Some dense thoughts on sparse polynomials

Mark Giesbrecht

August 25, 2009
Computer algebra understands (dense) polynomials

Computer algebra has changed mathematics and computer science in its computations with polynomials

\[ f = x^{88}y^{68} + 4x^{67}y^{49} + 4x^{46}y^{30} - 10x^{65}y^{53} - 40x^{44}y^{34} - 40x^{23}y^{15} + 25x^{42}y^{38} + 100x^{21}y^{19} + 100 \]

We have extremely effective algorithms for many problems:

**Fundamental algorithms**

- Multiplication, division with remainder
- Fast evaluation/interpolation, GCD, resultants
- Functional decomposition
- Sparsest shifts
Computer algebra understands (dense) polynomials

Computer algebra has changed mathematics and computer science in its computations with polynomials

\[ f = x^{88}y^{68} + 4x^{67}y^{49} + 4x^{46}y^{30} - 10x^{65}y^{53} - 40x^{44}y^{34} - 40x^{23}y^{15} + 25x^{42}y^{38} + 100x^{21}y^{19} + 100 \]

We have extremely effective algorithms for many problems:

Factoring

\[ f \rightarrow (x^{23}y^{15} - 5)^2 (x^{21}y^{19} + 2)^2 \]

Polynomial time:

- Lenstra, Lenstra, Lovasz (1982) over \( \mathbb{Q} \) (and LLL...)

Many, many subsequent improvements (esp. van Hoeij 2002)
But humans write mathematics more sparsely

And computer algebra systems are quite happy to comply:

\[ f = x^{26664}y^{60588} + 4x^{20301}y^{43659} + 4x^{13938}y^{26730} - 10x^{19695}y^{47223} - 40x^{13332}y^{30294} - 40x^{6969}y^{13365} + 25x^{12726}y^{33858} + 100x^{6363}y^{16929} + 100 \]
But humans write mathematics more sparsely

And computer algebra systems are quite happy to comply:

\[
f = x^{26664}y^{60588} + 4x^{20301}y^{43659} + 4x^{13938}y^{26730} - 10x^{19695}y^{47223} - 40x^{13332}y^{30294} \\
- 40x^{6969}y^{13365} + 25x^{12726}y^{33858} + 100x^{6363}y^{16929} + 100
\]

Computationally

A representation of polynomials as coefficient-exponent pairs

\[
f = c_1x^{e_1} + c_2x^{e_2} + \cdots + c_t x^{e_t} \in \mathbb{F}[x], \quad e_1 < \cdots < e_t, \quad c_i \in \mathbb{F}^*
\]

\[
\{(c_1, e_1), (c_2, e_2), \ldots, (c_t, e_t)\}
\]

Size is

\[
\sum_{i=1}^{t} \log |c_i| + \log |e_i| = O(t(\log \deg n + \log \|f\|))
\]
But humans write mathematics more sparsely

And computer algebra systems are quite happy to comply:

\[ f = x^{26664}y^{60588} + 4x^{20301}y^{43659} + 4x^{13938}y^{26730} - 10x^{19695}y^{47223} - 40x^{13332}y^{30294} - 40x^{6969}y^{13365} + 25x^{12726}y^{33858} + 100x^{6363}y^{16929} + 100 \]

\[ \text{factor}(f) \Rightarrow (x^{6363}y^{16929} + 2)^2 (x^{6969}y^{13365} - 5)^2 \]

But not on this laptop...

### Complexity

Fast algorithms run in time polynomial in the size

\[ (t + \log \deg f + \log \|f\|)^{O(1)} \]

machine operations.
We must ask the right questions

Don’t ask to factor

\[ x^{10000000} - 1 \]

Carette & Davenport (2009) take a more sophisticated approach
We must ask the right questions

Kaltofen (May 1, 2009 @ 1:14pm): Sparse interpolation of

\[
\frac{x^{100000} - 1}{x^{50000} - 1}
\]
We must ask the right questions

Kaltofen (May 1, 2009 @ 1:14pm): Sparse interpolation of

\[
x^{100000} - 1
\]

\[
\frac{x^{50000} - 1}{x^{50000} - 1}
\]

We should think about representation

The sparse power basis may simply be the wrong output format

- Straight-line programs (Kaltofen 1989 - ...)
- Black boxes (Kaltofen & Trager 1990)
- Other polynomial bases (Giesbrecht, Labahn, Lee 2004)
- Symbolic exponents (Watt 2007)

**Power bases are a natural and capture structural information.**
Some fundamental problems are intractable

Plaisted (1977, 1984) essentially showed that 3-SAT can be reduced to deciding if

\[ \gcd(\sum_{i=1}^{t} c_i x^{e_i}, x^N - 1) = 1 \]

Idea is to map truth assignments to roots of unity.

A catalogue of intractable problems was developed

- (bivariate) irreducibility
- Roots on unit circle
- Squarefreeness (Shparlinski, 1999)
- ...
Fast algorithms do exist

There has been much, often surprising, progress:
- Sparse interpolation (exact and approximate)
- Integer root finding
- Small degree factors (univariate and multivariate)
- Sparse shift interpolation
- Perfect power detection
- Polynomial decomposition
Fast algorithms do exist

There has been much, often surprising, progress:
- Sparse interpolation (exact and approximate)
- Integer root finding
- Small degree factors (univariate and multivariate)
- Sparse shift interpolation
- Perfect power detection
- Polynomial decomposition

And there are many enticing problems to work on
- Sparse division
- Cyclotomic-free algorithms
- Approximate/floating point algorithms
- Implementation
- ...
Cucker, Koiran, and Smale (1999)

Show how to find all integer roots of

$$f = \sum_i c_i x^{e_i} \in \mathbb{Z}[x]$$  \hspace{1cm} (*)

with cost \((t + \log \deg f + \log \|f\|)^{O(1)} ( !)\)
Cucker, Koiran, and Smale (1999)

Show how to find all integer roots of

\[ f = \sum_i c_i x^{e_i} \in \mathbb{Z}[x] \] \hspace{1cm} (*)

with cost \((t + \log \deg f + \log \| f \|)^{O(1)} \)

Lenstra (1999)

Find all irreducible factors of (*) of degree \( \leq r \) with cost

\((r + t + \log \deg f + \log \| f \|)^{O(1)} \)
### From integer root finding to small factors

**Cucker, Koiran, and Smale (1999)**

Show how to find all integer roots of

\[
    f = \sum_i c_i x^{e_i} \in \mathbb{Z}[x] \quad (\ast)
\]

with cost \((t + \log \deg f + \log \|f\|)^{O(1)}\) (!)

**Lenstra (1999)**

Find all irreducible factors of \((\ast)\) of degree \(\leq r\) with cost

\((r + t + \log \deg f + \log \|f\|)^{O(1)}\) (!!)

**Kaltofen & Koiran (2005,2006)**

\(f \in \mathbb{Q}[x_1, \ldots, x_m]\) \(t\)-sparse, and \(r\)

find all irreducible factors of degree \(\leq r\) with cost

\((r + t + \log \deg f + \log \|f\|)^{O(1)}\) (!!!)
If we know (or suspect) a polynomial is sparse, but can only evaluate it, we can interpolate it.

Given a “black box” for $f$

$$\alpha \rightarrow \blackbox \rightarrow f(\alpha)$$

Can find

$$f = \sum_{i=1}^{t} c_i x^{e_i}$$

with $2t$ evaluations of the black box, an $O(t^2)$ other operations (glossing over many theoretical and practical issues)

Need to go back to the ’90s for a solution
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. For a function $F : \mathbb{R} \to \mathbb{R}$, and $t \in \mathbb{Z}_{>0}$, find $F_i$, $a_i$ such that

$$F(x) = \sum_{1 \leq j \leq t} F_j e^{\delta_j x}$$
Sparse Interpolation

Ben-Or & Tiwari (1988)

Prony’s algorithm in a different setting. Given the ability to evaluate \( f(\omega^i) \) quickly, for some \( \omega \) and \( i = 1 \ldots 2t \), can reconstruct a \( t \)-sparse \( f \) as

\[
f = c_1 x^{e_1} + c_2 x^{e_2} + \cdots + c_t x^{e_t}
\]

Many subsequent improvements

- Kaltofen, Lakshman and Wiley (1990) – interpolation over \( \mathbb{Q} \)
- Grigorieve, Karpinski and Singer (1991) – rational functions
- Kaltofen & Lee (2003) - early termination
- Garg & Schost (2008) - CRT on exponents
Approximate sparse interpolation


Unknown “approximate” sparse polynomial

\[ f(x) = \sum_{i=1}^{t} c_i x^{e_i} \in \mathbb{C}[x] \]

Recovery from approximate evaluations at

\[ f(\omega^0), f(\omega^1), f(\omega^2), \ldots, f(\omega^{2t-1}) \]

for a “random” primitive root of unity \( \omega \in \mathbb{C} \).

- Provably robust algorithm to recover \( e_1, \ldots, e_t \) and \( c_1, \ldots, c_t \)
- Numeric stability with high probability (Monte Carlo)
- Reduce to showing a class of Vandermonde matrices are well-conditioned with high probability.
- Experimental results are even better!
Perfect powers

Back to (exact) sparse integer polynomials

Shparlinski (2000) looked at detecting if $f(x)$ is a perfect square

$$f(x) = x^{82470} y^{6288} + 6x^{54676} y^{4366} + 2x^{41235} y^{3144} + 9x^{26882} y^{2444} + 6x^{13441} y^{1222} + 1$$

$$= (x^{41235} y^{3144} + 3 x^{13441} y^{1222} + 1)^2$$

Proposed to look at more general powers:

$$f(x) = x^{123705} y^{9432} + 9x^{95911} y^{7510} + 3x^{82470} y^{6288} + 27x^{68117} y^{5588} + 18x^{54676} y^{4366}$$

$$+ 3x^{41235} y^{3144} + 27x^{40323} y^{3666} + 27x^{26882} y^{2444} + 9x^{13441} y^{1222} + 1$$

$$= (x^{41235} y^{3144} + 3 x^{13441} y^{1222} + 1)^3$$
Perfect powers

Sparsity-sensitive approaches

- Kaltofen (1987) - Straight-line programs; poly-time in $\deg h$
- Shparlinski (2000) - perfect square detection
### Perfect powers

#### Sparsity-sensitive approaches
- Kaltofen (1987) - Straight-line programs; poly-time in $\deg h$
- Shparlinski (2000) - perfect square detection

#### Giesbrecht & Roche (2008)
- Detect if $f = h^r$ and find $r$; Monte Carlo probabilistic
- Find $h$ if it exists; subject to conjecture of Schinzel and Erdös
- Detection: $\tilde{O}(t \log^2 \|f\| \log^2 n)$
- Root finding: $\tilde{O}(t(t + m)^4) \log \|f\| \log n)$, $m = \text{sparsity of } f$
Perfect power detection algorithm (for given $r$)

**Input:** $f \in \mathbb{Z}[x]$ and prime $r \in \mathbb{N}$

**Output:** Does there exists an $h \in \mathbb{Z}[x]$ that $f = h^r$?

1. Choose random prime $p \approx 2^n (\log n + \log ||f||)$
2. Find an irreducible $\Gamma \in \mathbb{F}_p[z]$ of degree $r - 1$; Let $\mathbb{F}_q \cong \mathbb{F}_p[z]/(\Gamma)$
3. For 5 random $\alpha \in \mathbb{F}_q$ do
4. $\xi := f(\alpha)^{(q-1)/r} \in \mathbb{F}_q$
5. If $\xi \neq 1$ return false
6. Return true

**Theorem**

If $f = h^r$ then this is always reported correctly.
If $f \neq h^r$ then this is reported with probability $> 1/2$. 
Perfect power detection algorithm (for given $r$)

**Input:** $f \in \mathbb{Z}[x]$ and prime $r \in \mathbb{N}$

**Output:** Does there exist an $h \in \mathbb{Z}[x]$ that $f = h^r$?

1. Choose random prime $p \approx 2n(\log n + \log \|f\|)$
2. Find an irreducible $\Gamma \in \mathbb{F}_p[z]$ of degree $r - 1$; Let $\mathbb{F}_q \cong \mathbb{F}_p[z]/(\Gamma)$
3. For 5 random $\alpha \in \mathbb{F}_q$ do
4. $\xi := f(\alpha)^{(q-1)/r} \in \mathbb{F}_q$
5. If $\xi \neq 1$ return false
6. Return true

**Proof**

Character sum argument + Weil’s (1948) theorem on character sums with polynomial arguments.
Computing perfect roots of lacunary polynomials

The perfect root problem

Given $f$ and $r$, such that we know there exists an $h$ with $f = h^r$.

How do we compute $h$?

Problem: is $h$ sparse?

Let $\tau(h)$ be the number of non-zero coefficients in $h$.

- Erdős (1947) conjecture if $f = h^2$, then $\tau(h) < \tau(f)$.
- Zannier (2008) shows that $\tau(h)$ a (computable) function of $\tau(f)$.

Avoid the problem: Output sensitive square root

$$f = c_1 x^{e_1} + \cdots + c_t x^{e_t} = h^2 = \left(b_1 x^{d_1} + \cdots + b_m x^{e_m}\right)^2 \in \mathbb{Z}[x]$$

Giesbrecht & Roche (2008): $(t + m + \log \deg f + \|f\|)O(1)$.  

Mark Giesbrecht

Some dense thoughts on sparse polynomials

25/41
Algorithm 1: Map to the unit circle

ω ∈ $\overline{\mathbb{Q}}$ a $p$th friendly primitive root of unity:

$h(\omega) = \sum_{1 \leq i \leq m} b_i \omega^{d_i \text{rem } p} \in \mathbb{Q}(\omega) \cong \mathbb{Q}(z)/(\Phi_p[z])$

Factor

$Y^2 - f(\omega) = (Y - h(\omega)) (Y + h(\omega)) \in \mathbb{Q}(\omega)[Y]$

Recover $h$ from $h(\omega)$
**Algorithm 1: Map to the unit circle**

\( \omega \in \overline{\mathbb{Q}} \) a \( p \)th friendly primitive root of unity:

\[
h(\omega) = \sum_{1 \leq i \leq m} b_i \omega^{d_i \text{rem} p} \in \mathbb{Q}(\omega) \cong \mathbb{Q}(z)/(\Phi_p[z])
\]

Factor

\[
Y^2 - f(\omega) = (Y - h(\omega))(Y + h(\omega)) \in \mathbb{Q}(\omega)[Y]
\]

Recover \( h \) from \( h(\omega) \)

**Algorithm 2: Sparse Newton Iteration**

- Compute \( h = f^{1/r} \in F[[x]] \) using Newton-like iteration
- Carefully preserve sparsity at each step.
- Can prove less, but can potentially do much more.
Shifted interpolation: a simple example

Suppose we can evaluate a polynomial at any chosen point:

\[ f(x) = (x - 3)^{1073} - 485(x - 3)^{524} \]

\[ \alpha \mapsto f(\alpha) \]

\( f \) is sparse in an unknown shifted basis.

How can we interpolate \( f(x) \)?
Suppose we can evaluate a polynomial at any chosen point:

\[ f(x) = (x - 3)^{1073} - 485(x - 3)^{524} \]

\[ \alpha \mapsto f(\alpha) \]

\( f \) is sparse in an unknown shifted basis.

How can we interpolate \( f(x) \)?

More generally, suppose

\[ f(x) = \sum_{i=1}^{t} c_i(x - \alpha)^{e_i} \]

Find this representation of \( f \) by interpolation.

Cost should be \((t + \log ||f|| + \log |\alpha|)^{O(1)}\).
Suppose we can evaluate a polynomial at any chosen point:

\[ f(x) = (x - 3)^{1073} - 485(x - 3)^{524} \]

\[ \alpha \mapsto f(\alpha) \]

\( f \) is sparse in an unknown shifted basis.

How can we interpolate \( f(x) \)?

---

**The “Modular Black-Box”**

\[ p \in \mathbb{N}, \ \theta \in \mathbb{Z}_p \quad \rightarrow \quad f(\theta) \mod p \]

\[ f(x) \in \mathbb{Q}[x] \]

Provides (necessary) control of evaluation size.
Computing the Sparsest Shift (of a dense polynomial)

- Borodin & Tiwari (1991)
  Compute sparsest shift from evaluation points (open)

- Grigoriev & Karpinski (1993)
  Compute sparsest shift from a black-box function.

- Lakshman & Saunders (1996)
  Compute sparsest shift from dense representation

  **Theorem:** If the degree is at least twice the sparsity, then the sparsest shift is unique and rational.

- Giesbrecht, Kaltofen & Lee (2003)
  Current best results in dense representation (deterministic & probabilistic)
Sparse shifts of sparse polynomials

**Input**: A bound $B$ on the bit length of the lacunary-shifted representation

1. Choose a prime $p$ with $p \in O(B^{O(1)})$
2. Evaluate $f(1), \ldots, f(p-1) \mod p$
   to attempt to interpolate $f_p(x)$
3. Use a dense sparsest shift method to compute $\alpha_p$
4. Repeat Steps 1–3 enough times to recover $\alpha$
5. Recover $f$ from $f_{p_1}, f_{p_2}, \ldots$ using Garg & Schost (2008)

The hard part is proving it works ...
Perfect power detection

The NTL implementation shows this is the “right” algorithm for this problem, even in the fairly dense case.
Adaptive Polynomial Representation

- Algorithms rely on one of sparse or dense representation.
- Combine the best of both worlds: Sparse polynomials with dense polynomial coefficients (the “chunks”)
- Can use existing sparse and dense algorithms simultaneously
- Choosing where to make the chunks is the tricky part.

Example

A polynomial: \( f = 5x^6 + 6x^7 - 4x^9 - 7x^{52} + 4x^{53} + 3x^{76} + x^{78} \)

\( f_1 = 5 + 6x - 4x^3, \quad f_2 = -7 + 4x, \quad f_3 = 3 + x^2 \)

\( f = f_1x^6 + f_2x^{52} + f_3x^{76} \)
Goal: Adaptive methods should be often faster, but never too much slower, than their non-adaptive counterparts.

Timing degree-10,000 multiplications with varying sparsity:
(Red line is NTL; others represent variations on “chunky” strategy)
Conclusion

- Sparse representations matter (in theory and practice)
- Some important problems are hard (but we can’t give up)
- Some important problems are surprisingly tractable.
Conclusion

- Sparse representations matter (in theory and practice)
- Some important problems are hard (but we can’t give up)
- Some important problems are surprisingly tractable.

Mark’s sparse agenda

- New algorithms need to be developed
Conclusion

- Sparse representations matter (in theory and practice)
- Some important problems are hard (but we can’t give up)
- Some important problems are surprisingly tractable.

Mark’s sparse agenda

- New algorithms need to be developed
- Hard problems need to be avoided
Conclusion
- Sparse representations matter (in theory and practice)
- Some important problems are hard (but we can’t give up)
- Some important problems are surprisingly tractable.

Mark’s sparse agenda
- New algorithms need to be developed
- Hard problems need to be **identified and avoided**
Conclusion

- Sparse representations matter (in theory and practice)
- Some important problems are hard (but we can’t give up)
- Some important problems are surprisingly tractable.

Mark’s sparse agenda

- New algorithms need to be developed
- Hard problems need to be identified and avoided
- Implementations should appreciate and exploit sparsity
Conclusion

- Sparse representations matter (in theory and practice)
- Some important problems are hard (but we can’t give up)
- Some important problems are surprisingly tractable.

Mark’s sparse agenda

- New algorithms need to be developed
- Hard problems need to be identified and avoided
- Implementations should appreciate and exploit sparsity