_i$ in lowest terms for $i = 1, \ldots, t$, and define

$$C_1 = \prod_{i=1}^{t} a_i b_i \,, \quad C_2 = \prod_{i=1}^{t} \prod_{j=i+1}^{t} (e_j - e_i) \,.$$

Now suppose p is a prime such that $p \nmid C_1$ and $(p-1) \nmid C_2$. From the first condition, we see that each $c_i \bmod p$ is well-defined and nonzero, and so none of the terms of $f^{(p)}$ vanish. Furthermore, from the second condition, $e_i \not\equiv e_k \bmod p-1$ for all $i \neq j$, so that none of the terms of $f^{(p)}$ collide. Therefore $f^{(p)}$ contains exactly t non-constant terms. The bounds on C_1 and C_2 follow from the facts that each $a_i, b_i \leq 2^{B_H}$ and each difference of exponents is at most 2^{B_N}. □

4. Generating primes

We now turn our attention to the problem of generating primes for the sparsest shift and interpolation algorithms. In previous sections we assumed we had an "oracle" for this, but now we present an explicit and analyzed algorithm.

The definition of a "good prime" is not the same for the algorithms in Section 2 and Section 3. However, Corollary 2.8 and Theorem 3.7 provide a unified presentation of sufficient conditions for primes being "good". Here we call a prime which satisfies those sufficient conditions a *useful prime*. So every useful prime is good (with the bounds appropriately specified for the relevant algorithm), but some good primes might not be useful.

We first describe a set $\mathcal{P}$ of primes such that the number and density of useful primes within the set is sufficiently high. We will assume that there exist numbers C_1, C_2, and useful primes p are those such that $p \nmid C_1$ and $(p-1) \nmid C_2$. The numbers C_1 and C_2 will be unknown, but we will assume we are given bounds β_1, β_2 such that $\log_2 C_1 \leq \beta_1$ and $\log_2 C_2 \leq \beta_2$. Suppose we want to

find ℓ useful primes. We construct $\mathcal{P}$ explicitly, of a size guaranteed to contain enough useful primes, then enumerate it.

The following fact is immediate from Mikawa (2001), though it has been somewhat simplified here, and the use of (unknown) constants is made more explicit. This will be important in our computational methods.

For $q \in \mathbb{Z}$, let $S(q)$ be the smallest prime p such that $q \mid (p-1)$.

FACT 4.1 (Mikawa 2001). *There exists a constant $\mu > 0$, such that for all $n > \mu$, and for all integers $q \in \{n, \ldots, 2n\}$ with fewer than $\mu n/\log^2 n$ exceptions, we have $S(q) < q^{1.89}$.*

Our algorithms for generating useful primes require explicit knowledge of the value of the constant μ in order to run correctly. So we will assume that we know μ in what follows. To get around the fact that we do not, we simply start by assuming that $\mu = 1$, and run any algorithm depending upon it. If the algorithm fails we simply double our estimate for μ and repeat. At most a constant number of doublings is required. We make no claim this is particularly practical.

For convenience we define

$$\Upsilon(x) = \frac{3x}{5\log x} - \frac{\mu x}{\log^2 x}\,.$$

THEOREM 4.2. *Let $\log_2 C_1 \le \beta_1$, $\log_2 C_2 \le \beta_2$ and ℓ be as above. Let n be the smallest integer such that $n > 21$, $n > \mu$ and $\Upsilon(n) > \beta_1 + \beta_2 + \ell$. Define*

$$\mathcal{Q} = \left\{q \text{ prime} : n \le q < 2n \text{ and } S(q) < q^{1.89}\right\}, \quad \mathcal{P} = \left\{S(q) : q \in \mathcal{Q}\right\}.$$

Then the number of primes in $\mathcal{P}$ is at least $\beta_1 + \beta_2 + \ell$, and the number of useful primes in $\mathcal{P}$, such that $p \nmid C_1$ and $(p-1) \nmid C_2$, is at least ℓ. For all $p \in \mathcal{P}$ we have $p \in O((\beta_1 + \beta_2 + \ell)^{1.89} \cdot \log^{1.89}(\beta_1 + \beta_2 + \ell))$.

PROOF. By Rosser & Schoenfeld (1962), the number of primes between n and $2n$ is at least $3n/(5\log n)$ for $n \ge 21$. Applying Fact 4.1, we see $\#\mathcal{Q} \ge 3n/(5\log n) - \mu n/\log^2 n$ when $n \ge \max\{\mu, 21\}$. Now suppose $S(q_1) = S(q_2)$ for $q_1, q_2 \in \mathcal{Q}$. If $q_1 < q_2$, then $S(q_1) > q_1^2$, a contradiction with the definition of $\mathcal{Q}$. So we must have $q_1 = q_2$, and hence

$$\#\mathcal{P} = \#Q \ge \Upsilon(n) > \beta_1 + \beta_2 + \ell\,.$$

We know that there are at most $\log_2 C_1 \le \beta_2$ primes $p \in \mathcal{P}$ such that $p \mid C_1$. We also know that there are at most $\log_2 C_2 \le \beta_2$ primes $q \in \mathcal{Q}$ such that $q \mid C_2$,

and hence at most $\log_2 C_2$ primes $p \in \mathcal{P}$ such that $p = S(q)$ and $q \mid (p-1) \mid C_1$. Thus, by construction $\mathcal{P}$ contains at most $\beta_1 + \beta_2$ primes that are not useful out of $\beta_1 + \beta_2 + \ell$ total primes.

To analyze the size of the primes in $\mathcal{P}$, we note that to make $\Upsilon(n) > \beta_1 + \beta_2 + \ell$, we have $n \in \Theta((\beta_1 + \beta_2 + \ell) \cdot \log(\beta_1 + \beta_2 + \ell))$ and each $q \in \mathcal{Q}$ satisfies $q \in O(n)$. Elements of $\mathcal{P}$ will be of magnitude at most $(2n)^{1.89}$ and hence $p \in O((\beta_1 + \beta_2 + \ell)^{1.89} \log^{1.89}(\beta_1 + \beta_2 + \ell))$. □

Given β_1, β_2 and ℓ as above (where $\log_2 C_1 \leq \beta_1$ and $\log_2 C_2 \leq \beta_2$ for unknown C_1 and C_2), we generate the primes in $\mathcal{P}$ as follows.

Start by assuming that $\mu = 1$, and compute n as the smallest integer such that $\Upsilon(n) > \beta_1 + \beta_2 + \ell$, $n \geq \mu$ and $n \geq 21$. List all primes between n and $2n$ using a Sieve of Eratosthenes. For each prime q between n and $2n$, determine $S(q)$, if it is less than $q^{1.89}$, by simply checking if $kq + 1$ is prime for $k = 1, 2, \ldots, \lfloor q^{0.89} \rfloor$. If we find a prime $p = S(q) < q^{1.89}$, add p to $\mathcal{P}$. This is repeated until $\mathcal{P}$ contains $\beta_1 + \beta_2 + \ell$ primes. If we are unable to find this number of primes, we have underestimated μ (since Theorem 4.2 guarantees their existence), so we double μ and restart the process. Obviously in practice we would not redo primality tests already performed for smaller μ, so really no work need be wasted.

THEOREM 4.3. *For $\log_2 C_1 \leq \beta_1$, $\log_2 C_2 \leq \beta_2$, ℓ, and n as in Theorem 4.2, we can generate $\beta_1+\beta_2+\ell$ elements of $\mathcal{P}$ with $O((\beta_1+\beta_2+\ell)^2 \cdot \log^{7+o(1)}(\beta_1+\beta_2+\ell))$ bit operations. At least ℓ of the primes in $\mathcal{P}$ will be useful.*

PROOF. The method and correctness follows from the above discussion. The Sieve of Eratosthenes can be run with $O(n \log\log\log n)$ bit operations (see Knuth (1981), Section 4.5.4), and returns $O(n/\log n)$ primes q between n and $2n$. Each primality test of $kq + 1$ can be done with $(\log n)^{6+o(1)}$ bit operations (Lenstra & Pomerance 2005), so the total cost is $O(n^2(\log n)^{5+o(1)})$ bit operations. Since $n \in O((\beta_1 + \beta_2 + \ell) \cdot \log(\beta_1 + \beta_2 + \ell))$ the stated complexity follows. □

The analysis of our methods will be significantly improved when more is discovered about the behavior of the least prime congruent to one modulo a given prime, which we have denoted $S(q)$. An asymptotic lower bound of $S(q) \in \Omega(q \log^2 q)$ is conjectured in Granville & Pomerance (1990), and we have employed the upper bound from Mikawa (2001) of $S(q) \in O(q^{1.89})$ (with exceptions). From our own brief computational search we have evidence that the conjectured lower bound may well be an upper bound: for all primes $q \leq$

2^{32}, $S(q) < 2q \ln^2 q$. If something similar could be proven to hold asymptotically (even with some exceptions), the complexity results of this and the next section would be improved significantly. In any case, the actual cost of the algorithms discussed will be a reflection of the true behavior of $S(q)$, even before it is completely understood by us.

Even more improvements might be possible if this rather complicated construction is abandoned altogether, as useful primes would naturally seem to be relatively plentiful. In particular, one would expect that if we randomly choose primes p directly from a set which has, say, $4(\beta_1 + \beta_2 + \ell)$ primes, we might expect that the probability that $p \,|\, C_1$ or $(p-1) \,|\, C_2$ to less than, say, $1/4$. Proving this directly appears to be difficult. Perhaps most germane results to this are lower bounds on the Carmichael Lambda function (which for the product of distinct primes $p_1, \ldots, p_m$ is the LCM of $p_1 - 1, \ldots, p_m - 1$), which are too weak for our purposes. See Erdös *et al.* (1991).

5. Complexity analysis

We are now ready to give a formal complexity analysis for the algorithms presented in Section 2 and Section 3. For all algorithms, the complexity is polynomial in the four bounds B_A, B_T, B_H, and B_N defined in Section 2.2, and since these are each bounded above by size(f), our algorithms will have polynomial complexity in the size of the output if these bounds are sufficiently tight.

5.1. Complexity of sparsest shift computation. Algorithm 2.3 gives our algorithm to compute the sparsest shift α of an unknown polynomial $f \in \mathbb{Q}[x]$ given bounds B_A, B_T, B_H, and B_N and an oracle for choosing primes. The details of this oracle are given in Section 4.

To choose primes, we set $\ell = 2B_A + 1$, and $\beta_1 = 2B_H$ and $\beta_2 = B_N(3B_T - 1)$ (according to Corollary 2.8). For the sake of notational brevity, define $B_\Sigma = B_A + B_H + B_N B_T$ so that $\beta_1 + \beta_2 + \ell \in O(B_\Sigma)$.

THEOREM 5.1. *Suppose $f \in \mathbb{Q}[x]$ is an unknown polynomial given by a black box, with bounds B_A,B_T,B_H, and B_N given as above. If $\deg f > 2B_T$, then the sparsest shift $\alpha \in \mathbb{Q}$ of f can be computed deterministically using*

$$O\big(B_A B_\Sigma^{1.89} \cdot \log^{2.89} B_\Sigma \cdot \mathsf{M}(B_\Sigma^{1.89} \log^{1.89} B_\Sigma) \cdot \mathsf{M}(\log B_\Sigma)\big)$$

bit operations, plus $O(\kappa_f B_\Sigma^{2.89} \log^{1.89} B_\Sigma \mathsf{M}(\log B_\Sigma))$ bit operations for the black-box evaluations.

PROOF. Algorithm 2.3 will always terminate (by satisfying the conditions of Step 2) after $2B_A + 1$ good primes have been produced by the oracle.

Using the oracle to choose primes, and because $\beta_1 + \beta_2 + \ell \in O(B_\Sigma)$, $O(B_\Sigma^2 \log^{7+o(1)} B_\Sigma)$ bit operations are used to compute all the primes on Step 3, by Theorem 4.3. And by Theorem 4.2, each chosen p is bounded by $O(B_\Sigma^{1.89} \log^{1.89} B_\Sigma)$.

All black-box evaluations are performed on Step 4; there are p evaluations at each iteration, and $O(B_\Sigma)$ iterations, for a total cost of $O(\kappa_f B_\Sigma p \cdot \mathsf{M}(\log p))$ bit operations. The stated complexity bound follows from the size of each prime p.

Steps 10–14 are never executed when $\deg f > 2B_T$. Step 15 is only executed once and never dominates the complexity.

Dense polynomial interpolation over $\mathbb{Z}_p$ is performed at most $O(B_\Sigma)$ times on Step 5 and $O(p)$ times at each of $O(B_A)$ iterations through Step 7. Since $p \gg B_\Sigma$, the latter step dominates. Using asymptotically fast methods, each interpolation of $f^{(p)}(x + \gamma)$ uses $O(\mathsf{M}(p) \log p)$ field operations in $\mathbb{Z}_p$, each of which costs $O(\mathsf{M}(\log p))$ bit operations. This gives a total cost over all iterations of $O(B_A p \cdot \log p \cdot \mathsf{M}(p) \cdot \mathsf{M}(\log p))$ (a slight abuse of notation here since the value of p varies). Again, using the fact that $p \in O(B_\Sigma^{1.89} \log^{1.89} B_\Sigma)$ gives the stated result. □

To simplify the discussion somewhat, consider the case that we have only a *single* bound on the size of the output polynomial, say $B_f \geq \text{size}(f)$. By setting each of B_T, B_H, and B_N equal to B_f, and by using the multiplication algorithm from Cantor & Kaltofen (1991), we obtain the following comprehensive result:

COROLLARY 5.2. *The sparsest shift α of an unknown polynomial $f \in \mathbb{Q}[x]$, whose shifted-lacunary size is bounded by B_f, can be computed using*

$$O\big(B_f^{8.56} \cdot \log^{6.78} B_f \cdot (\text{loglog}\, B_f)^2 \cdot \text{logloglog}\, B_f\big)$$

bit operations, plus

$$O\big(\kappa_f B_f^{5.78} \cdot \log^{2.89} B_f \cdot \text{loglog}\, B_f \cdot \text{logloglog}\, B_f\big)$$

bit operations for the black-box evaluations.

PROOF. The stated complexities follow directly from Theorem 5.1 above, using the fact that $\mathsf{M}(n) \in O(n \log n \,\text{loglog}\, n)$ and $B_\Sigma \in O(B_f^2)$. Using the single bound B_f, we see that these costs always dominate the cost of Steps 10–14 given in Theorem 2.5, and so we have the stated general result. □

In fact, if we have no bounds at all *a priori*, we could start by setting B_f to some small value (perhaps dependent on the size of the black box or κ_f), running Algorithm 2.3, then doubling B_f and running the algorithm again, and so forth until the same polynomial f is computed in successive iterations. This can then be tested on random evaluations. Such an approach yields an output-sensitive polynomial-time algorithm which should be correct on most input, though it could certainly be fooled into early termination.

This is a significant improvement over the algorithms from our original paper (Giesbrecht & Roche 2007), which had a dominating factor of B_f^{78} in the deterministic complexity. Also – and somewhat surprisingly – our algorithm is competitive even with the best-known sparsest shift algorithms which require a (dense) $f \in \mathbb{Q}[x]$ to be given explicitly as input. By carefully constructing the modular black box from a given $f \in \mathbb{Q}[x]$, and being sure to set $B_T < (\deg f)/2$, we can derive from Algorithm 2.3 a deterministic sparsest-shift algorithm with bit complexity close to the fastest algorithms in Giesbrecht *et al.* (2003); the dependence on degree n and sparsity t will be somewhat less, but the dependence on the size of the coefficients $\log \|f\|$ is greater.

To understand the limits of our computational techniques (as opposed to our current understanding of the least prime in arithmetic progressions) we consider the cost of our algorithms under the optimistic assumption that $S(q) \in O(q \ln^2 q)$, possibly with a small number of exceptions. In this case the sparsest shift α of an unknown polynomial $f \in \mathbb{Q}[x]$, whose shifted-lacunary size is bounded by B_f, can be computed using

$$O\big(B_f^5 \cdot \log^6 B_f \cdot (\operatorname{loglog} B_f)^2 \cdot \operatorname{logloglog} B_f\big)$$

bit operations. As noted in the previous section, we have verified computationally that $S(q) \leq 2q \ln^2 q$ for $q < 2^{32}$. This would suggest the above complexity for all sparsest-shift interpolation problems that we would expect to encounter.

5.2. Complexity of interpolation. The complexity analysis of the sparse interpolation algorithm given in Algorithm 3.5 will be quite similar to that of the sparsest shift algorithm above. Here, we need $\ell = \max\{2B_H + 1, B_N\}$ good primes to satisfy the conditions of Step 2, and from Theorem 3.7, we set $\beta_1 = 2B_H B_T$ and $\beta_2 = \frac{1}{2} B_N B_T (B_T - 1)$. Hence for this subsection we set $B_S = B_T(B_H + B_N B_T)$ so that $\beta_1 + \beta_2 + \ell \in O(B_S)$.

THEOREM 5.3. *Suppose $f \in \mathbb{Q}[x]$ is an unknown polynomial given by a modular black box, with bounds B_T, B_H, B_N, and B_S given as above. The sparse*

representation of f as in (1.3) can be computed with

$$O\Big(B_S \log B_S \cdot \mathsf{M}(B_S^{1.89} \log^{1.89} B_S) \cdot \mathsf{M}(\log B_S) + B_N^2 \cdot \mathsf{M}\big((B_N + \log B_T) \log(B_N + \log B_T)\big)\Big)$$

bit operations, plus $O(\kappa_f B_S^{2.89} \log^{1.89} B_S \mathsf{M}(\log B_S))$ bit operations for the black-box evaluations.

Proof. As in the sparsest-shift computation, the cost of choosing primes in Step 3 requires $O(B_S^2 \log^{7+o(1)} B_S)$ bit operations, and each chosen prime p_k satisfies $p_k \in O(B_S^{1.89} \log^{1.89} B_S)$. The total cost over all iterations of Step 4 is also similar to before, $O(B_S \cdot \mathsf{M}(p_k) \log p_k \cdot \mathsf{M}(\log p_k))$ bit operations, plus $O(\kappa_f B_S p \mathsf{M}(\log p_k))$ for the black-box evaluations.

We can compute each $g^{(p_i)}$ in Step 10 using $O(\mathsf{M}(t) \log t)$ ring operations modulo $p_i - 1$. Note that $k \in O(\ell)$, which is $O(B_H + B_N)$, so the total cost in bit operations for all iterations of this step is $O((B_H + B_N) \log B_T \cdot \mathsf{M}(B_T) \cdot \mathsf{M}(\log B_S))$.

Step 11 performs t Chinese Remainderings each of k modular images, and the size of each resulting integer is bounded by 2^{B_N}, for a cost of $O(B_T \log B_N \cdot \mathsf{M}(B_N))$ bit operations.

To factor g in Step 12, we can use Algorithm 14.17 of von zur Gathen & Gerhard (2003), which has a total cost in bit operations of

$$O\Big(B_T^2 \cdot \mathsf{M}(B_N + \log B_T) + B_N(B_N + \log B_T) \cdot \mathsf{M}\big((B_N + \log B_T) \log(B_N + \log B_T)\big)\Big)$$

because the degree of g is t, g has t distinct roots, and each coefficient is bounded by 2^{B_N}.

In Step 15, we must first compute the modular image of $e_i \bmod p_j - 1$ and then look through all t exponents of $f^{(p_j)}$ to find a match. This is repeated tk times. We can use fast modular reduction to compute all the images of each e_i using $O(\mathsf{M}(B_N) \log B_N)$ bit operations, so the total cost is $O(B_T(B_H B_T + B_N B_T + \mathsf{M}(B_N) \log B_N))$ bit operations.

Finally, we perform Chinese remaindering and rational reconstruction of $t + 1$ rational numbers, each of whose size is bounded by B_H, for a total cost of $O(B_T \cdot \mathsf{M}(B_H) \log B_H)$.

Therefore we see that the complexity is dominated either by the dense interpolation in Step 4 or the root-finding algorithm in Step 12, depending essentially on whether B_N dominates the other bounds. □

Once again, by having only a single bound on the size of the output, the complexity measures are greatly simplified.

Corollary 5.4. *Given a modular black box for an unknown polynomial $f \in \mathbb{Q}[x]$ and a bound B_f on the size of its lacunary represenation, that representation can be interpolated using*

$$O\big(B_f^{8.67} \log^{4.89} B_f (\operatorname{loglog} B_f)^2 \operatorname{logloglog} B_f\big)$$

bit operations, plus

$$O\big(\kappa_f B_f^{8.67} \log^{2.89} B_f \operatorname{loglog} B_f \operatorname{logloglog} B_f\big)$$

bit operations for the black-box evaluations.

Similar improvements to those discussed at the end of Section Section 5.1 can be obtained under stronger (but unproven) number theoretic assumptions.

6. Conclusions and future work

Here we provide the first algorithm to interpolate an unknown univariate rational polynomial into the sparsest shifted power basis in time polynomial in the size of the output. The main tool we have introduced is mapping down modulo small primes where the sparse shift is also mapped nicely. This technique could be useful for other problems involving lacunary polynomials as well, although it is not clear how it would apply in finite domains where there is no notion of "size".

There are many further avenues to consider, the first of which might be multivariate polynomials with a shift in each variable (see, e.g., Grigoriev & Lakshman (2000)). It would be easy to adapt our algorithms to this case provided that the degree in *each variable* is more than twice the sparsity (this is called a "very sparse" shift in Giesbrecht *et al.* (2003)). Finding multivariate shifts in the general case seems more difficult. Even more challenging would be allowing multiple shifts, for one or more variables – for example, finding sparse $g_1, \ldots, g_k \in \mathbb{Q}[x]$ and shifts $\alpha_1, \ldots, \alpha_k \in \mathbb{Q}$ such that the unknown polynomial $f(x)$ equals $g_1(x - \alpha_1) + \cdots + g_k(x - \alpha_k)$. The most general problem of this type, which we are very far from solving, might be to compute a minimal-length formula or minimal-size algebraic circuit for an unknown function. We hope that the small step taken here might provide some insight towards this ultimate goal.

Acknowledgements

The authors would like to thank Igor Shparlinski for pointing out the paper of Mikawa (2001), and for suggesting how to discard "exceptional" primes q. This avoids the use of Linnik's theorem, as employed in Giesbrecht & Roche (2007), and improves the complexity considerably.

The authors would also like to thank Erich Kaltofen for discussions and sharing of his early unpublished work on rational interpolation, and Éric Schost for discussions and sharing a pre-print of Garg & Schost (2009).

Finally, the authors would like to thank the anonymous reviewers for their careful readings and useful suggestions.

An extended abstract of a preliminary version of this work appeared at the MACIS 2007 conference (Giesbrecht & Roche 2007).

References

M. Avendaño, T. Krick & A. Pacetti, Newton-hensel interpolation lifting. *Foundations of Computational Mathematics* (2006), 81–120.

M. Ben-Or & P. Tiwari, A deterministic algorithm for sparse multivariate polynomial interpolation. In *STOC '88: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, New York, NY, USA, 1988, ACM Press, 301–309.

M. Bläser, M. Hardt, R. J. Lipton & N. K. Vishnoi, Deterministically testing sparse polynomial identities of unbounded degree. *Inf. Process. Lett.* **109**(3) (2009), 187–192.

A. Borodin & I. Munro, *The Computational Complexity of Algebraic and Numeric Problems*. American Elsevier Publishing Co., Inc., New York, London, Amsterdam, 1975. Elsevier Computer Science Library; Theory of Computation Series, No. 1.

A. Borodin & P. Tiwari, On the decidability of sparse univariate polynomial interpolation. *Comput. Complexity* **1**(1) (1991), 67–90.

D. Cantor & E. Kaltofen, Fast multiplication of polynomials over arbitrary algebras. *Acta Informatica* **28** (1991), 693–701.

P. Erdös, C. Pomerance & E. Schmutz, Carmichael's lambda function. *Acta Arithmetica* **58** (1991), 363–385.

S. Garg & É. Schost, Interpolation of polynomials given by straight-line programs. *Theoretical Computer Science* **410**(27–29) (2009), 2659–2662.

J. von zur Gathen & J. Gerhard, *Modern Computer Algebra*. Cambridge University Press, Cambridge, second edition, 2003.

M. Giesbrecht & D. Roche, Interpolation of shifted-lacunary polynomials. In *Proc. Mathematical Aspects of Computer and Information Sciences (MACIS'07)*, Paris, France, 2007.

M. Giesbrecht, E. Kaltofen & Wen-shin Lee, Algorithms for computing sparsest shifts of polynomials in power, Chebyshev and Pochhammer bases. *J. Symbolic Comput.* **36**(3–4) (2003), 401–424. International Symposium on Symbolic and Algebraic Computation (ISSAC'2002) (Lille).

M. Giesbrecht, G. Labahn & Wen-shin Lee, Symbolic-numeric sparse interpolation of multivariate polynomials. In *Proc. ACM International Symposium on Symbolic and Algebraic Computation (ISSAC)*, 2006, 116–123.

A. Granville & C. Pomerance, On the least prime in certain arithmetic progressions. *J. London Math. Soc.* **s2-41**(2) (1990), 193–200.

D. Grigoriev & M. Karpinski, The matching problem for bipartite graphs with polynomially bounded permanents is in NC. In *Foundations of Computer Science (FOCS)*, 1987, 166–172.

D. Grigoriev & M. Karpinski, A zero-test and an interpolation algorithm for the shifted sparse polynomials. In *Applied Algebra, Algebraic Algorithms and Error-correcting Codes (San Juan, PR, 1993)*, vol. 673 of *Lecture Notes in Comput. Sci.*, 162–169. Springer, Berlin, 1993.

D. Grigoriev & Y. Lakshman, Algorithms for computing sparse shifts for multivariate polynomials. *Appl. Algebra Engrg. Comm. Comput.* **11**(1) (2000), 43–67.

E. Kaltofen, Notes on polynomial and rational function interpolation. (1988). Unpublished manuscript.

E. Kaltofen & Wen-shin Lee, Early termination in sparse interpolation algorithms. *J. Symbolic Comput.* **36**(3–4) (2003), 365–400. International Symposium on Symbolic and Algebraic Computation (ISSAC'2002) (Lille).

E. Kaltofen, Y. N. Lakshman & J.-M. Wiley, Modular rational sparse multivariate polynomial interpolation. In *ISSAC '90: Proceedings of the International Symposium on Symbolic and Algebraic Computation*, New York, NY, USA, 1990, ACM Press, 135–139.

D. E. Knuth, *The Art of Computer Programming, Vol. 2, Seminumerical Algorithms*. Addison-Wesley, Reading MA, 2 edition, 1981.

Y. N. Lakshman & B. D. Saunders, Sparse shifts for univariate polynomials. *Appl. Algebra Engrg. Comm. Comput.* **7**(5) (1996), 351–364.

H. W. Lenstra, Jr. & C. Pomerance, Primality testing with Gaussian periods. Preprint, 2005.

H. Mikawa, On primes in arithmetic progressions. *Tsukuba journal of mathematics* **25**(1) (2001), 121–153.

J. Barkley Rosser & L. Schoenfeld, Approximate formulas for some functions of prime numbers. *Illinois J. Math.* **6** (1962), 64–94.

Manuscript received 31 October 2008

Mark Giesbrecht
School of Computer Science
University of Waterloo
Waterloo, ON, N2L 3G1, Canada
mwg@cs.uwaterloo.ca
http://www.cs.uwaterloo.ca/~mwg

Daniel S. Roche
School of Computer Science
University of Waterloo
Waterloo, ON, N2L 3G1, Canada
droche@cs.uwaterloo.ca
http://www.cs.uwaterloo.ca/~droche