On Factorization of Sparse Polynomials of Bounded Individual Degree

Aminadav Chuyoon Amir Shpilka

School of Computer Science and AI
Tel-Aviv University

August 4, 2026

Sparse Polynomials

Definition (Sparse Polynomial)

An n-variate polynomial of degree poly(n) is said to be s-sparse if it is the sum of at most s monomials. We will sometimes denote the sparsity of a polynomial f by $||f||$.

- Though structurally simple, sparse polynomials have been extensively studied, both from structural and algorithmic perspectives.
- The motivating subject for the present work is the structure of their *factors*.

Example

Example (von zur Gathen–Kaltofen '85)

Consider

$$g(\mathbf{x}) = \prod_{i=1}^{n}(x_i - 1) \quad \text{and} \quad h(\mathbf{x}) = \prod_{i=1}^{n}(1 + x_i + \cdots + x_i^{n-1}) + n.$$

And

$$f(\mathbf{x}) = g \cdot h = \prod_{i=1}^{n}(x_i^n - 1) + n\prod_{i=1}^{n}(x_i - 1).$$

Then, $||f|| = (2^{n+1} - 1)$, and it has an irreducible factor h with $||h|| = n^n$.

This example shows that factors of a sparse polynomial may have quasi-polynomial sparsity.

Sparsity of Factors

Question 1

Is it always true that the sparsity of factors of a polynomial f is quasi-polynomial in the sparsity of f?

Sparsity of Factors

Question 1

Is it always true that the sparsity of factors of a polynomial f is quasi-polynomial in the sparsity of f?

Some special cases are known:

Sparsity of Factors

Question 1

Is it always true that the sparsity of factors of a polynomial f is quasi-polynomial in the sparsity of f?

Some special cases are known:

- $f|g, ||g|| \leq s$, f multilinear $\implies ||f|| \leq s$ (Volkovich '15).

Sparsity of Factors

Question 1

Is it always true that the sparsity of factors of a polynomial f is quasi-polynomial in the sparsity of f?

Some special cases are known:

- $f|g, ||g|| \leq s$, f multilinear $\implies ||f|| \leq s$ (Volkovich '15).
- $f|g, ||g|| \leq s$, g multiquadratic $\implies ||f|| \leq s$ (Volkovich '17).

Sparsity of Factors

Question 1

Is it always true that the sparsity of factors of a polynomial f is quasi-polynomial in the sparsity of f?

Some special cases are known:

- $f|g, ||g|| \leq s$, f multilinear $\implies ||f|| \leq s$ (Volkovich '15).
- $f|g, ||g|| \leq s$, g multiquadratic $\implies ||f|| \leq s$ (Volkovich '17).
- $f|g, ||g|| \leq s$, g is n-variate with bounded individual degree d $\implies ||f|| \leq s^{d^2 \log n}$ (Bhargava–Saraf–Volkovich '20).

Factorization of Sparse Polynomials

Question 2 (Dutta–Sinhababu–Thierauf '24)

Can we effectively factorize polynomials whose irreducible factors are all sparse?

Factorization of Sparse Polynomials

Question 2 (Dutta–Sinhababu–Thierauf '24)

Can we effectively factorize polynomials whose irreducible factors are all sparse?

Similarly to Question 1, only special cases are known:

Factorization of Sparse Polynomials

Question 2 (Dutta–Sinhababu–Thierauf '24)

Can we effectively factorize polynomials whose irreducible factors are all sparse?

Similarly to Question 1, only special cases are known:

- Sparse polynomials whose factors are all multilinear can be factored in polynomial time (Volkovich '15).

Factorization of Sparse Polynomials

Question 2 (Dutta–Sinhababu–Thierauf '24)

Can we effectively factorize polynomials whose irreducible factors are all sparse?

Similarly to Question 1, only special cases are known:

- Sparse polynomials whose factors are all multilinear can be factored in polynomial time (Volkovich '15).
- Sparse multiquadratic polynomials can be factored in polynomial time (Volkovich '17).

Factorization of Sparse Polynomials

Question 2 (Dutta–Sinhababu–Thierauf '24)

Can we effectively factorize polynomials whose irreducible factors are all sparse?

Similarly to Question 1, only special cases are known:

- Sparse polynomials whose factors are all multilinear can be factored in polynomial time (Volkovich '15).
- Sparse multiquadratic polynomials can be factored in polynomial time (Volkovich '17).
- An n-variate, s-sparse polynomial with bounded individual degree d can be factored in time $s^{O(d^7 \log n)}$ (Bhargava–Saraf–Volkovich '20).

Our Results I

- Notation: $\mathcal{P}(n, s, d)$ is the class of n-variate, s-sparse polynomials of bounded individual degree d.
- Elements of $\mathcal{P}(n, s, d)$ are called (n, s, d)-sparse polynomials.

Our Results I

- Notation: $\mathcal{P}(n, s, d)$ is the class of n-variate, s-sparse polynomials of bounded individual degree d.
- Elements of $\mathcal{P}(n, s, d)$ are called (n, s, d)-sparse polynomials.

Theorem

Let $f \in \mathbb{F}[\mathbf{x}]$ be an (n, s, d)-sparse polynomial not divisible by a monomial. Then:

- *f has at most $d \log s$ irreducible factors, thus at most s^d divisors overall.*
- *There is a $\mathrm{poly}(n, s^d)$ time algorithm that finds the s-sparse divisors of f.*

Our Results II

Theorem

There is a $\mathrm{poly}(n, s^{d \log \ell})$ *time algorithm that, given blackbox access to a product of* ℓ *irreducible polynomials in* $\mathcal{P}(n, s, d)$*, recovers these factors.*

Our Results II

Theorem

There is a $\mathrm{poly}(n, s^{d \log \ell})$ *time algorithm that, given blackbox access to a product of* ℓ *irreducible polynomials in* $\mathcal{P}(n, s, d)$*, recovers these factors.*

Theorem

An element of $\mathcal{P}(n, s, d)$ *can be factored in* $s^{O(d^2 \log n)}$ *time.*

Our Results II

Theorem

There is a $\mathrm{poly}(n, s^{d \log \ell})$ *time algorithm that, given blackbox access to a product of* ℓ *irreducible polynomials in* $\mathcal{P}(n, s, d)$*, recovers these factors.*

Theorem

An element of $\mathcal{P}(n, s, d)$ *can be factored in* $s^{O(d^2 \log n)}$ *time.*

- Note: the results above are valid when $\mathrm{char}(\mathbb{F}) \in \{0\} \cup [2d, \infty)$.
- It's possible to (essentially) remove this assumption.

High-Level Algorithmic Plan

Roughly, all our algorithms follow the following general paradigm.

High-Level Algorithmic Plan

Roughly, all our algorithms follow the following general paradigm.

- Reduce the polynomial to a constant number of variables.

High-Level Algorithmic Plan

Roughly, all our algorithms follow the following general paradigm.

- Reduce the polynomial to a constant number of variables.
- Factorize the resulting polynomial.

High-Level Algorithmic Plan

Roughly, all our algorithms follow the following general paradigm.

- Reduce the polynomial to a constant number of variables.
- Factorize the resulting polynomial.
- Interpolate to extract desired factors from the factorization.

High-Level Algorithmic Plan

Roughly, all our algorithms follow the following general paradigm.

- Reduce the polynomial to a constant number of variables.
- Factorize the resulting polynomial.
- Interpolate to extract desired factors from the factorization.

The change of variables will be based on a combination of the Klivans–Spielman (KS) and the Shpilka–Volkovich (SV) generators (Klivans–Spielman '01, Shpilka–Volkovich '09).

Generator Construction

Theorem (Klivans–Spielman '01)

Fix n, s, d, and let $m = (2nsd)^2$. There is a polynomial map $G^{KS}_{(m)} : \mathbb{F}^4 \to \mathbb{F}^n$, of degree $\mathrm{poly}(m)$*, so that an (n, s, d)-sparse polynomial f can be reconstructed from $f(G^{KS}_{(m)})$.*

Generator Construction

Theorem (Klivans–Spielman '01)

Fix n, s, d, and let $m = (2nsd)^2$. There is a polynomial map $G^{KS}_{(m)} : \mathbb{F}^4 \to \mathbb{F}^n$, of degree $\mathrm{poly}(m)$*, so that an (n, s, d)-sparse polynomial f can be reconstructed from $f(G^{KS}_{(m)})$.*

Definition (Shpilka–Volkovich '09)

$G^{\mathrm{SV}}(u, v) = (C_1(v)u, \ldots, C_n(v)u)$ for a set of Lagrange interpolating polynomials $C_1, \ldots, C_n$ for a set of points $\gamma_1, \ldots, \gamma_n$.

Generator Construction

Theorem (Klivans–Spielman '01)

Fix n, s, d, and let $m = (2nsd)^2$. There is a polynomial map $G^{KS}_{(m)} : \mathbb{F}^4 \to \mathbb{F}^n$, of degree poly$(m)$*, so that an (n, s, d)-sparse polynomial f can be reconstructed from $f(G^{KS}_{(m)})$.*

Definition (Shpilka–Volkovich '09)

$G^{\mathrm{SV}}(u, v) = (C_1(v)u, \ldots, C_n(v)u)$ for a set of Lagrange interpolating polynomials $C_1, \ldots, C_n$ for a set of points $\gamma_1, \ldots, \gamma_n$.

Definition (Our Generator)

Fix a parameter m, define
$G_{(m)}(x, y, z, w, u, v) = G^{\mathrm{KS}}_{(m)}(x, y, z, w) + G^{\mathrm{SV}}(u, v)$.

Generator Key Property

Definition (Reviving a variable)

For $1 \leq i \leq n$, let $G_{(m,i)}$ be the generator obtained from $G_{(m)}$ by restricting to $u = u - G^{\mathrm{KS}}_{(m)}(x, y, z, w)$ and $v = \gamma_i$.

Generator Key Property

Definition (Reviving a variable)

For $1 \leq i \leq n$, let $G_{(m,i)}$ be the generator obtained from $G_{(m)}$ by restricting to $u = u - G^{\mathrm{KS}}_{(m)}(x, y, z, w)$ and $v = \gamma_i$.

Our generator preserves the following information on the factors.

Lemma (Generator Key Property)

Suppose ϕ, ψ are two coprime factors of (n, s, d)-polynomials. Then, $\phi(G_{(m,i)}), \psi(G_{(m,i)})$ are coprime as polynomials in u, for some explicit constant $m = \mathrm{poly}(n, s^d, d!)$.

Generator Key Property

Definition (Reviving a variable)

For $1 \leq i \leq n$, let $G_{(m,i)}$ be the generator obtained from $G_{(m)}$ by restricting to $u = u - G^{\mathrm{KS}}_{(m)}(x, y, z, w)$ and $v = \gamma_i$.

Our generator preserves the following information on the factors.

Lemma (Generator Key Property)

Suppose ϕ, ψ are two coprime factors of (n, s, d)-polynomials. Then, $\phi(G_{(m,i)}), \psi(G_{(m,i)})$ are coprime as polynomials in u, for some explicit constant $m = \mathrm{poly}(n, s^d, d!)$.

The proof is by considering the rank of the Sylvester matrix $M := M_{x_i}(fg, (fg)')$, that counts the number of distinct factors of fg, and showing its preserved when composing with $G_{(m,i)}$.

Our Results III: Divisibility Testing and Rational Interpolation

Theorem (General Divisibility Testing)

There exists a deterministic $\mathrm{poly}(n, s^d, d!, \ell)$ *time algorithm that given blackbox access to two products* f, g *of factors of* (n, s, d)*-sparse polynomials, returns if* f *divides* g*.*

Our Results III: Divisibility Testing and Rational Interpolation

Theorem (General Divisibility Testing)

There exists a deterministic $\mathrm{poly}(n, s^d, d!, \ell)$ *time algorithm that given blackbox access to two products* f, g *of factors of* (n, s, d)*-sparse polynomials, returns if* f *divides* g*.*

Theorem (Rational Interpolation Theorem, simplified)

There is a $\mathrm{poly}(n, s^d, d!)$*-time algorithm that, given a blackbox access to a quotient of coprime* (n, s, d)*-sparse polynomials* $\frac{a}{b}$*, and a set of potential irreducible factors of* b*, reconstructs* a *and* b*.*

Meta-Algorithm

Our algorithms all rely on the following paradigm, which we call a "meta-algorithm".
To understand it well, please keep in mind the following two problems:

1. Finding all the sparse divisors of a sparse polynomial.
2. Recovering all the multiquadratic, *irreducible*, s-sparse factors of a product of ℓ factors of sparse polynomials.

Step 1 - Preprocessing

1. Write $f = \sum_{0 \leq i \leq d} f_i x_0^i$, where $f_i \in \mathbb{F}[\mathbf{x}]$. For simplicity, assume $f_0 \neq 0$.

Step 1 - Preprocessing

1. Write $f = \sum_{0 \leq i \leq d} f_i x_0^i$, where $f_i \in \mathbb{F}[\mathbf{x}]$. For simplicity, assume $f_0 \neq 0$.
2. Consider the *normalization* of f, given by $\tilde{f}(y, x_1, \ldots, x_n) = f(yf_0, x_1, \ldots, x_n)/f_0$. $\tilde{f}$ is *reversed monic*: $\tilde{f}(0) = 1$.

Step 1 - Preprocessing

1. Write $f = \sum_{0 \le i \le d} f_i x_0^i$, where $f_i \in \mathbb{F}[\mathbf{x}]$. For simplicity, assume $f_0 \neq 0$.
2. Consider the *normalization* of f, given by $\tilde{f}(y, x_1, \ldots, x_n) = f(yf_0, x_1, \ldots, x_n)/f_0$. $\tilde{f}$ is *reversed monic*: $\tilde{f}(0) = 1$.
3. We have that $\sum_i c_i x_0^i \mid f$ corresponds to $\sum_i \frac{c_i}{c_0} f_0^i y^i \mid \tilde{f}$.

Step 1 - Preprocessing

1. Write $f = \sum_{0 \le i \le d} f_i x_0^i$, where $f_i \in \mathbb{F}[\mathbf{x}]$. For simplicity, assume $f_0 \neq 0$.
2. Consider the *normalization* of f, given by $\tilde{f}(y, x_1, \ldots, x_n) = f(yf_0, x_1, \ldots, x_n)/f_0$. $\tilde{f}$ is *reversed monic*: $\tilde{f}(0) = 1$.
3. We have that $\sum_i c_i x_0^i \mid f$ corresponds to $\sum_i \frac{c_i}{c_0} f_0^i y^i \mid \tilde{f}$.
4. Use interpolation and factorization of polynomials in a fixed number of variables to write down $\tilde{f}(y, G)$ and its factorization.

Step 1 - Preprocessing

1. Write $f = \sum_{0 \le i \le d} f_i x_0^i$, where $f_i \in \mathbb{F}[\mathbf{x}]$. For simplicity, assume $f_0 \neq 0$.
2. Consider the *normalization* of f, given by $\tilde{f}(y, x_1, \ldots, x_n) = f(yf_0, x_1, \ldots, x_n)/f_0$. $\tilde{f}$ is *reversed monic*: $\tilde{f}(0) = 1$.
3. We have that $\sum_i c_i x_0^i \mid f$ corresponds to $\sum_i \frac{c_i}{c_0} f_0^i y^i \mid \tilde{f}$.
4. Use interpolation and factorization of polynomials in a fixed number of variables to write down $\tilde{f}(y, G)$ and its factorization.

Note:

- The normalization process ensures that after composing with the generator, the resulting polynomial will not have any content (that is, all non-constant polynomials dividing it will depend on y).
- The cost is that it forces us to work with *rational functions*.

Step 2 - Recursion

By applying one recursive call, run the algorithm on the polynomial f_0.

Step 2 - Recursion

By applying one recursive call, run the algorithm on the polynomial f_0.

- For the problem of finding all sparse divisors, this returns all the sparse polynomials dividing f_0.

Step 2 - Recursion

By applying one recursive call, run the algorithm on the polynomial f_0.

- For the problem of finding all sparse divisors, this returns all the sparse polynomials dividing f_0.
- For the problem of finding multiquadratic factors, this returns all the multiquadratic, irreducible, s-sparse polynomials dividing f_0.

Step 2 - Recursion

By applying one recursive call, run the algorithm on the polynomial f_0.

- For the problem of finding all sparse divisors, this returns all the sparse polynomials dividing f_0.
- For the problem of finding multiquadratic factors, this returns all the multiquadratic, irreducible, s-sparse polynomials dividing f_0.

Note that f_0 has the same properties as f, so this is fine.

Step 3 - Generating a List of Candidates

In this step, we generate a list of candidates for the desired factors of f.

Step 3 - Generating a List of Candidates

In this step, we generate a list of candidates for the desired factors of f.
As we already know:
$\sum_i c_i x_0^i \mid f$ corresponds to $\sum_i \frac{c_i}{c_0} f_0^i y^i \mid \tilde{f}$, and thus to $\sum_i \frac{c_i(G)}{c_0(G)} f_0^i y^i \mid \tilde{f}(y, G)$.

Step 3 - Generating a List of Candidates

In this step, we generate a list of candidates for the desired factors of f.
As we already know:
$\sum_i c_i x_0^i \mid f$ corresponds to $\sum_i \frac{c_i}{c_0} f_0^i y^i \mid \tilde{f}$, and thus to $\sum_i \frac{c_i(G)}{c_0(G)} f_0^i y^i \mid \tilde{f}(y, G)$.
To recover the desired factors:

- Enumerate over all the relevant factors of $\tilde{f}(y, G)$.
- For each such factor, reconstruct the potential c_is that produced it.

Step 3 - Generating a List of Candidates - Continued

We now have to reconstruct c_i from the information about $c_i(G)/c_0(G)$:

Step 3 - Generating a List of Candidates - Continued

We now have to reconstruct c_i from the information about $c_i(G)/c_0(G)$:

- In the case of finding all sparse divisors, c_0 is a sparse divisor of f_0, so we know all the options for it from the recursive call!

Step 3 - Generating a List of Candidates - Continued

We now have to reconstruct c_i from the information about $c_i(G)/c_0(G)$:

- In the case of finding all sparse divisors, c_0 is a sparse divisor of f_0, so we know all the options for it from the recursive call!
- In the case of finding multiquadratic factors, c_0 is a multiquadratic factor of f_0, thus all its irreducible factors are found in the recursive call - and the rational interpolation result applies.

Step 3 - Generating a List of Candidates - Continued

We now have to reconstruct c_i from the information about $c_i(G)/c_0(G)$:

- In the case of finding all sparse divisors, c_0 is a sparse divisor of f_0, so we know all the options for it from the recursive call!
- In the case of finding multiquadratic factors, c_0 is a multiquadratic factor of f_0, thus all its irreducible factors are found in the recursive call - and the rational interpolation result applies.

We note that for this step to work, we crucially needed some *closure* property to hold.

Step 4 - Pruning the List of Candidates

To rule out fake candidates:

- For the first problem, as all the candidates are sparse, just use divisibility testing.

Step 4 - Pruning the List of Candidates

To rule out fake candidates:

- For the first problem, as all the candidates are sparse, just use divisibility testing.
- For the second problem, further use divisibility testing to understand which of the factors is minimal with respect to divisibility. These minimal factors are the irreducible factors.

Number of Irreducible Factors

Lemma

If f is (n, s, d)-sparse not divisible by a monomial, it has at most $d \log s$ irreducible factors.

Number of Irreducible Factors

Lemma

If f is (n, s, d)-sparse not divisible by a monomial, it has at most $d \log s$ irreducible factors.

- Show that there is a set of at most $\log s$ variables, so that each irreducible factor depends on one of them. Denote by k the smallest number with this property.

Number of Irreducible Factors

Lemma

If f is (n, s, d)-sparse not divisible by a monomial, it has at most $d \log s$ irreducible factors.

- Show that there is a set of at most $\log s$ variables, so that each irreducible factor depends on one of them. Denote by k the smallest number with this property.
- Note that by the bound on the individual degree, f has at most kd irreducible factors.

Number of Irreducible Factors

Lemma

If f is (n, s, d)-sparse not divisible by a monomial, it has at most $d \log s$ irreducible factors.

- Show that there is a set of at most $\log s$ variables, so that each irreducible factor depends on one of them. Denote by k the smallest number with this property.
- Note that by the bound on the individual degree, f has at most kd irreducible factors.
- Analyze the *Newton Polytope* of f, and show that it must have at least 2^k vertices, hence $k \leq \log s$.

Follow-up Work

More work has been published on the topic since the paper's publication:

- In a follow-up work (Chuyoon–Shpilka '26+), we prove a general rational interpolation result and use it to get a $\mathrm{poly}(n, (s\ell d)^d)$-time algorithm for factoring a product of ℓ (n, s, d)-sparse polynomials.
- Very recently, (Bhattacharjee–Kothary–Rai–Saraf '26) were able to design an efficient algorithm that output a set of circuits computing factors of sparse polynomials.

Thank You!