New Algorithms for Lacunary (Supersparse) Polynomials
Sparsest Shift Interpolation and Sparse Functional Decomposition

Daniel Roche    Mark Giesbrecht

University of Waterloo

31 May 2007
Representation of Sparse Polynomials

Let $F$ be a field and $f(x) \in F[x]$ of degree $n$.

$f(x)$ in *dense* form is

$$f(x) = f_0 + f_1x + f_2x^2 + \cdots + f_nx^n.$$  

$f(x)$ in *sparse* form is

$$f(x) = a_1x^{e_1} + a_2x^{e_2} + \cdots + a_tx^{e_t}$$

- $a_i$’s nonzero in $F$
- $e_i$’s in $\mathbb{Z}$ with $e_1 < e_2 < \ldots < e_t = n$
- $t$ is the *sparsity* of $f(x)$

Sparse size is $\sum_{i=1}^t (\text{size}(a_i) + \log e_i)$

- Can be exponentially smaller than the dense size
- This representation is the default in Maple, Mathematica, etc.
Alternate Notions of Sparsity

**Definition (Sparse Shifts)**

If $f(x)$ has at most $t$ nonzero terms in the shifted power basis $1, (x - \alpha), (x - \alpha)^2, \ldots$, for some $\alpha \in F$, then we say $\alpha$ is a $t$-sparse shift for $f(x)$.

**Theorem (Lakshman & Saunders [LS96])**

If $t \leq \frac{d+1}{2}$, then there is at most one $t$-sparse shift for any polynomial $f(x) \in F[x]$.

**Definition (Black Box)**

A black box for a polynomial $f(x) \in F[x]$ is a procedure which, when given any element $\theta \in F$, returns the value of $f(\theta)$.
Basic Arithmetic  Addition, subtraction, multiplication

Division  Euclidian division (quotient and remainder) — polynomial time in the size of the input and output polynomials.

Interpolation  Determine a sparse polynomial from its black box, provided we can compute logarithms in F [BOT88, KL03]

Root Finding  Find all distinct low-degree factors [CKS99, Len99]
Exponentiation  Raising a sparse polynomial to the $r$’th power could increase the size of the output by an exponential factor.

Factorization  The factors could be dense, meaning the operation could be exponential.

GCD  Provably hard to determine even if the GCD of two sparse polynomials is 1 [Pla84]

Divisibility?  A polynomial-time divisibility test for sparse polynomials is a basic open question in this area.
Sparsest Shift Interpolation

First polynomial-time algorithm to compute sparsest shift $\alpha$ of $f(x)$ given in [LS96], later improved in [GKL03].

- Requires that $f(x)$ be given explicitly in dense form.

**Question**

Can we find the sparsest shift $\alpha$ of a polynomial $f(x) \in F[x]$, given a black box for $f(x)$ and using time polynomial in the size of the sparsest shift?

We have a solution to a particular instance of this problem: Let $f(x) \in \mathbb{Z}[x]$, and suppose we are given a black box which takes $\theta \in \mathbb{Z}$ and prime $p$, and returns $f(\theta) \mod p$. 
Let $p$ be a prime with $p \geq t^2$.

From Fermat’s Little Theorem,

$$a^{p-1} \equiv 1 \mod p$$
whenver $p \nmid a$.

So $\exists f_p(x) \in \mathbb{Z}_p[x]$ of degree at most $p - 2$
such that $f_p(\theta) \equiv f(\theta) \mod p$ for all $\theta \in \mathbb{Z}$.

If $f(x) = \sum_{i=1}^{t} a_i (x - \alpha)^{e_i}$, then

$$f_p(x) = \sum_{i=1}^{t} (a_i \mod p)(x - (\alpha \mod p))^{e_i} \mod (p-1),$$

and therefore $\alpha$ is a $t$-sparse shift for $f_p(x)$. 

Algorithm: Sparsest Shift Interpolation

1. Choose a prime $p$ from a sufficiently large set such that $t^2 < p < t^{O(1)}$.

2. Use the black box to compute $v_i = f(i) \mod p$ for $i = 1, 2, \ldots, p - 1$.

3. Use (dense) Lagrange interpolation to find $f_p(x)$.

4. If $\deg(f_p(x)) \geq 2t - 1$, then use the algorithm from [GKL03] to find the sparsest shift $\alpha_p$ in $\mathbb{Z}_p$.

5. Repeat $O(\log \alpha)$ times until $\alpha$ can be recovered from the $\alpha_p$’s via Chinese Remaindering.
If \( \deg(f_p(x)) \geq 2t - 1 \), then we know from [LS96] that \( \alpha \mod p \) is the sparsest shift, since it is a \( t \)-sparse shift (from before).

\( \alpha \) must be the root of \( n - t \) derivatives of \( f(x) \).

Roots of any derivative of \( f(x) \) in \( \mathbb{Z} \) are bounded by the maximal and minimal roots of \( f(x) \) itself, which in turn must divide the trailing coefficient of \( f(x) \).

So the size of \( \alpha \) is less than the size of \( f(x) \).

The tricky part of the analysis is constructing the set of primes \( S \) in such a way that \( \deg f_p \geq 2t - 1 \) with high probability (not shown here).

Algorithm runs in polynomial time in the sparse size of \( f(x + \alpha) \).
The problem of (simple) functional decomposition of polynomials is, given \( f(x) \in \mathbb{F}[x] \), find \( g(x), h(x) \in \mathbb{F}[x] \), each with degree at least 2, such that \( f(x) = g(h(x)) \).

**Functional Decomposition Algorithms**

- Univariate [KL89, vzG90]
- Rational functions [Zip91]
- Sparsest complete decomposition [LS96]
- Algebraic functions [KLZ96]
- Multivariate [vzGGR03, FJ06]

All of these algorithms take polynomial time in the degree of \( f \). Can we compute a simple univariate decomposition in polynomial time in the sparse size of \( f \)?
Problem

Given \( f(x) \), find \( g(x) \) and \( h(x) \) such that \( f(x) = g(h(x)) \).

- \( f(x) \) is given in the \( \alpha \)-shifted power basis
- \( g(x) \) is returned in the sparsest shifted power basis, \( \beta \)
- \( h(x) \) is returned in the \( \alpha \)-shifted power basis
- Polynomial time in the size of the input \textit{and} output

Can assume that \( f, g, h \) are all monic and \( \alpha = \beta = 0 \), since

\[
\frac{f(x + \alpha)}{lc(f)} = \left( \frac{g(lc(h)(x + \beta))}{lc(f)} \right) \circ \left( \frac{h(x + \alpha)}{lc(h)} - \beta \right)
\]

(\( lc(f) \) and \( lc(h) \) are leading coefficients of \( f(x) \) and \( h(x) \))
Finding $h(x)$ of low degree

Lemma 2 from [KL89] tells us that $f(x)$ and $h(x)$ agree in their high-order $s$ coefficients. So define $\tilde{f}(x) = x^n f(\frac{1}{x})$ and $\tilde{h}(x) = x^s h(\frac{1}{x})$ to be the reversals of $f(x)$ and $h(x)$, respectively. Then

$$\tilde{f}(x) \equiv \tilde{h}(x)^r \mod x^s. \quad (1)$$

- Uniquely determines $h(x)$ up to the constant term
- Can solve with $O(s^{O(1)})$ field operations, as in [vzG90]

So if $s$ is sufficiently small, we can find it in polynomial time in the sparse size of $f(x)$. 
Certifying low-degree $h$

**Question**

How to efficiently check whether a given $h(x)$ is a right composition factor of $f(x)$?

Let $\Psi_h(x, y) = h(x) - h(y)$ and $\Psi_f(x, y) = f(x) - f(y)$

- $h(x)$ is a right composition factor of $f(x)$ iff $\Psi_h(x, y) \mid \Psi_f(x, y)$ [FM69]
- Note $\Psi_h(x, y)$ does not depend on $h(0)$

[KK05] gives a method to efficiently check whether a low-degree bivariate factor divides a high-degree sparse bivariate polynomial. We can use this method to efficiently (probabilistically) check whether $\Psi_h(x, y) \mid \Psi_f(x, y)$, therefore checking whether the $h(x)$ we have found is correct.
Finding \( h(x) \) of high degree

**Conjecture of Schinzel [Sch87]**

If any power of a polynomial is sparse, then the polynomial itself must also be sparse.

Subject to this conjecture, we can compute \( h(x) \) (up to its constant coefficient) in polynomial time in the size of \( f \) and the size of \( h \), by using a careful Newton-like iteration. Let \( \tilde{h}_1(x) \) and \( \tilde{h}_2(x) \) be polynomials of degree \( k \) and \( l \) such that

\[
\tilde{h}(x) \equiv \tilde{h}_1(x) + \tilde{h}_2(x)x^k \mod x^{k+l},
\]

where \( k, l \in \mathbb{Z} \) with \( 1 \leq l \leq k \) and \( k + l \leq s \). Then, from (1) and the binomial theorem,

\[
\tilde{f}(x) \equiv \tilde{h}_1(x)^r + r\tilde{h}_1(x)^{r-1}\tilde{h}_2(x)x^k \mod x^{k+l}. \tag{2}
\]
Finding $h(x)$ of high degree (2)

Through some careful manipulation, we obtain

$$\tilde{h}_1(x)^{r+1} \equiv \tilde{h}_1(x)\tilde{f}(x) - r\tilde{f}(x)\tilde{h}_2(x)x^k \mod x^{k+l}.$$ 

So $\tilde{h}_1(x)^{r+1} \mod x^{k+l}$ is sparse, and therefore from Shinzel’s conjecture, we can compute it by repeated squaring. Manipulating (1) again, we see that

$$\left(\frac{1}{rx^k}\right) \left(\tilde{h}_1(x)\tilde{f}(x) - \tilde{h}_1(x)^{r+1}\right) \equiv \tilde{f}(x)\tilde{h}_2(x) \mod x^l.$$

We can compute the quotient of the left-hand side divided by $\tilde{f}(x) \mod x^l$ in polynomial time since the quotient, $\tilde{h}_2(x)$, is sparse, and $\tilde{f}(x)$ has constant coefficient 1. Thus we can compute $\tilde{h}_2(x)$ in polynomial time.
Algorithm: Finding high-degree $h(x)$

1. $\tilde{h}_1(x) \leftarrow 1; k \leftarrow 1$
2. $l \leftarrow \min\{k, s - k\}$
3. Perform iteration from before to find $\tilde{h}_2(x)$ of degree $l$
4. $\tilde{h}_1(x) \leftarrow \tilde{h}_1(x) + \tilde{h}_2(x)x^k; k \leftarrow k + l$
5. Repeat steps 2–4 until $k = s$
6. Return $x^s\tilde{h}_1(\frac{1}{x})$

Note:
- $\tilde{h}(x) \equiv 1 \mod x$ since $h(x)$ is monic; this is the starting point for our iteration.
- The last step just computes the reversal of $\tilde{h}_1(x)$ — this can be done “for free”. So the whole algorithm runs in polynomial time in the sparse sizes of $f(x)$ and $h(x)$. 
Finding $g(x)$ when $r$ is small

We now show how to find $g(x)$ when $h(x) - h(0)$ is known, using dense interpolation.

1. Choose $r + 1$ distinct points $\theta_0, \ldots, \theta_r \in F$
2. Compute $u_i = h(\theta_i) - h(0)$ and $v_i = f(\theta_i)$ for $i = 0, \ldots, r$
3. Use Lagrange interpolation to compute $g(x + h(0))$.
4. Use the sparsest shift algorithm of [GKL03] to find $h(0)$, and finally compute $g(x)$ and $h(x)$.

We need the $u_i$’s to all be distinct; the Schwartz-Zippell Lemma guarantees this with high probability if the $\theta_i$’s are chosen from a large enough set.
Future Work

- Using our sparsest shift interpolation algorithm to find $g(x)$ of high degree given $h(x)$
- Extending the sparsest shift interpolation algorithm to work over fields other than $\mathbb{Z}[x]$
- Eliminating the dependency of the algorithm for finding high-degree $h(x)$ on any conjectures
- Removing the output-sensitivity of the runtime (i.e. proving that $h(x)$ and $g(x)$ are always sparse when $f(x)$ is sparse) — relates to [Erd49, CD91, Abb02]
References I


