Deterministic Algorithms for Low Individual Degree Factors of Sparse Polynomials

Joint Work with Somnath Bhattacharjee [U of T], Shanthanu S. Rai [TIFR], Shubhangi Saraf [U of T]

Key Question

Question: Given $f \in \mathbb{F}[x_1, \ldots, x_n]$ can we efficiently return the list of factors of f?

Key Question

Question: Given $f \in \mathbb{F}[x_1, \ldots, x_n]$ can we efficiently return the list of all factors?

Points to Ponder:

1. How are we representing our polynomials?
2. What does efficient mean?

Polynomial Representation

- Represent f by its monomials and coefficients. [Also called Sparse Representation]

Polynomial Representation

- Represent f by its monomials and coefficients. [Also called Sparse Representation]

- Example: $f: 10x^{10}y^2 + 17x^3y^5 - 21$

  $\rightarrow \{(x^{10}y^2, 10), (x^3y^5, 17), (1, -21)\}$

Polynomial Representation

- Represent f by its monomials and coefficients. [Also called Sparse Representation]

- Example: $f : 10x^{10}y^2 + 17x^3y^5 - 21$
  $\rightarrow \{(x^{10}y^2, 10), (x^3y^5, 17), (1, -21)\}$

- We call $f \in \mathbb{F}[x_1, \ldots, x_n]$ a s-sparse polynomial if f has atmost s monomials with non zero coefficients. Let $\|f\|$:= Number of monomials with non zero coefficients.

Polynomial Representation

- Represent f by its monomials and coefficients. [Also called Sparse Representation]

- Example: $f : 10x^{10}y^2 + 17x^3y^5 - 21$ $(\|f\| = 3)$

  $\rightarrow \{(x^{10}y^2, 10), (x^3y^5, 17), (1, -21)\}$

- We call $f \in \mathbb{F}[x_1, \ldots, x_n]$ a s-sparse polynomial if f has atmost s monomials with non zero coefficients. Let $\|f\|$:= Number of monomials with non zero coefficients.

Key Question Updated

- **Question:** Given $f \in \mathbb{F}[x_1, \ldots, x_n]$ an s-sparse polynomial of degree d, can we output the list of all factors of f in time $\text{poly}(s, n, d)$?

Key Question Updated

- **Question:** Given $f \in \mathbb{F}[x_1, \ldots, x_n]$ an s-sparse polynomial of degree d, can we output the list of all factors of f in time poly(s, n, d) ?
  - ↳ notion of efficiency
- Are factors of f poly(s, n, d)-sparse ?

Key Question Updated

- **Question:** Given $f \in \mathbb{F}[x_1, \ldots, x_n]$ an s-sparse polynomial of degree d, can we output the list of all factors of f in time poly(s, n, d) ?
  - ↳ notion of efficiency
- Are factors of f poly(s, n, d)-sparse ? Sadly No !

Sparsity Blowup of Factors

- Let $f(x_1, \ldots, x_n) = \prod_{i=1}^{n} (x_i^d - 1)$

  then $\|f\| = 2^n$ and $\deg(f) = d \cdot n$.

Sparsity Blowup of Factors

- Let $f(x_1, \ldots, x_n) = \prod_{i=1}^{n} (x_i^d - 1)$

  then $\|f\| = 2^n$ and $\deg(f) = d \cdot n$.

- Consider $g(x_1, \ldots, x_n) = \prod_{i=1}^{n} (1 + x_i + \cdots + x_i^{d-1})$, then $g \mid f$

  and $\|g\| = d^n = \|f\|^{\log(d)}$ [GK'85]

- The above example works over fields of any characteristic.

Sparsity Blowup of Factors

- Let $f(x_1, \ldots, x_n) = x_1^p + \cdots + x_n^p$ where $f \in \mathbb{F}_p[x_1, \ldots, x_n]$

  then $\|f\| = n$ and $\deg(f) = p$.

Sparsity Blowup of Factors

- Let $f(x_1, \ldots, x_n) = x_1^p + \cdots + x_n^p$ where $f \in \mathbb{F}_p[x_1, \ldots, x_n]$

  then $\|f\| = n$ and $\deg(f) = p$.

- Let $g(x_1, \ldots, x_n) = (x_1 + \cdots + x_n)^{p-1}$ then $g \mid f$ and

  $\|g\| = \binom{n+p-1}{p} = \Theta(\|f\|^p)$ [BSV'20]

Relaxing our Goals

- The sparsity blowup won't allow us a poly(n, s, d) algorithm.

Relaxing our Goals

- The sparsity blowup won't allow us a poly(n, s, d) algorithm.

- In this work we consider two relaxations of the problem:

  Relaxation 1: f be an s-sparse polynomial with bounded individual degree d.

  - f has individual degree at most d if $\forall i \in [n]$ $\deg_{x_i}(f) \leq d$.

  - Think $d : O(1)$ for this relaxation.

Relaxing our Goals

- The sparsity blowup won't allow us a poly(n, s, d) algorithm.

- In this work we consider two relaxations of the problem:

Relaxation 1: f be an s-sparse polynomial with bounded individual degree d.

Relaxation 2: f be an s-sparse polynomial and output all factors of f which have bounded individual degree d.

Again think $d = O(1)$.

Bounded Ind. Deg Factorization

- We have $f \in \mathbb{F}[\bar{x}_n]$ s.t. f is s-sparse and has individual degree at most d.

- [BSV'20]: For any $g \mid f$, $\|g\| \leq s^{O(d^2 \log n)}$ [Factor Sparsity Bound]

Bounded Ind. Deg Factorization

- We have $f \in \mathbb{F}[\bar{x}_n]$ s.t. f is s-sparse and has individual degree at most d.

- [BSV'20]: For any $g \mid f$, $\|g\| \leq s^{O(d^2 \log n)}$ [Factor Sparsity Bound]

- [BSV'20]: $s^{O(d^7 \log n)}$ algorithm that computes factorization of f into irreducible polynomials.

- The algorithmic result is closely tied to factor sparsity bound.

Bounded Ind. Deg Factorization

- We have $f \in \mathbb{F}[\bar{x}_n]$ s.t. f is s-sparse and has individual degree at most d.

- [BSV'20]: For any $g \mid f$, $\|g\| \leq s^{O(d^2 \log n)}$ [Factor Sparsity Bound]

- [BSV'20]: $s^{O(d^7 \log n)}$ algorithm that computes factorization of f into irreducible polynomials.

- The algorithmic result is closely tied to factor sparsity bound.

- [Vol'17] conjectured that $\|g\| \leq s^{\text{poly}(d)}$. If the conjecture is true above algorithm runs in time $s^{\text{poly}(d)}$

Bounded Ind. Deg Factorization

- Recently Chuyoon and Shpilka made further progress on the algorithmic aspects.

- [CS'26] proved:

① $\exists$ poly($s^{d^2 \log n}$, n) algorithm which outputs the factorization of f into its irreducibles.

Bounded Ind. Deg Factorization

- Recently Chuyoon and Shpilka made further progress on the algorithmic aspects.

- [CS'26] proved:

① $\exists$ poly($s^{d^2 \log n}$, n) algorithm which outputs the factorization of f into its irreducibles.

② $\exists$ poly(s^d, d!, n) algorithms which outputs all s-sparse factors* of f.

* → factors not divisible by a monomial.

How to make Further Progress

- for small d, can we show $\exists$ $poly(s^d, n)$ algorithm to output all factor of f ?

How to make Further Progress

- for small d, can we show $\exists$ $poly(s^{d^{O(1)}}, n)$ algorithm to output all factors of f?

- One major hurdle is how to represent factors?

- If polynomial sparsity conjecture were true then we could represent using sparse representation. That is a big if :(

How to make Further Progress

- for small d, can we show $\exists$ poly(s, n) algorithm to output all factors of f?

- One major hurdle is how to represent factors?

- If polynomial sparsity conjecture were true then we could represent using sparse representation. That is a big if :(

- In this work, we output factors as constant depth algebraic circuits.

Algebraic Circuits

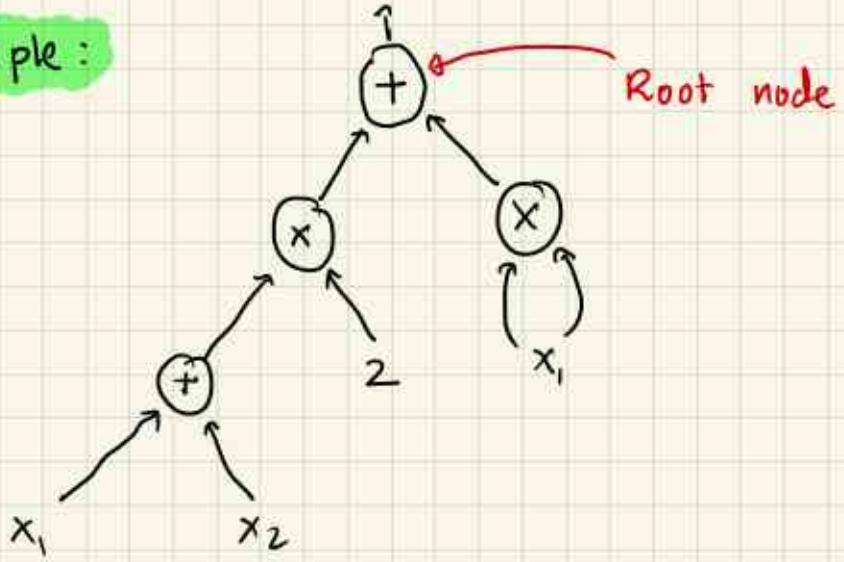
- Internal Gates : ⊗ → Multiplication Gates
  ⊕ → Addition Gates

- Leaf Nodes : Variables, Constants.

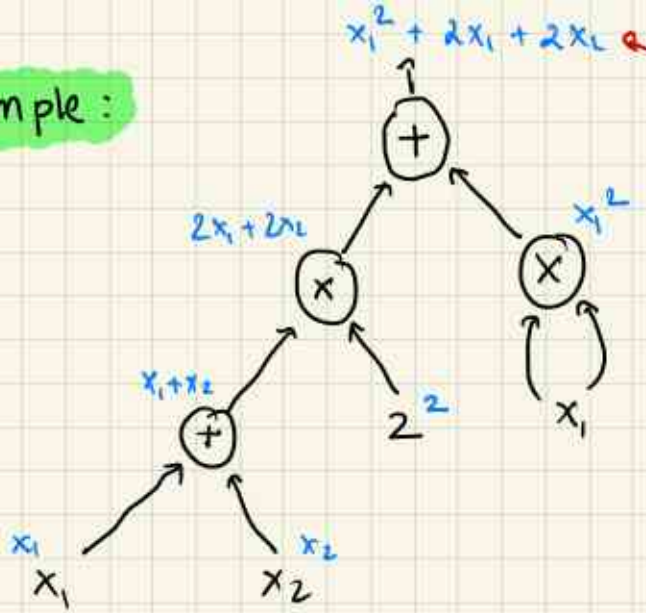
- An algebraic circuit is a directed and rooted acyclic graph with internal gates and leaf nodes as defined above.

- No restriction of fan-in / fan-out

- Example :



- **Example:**

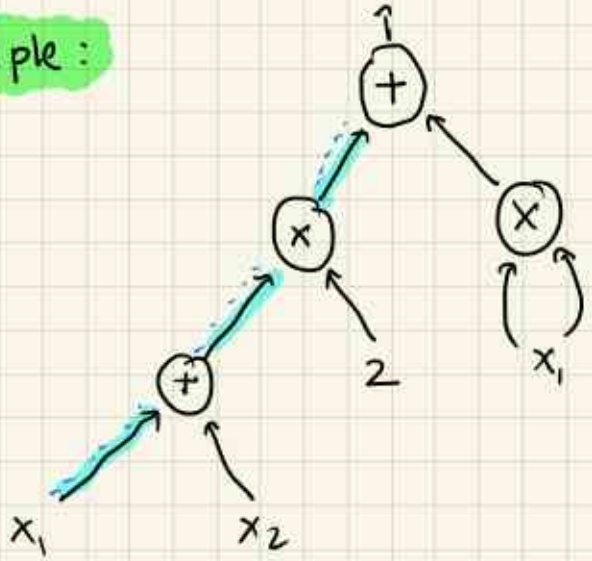

$x_1^2 + 2x_1 + 2x_2$
Polynomial computed by the circuit.
+
$2x_1 + 2x_2$
x
x_1^2
x
$x_1 + x_2$
+
2
x_1
x_1
x_2
x_2

- Some important measures for algebraic circuits:

  * Size(C) = # leaf nodes + internal gates in the circuit.

  * Depth(C) = Length of longest path from root to a leaf node.

- Example :



Size = 8

Depth = 3

Factors as Circuit

- Foundational works by Kaltofen [Kal'89], Kaltofen and Trager [KT'90] already gave a poly time randomized algorithm when we allow circuits! This works for general sparse polynomials

Factors as Circuit

- Foundational works by Kaltofen [Kal'89], Kaltofen and Trager [KT'90] already gave a poly time randomized algorithm when we allow circuits! This works for general sparse polynomials

- For general sparse polynomials the best known deterministic algorithms run in subexponential by the works of Bhattacharjee et al [BKRSS'25][BKRRSS'26]

Factors as Circuit

- Foundational works by Kaltofen [Kal'89], Kaltofen and Trager [KT'90] already gave a poly time randomized algorithm when we allow circuits! This works for general sparse polynomials

- For general sparse polynomials the best known deterministic algorithms run in subexponential by the works of Bhattacharjee et al [BKRSS'25][BKRRSS'26]

- We believe efficient derandomization exists since this question is equivalent to PIT [KSS'15] (for general ckts)

Our Results

① **Theorem 1:** Let $f \in \mathbb{F}[\bar{x}_n]$ where $\mathrm{char}(\mathbb{F}) = 0$ or $\mathrm{char}(\mathbb{F}) > d$, be an s-sparse polynomial of individual degree at most d. There is a $\mathrm{poly}(s^d, d!, n)$ time algorithm that outputs a list that contains all factors of f as constant depth ckts.** *

Our Results

① **Theorem 1:** Let $f \in \mathbb{F}[\bar{x}_n]$, where $\mathrm{char}(\mathbb{F}) = 0$ or $\mathrm{char}(\mathbb{F}) > d$, be an s-sparse polynomial of individual degree at most d. There is a $\mathrm{poly}(s^d, d!, n)$ time algorithm that outputs a list that contains all factors of f as constant depth ckts.**

* Factors that are not divisible by any monomial.

* The list is of size at most s^d and may contain spurious entries.

② **Theorem 2:** Let $f \in \mathbb{F}[\bar{x}_n]$ be an s-sparse polynomial of individual degree at most D (unbounded).

Then $\exists$ $\text{poly}(D^{d \log s}, s^{d^2 \log n})$ time algorithm that outputs a list containing all factors of f which have bounded individual degree at most d (bounded) as constant depth circuits.

② **Theorem 2:** Let $f \in \mathbb{F}[\bar{x}_n]$ be an s-sparse polynomial of individual degree at most D (unbounded).

Then $\exists$ $\text{poly}(D^{d \log s}, s^{d^2 \log n})$ time algorithm that outputs a list containing all factors of f which have bounded individual degree at most d (bounded) as constant depth circuits.**

* Factors that are not divisible by a monomial
* The list is of size at most $\binom{D}{\leq d}^{\log s}$ and may contain spurious entries.

$$\binom{D}{\leq d} : \sum_{i=0}^{d} \binom{D}{i}$$

Technical Ideas

Plan: ① Recap [BSV'20] algorithm for factoring

② Recap new ideas used by [CS'26]

③ Highlight what we did different.

[BSV'20] Algorithm

- **Input:** $f \to$ s-sparse polynomial of bounded individual degree $\leq d$
- **Step 0:** Prove structural lemma about sparsity of factors of f.

[BSV'20] Algorithm

- **Input:** $f \to$ s-sparse polynomial of bounded individual degree $\leq d$
- **Step 0:** Prove structural lemma about sparsity of factors of f.
- **Step 1:** Normalize the Polynomial
  - Let $f(\bar{X}, Y) = \sum_{i=0}^{t} f_i(\bar{X}) Y^i$ where $f_t \not\equiv 0$

    $\hat{f}(\bar{X}, Y) = f(\bar{X}, Y/f_t) \cdot f_0^{t-1}$

    ↑ Normalization of f
  - $\|\hat{f}\| \leq s^{O(d)}$ and $\hat{f}$ has ind. deg at most $2d$.

[BSV'20] Algorithm

- **Input:** $f \to$ s-sparse polynomial of bounded individual degree $\leq d$
- **Step 0:** Prove structural lemma about sparsity of factors of f.
- **Step 1:** Normalize the Polynomial
- **Step 2:** Factorize the normalized polynomial $\hat{f}$ into irreducibles

[BSV'20] Monic Factorization

- Let $\hat{f}(\vec{X}, Y) = h_1(\vec{X}, Y)^{e_1} \cdot \cdots \cdot h_r(\vec{X}, Y)^{e_r}$ be its irreducible factorization.

- We project $\hat{f}$ to a low dimensional space that maintains the factorization pattern. [Takes time $\text{poly}(s^{d^3 \log n})$]

Find $\alpha \in \mathbb{F}^n$ s.t. $\forall i \neq j$, $\gcd(h_i(\alpha, Y), h_j(\alpha, Y)) = 1$.

$\Downarrow$

$\text{Res}_Y(h_i, h_j)(\alpha) \neq 0$.

$\hat{f}(\alpha, Y)$ is the low dimensional polynomial.

[BSV'20] Monic Factorization

- Let $\hat{f}(\vec{x}, Y) = h_1(\vec{x}, Y)^{e_1} \cdots h_r(\vec{x}, Y)^{e_r}$ be its irreducible factorization.
- We project $\hat{f}$ to a low dimensional space that maintains the factorization pattern.
- Factorize the low dimensional polynomial.

  Just use brute force factorization for this part

[BSV'20] Monic Factorization

- Let $\hat{f}(\vec{x}, Y) = h_1(\vec{x}, Y)^{e_1} \cdot \cdots \cdot h_r(\vec{x}, Y)^{e_r}$ be its irreducible factorization.
- We project $\hat{f}$ to a low dimensional space that maintains the factorization pattern.
- Factorize the low dimensional polynomial.
- Reconstruct the irreducible factorization from low dimensional factors.

[BSV'20] Algorithm

- **Input:** $f \to$ s-sparse polynomial of bounded individual degree $\leq d$
- **Step 0:** Prove structural lemma about sparsity of factors of f.
- **Step 1:** Normalize the Polynomial
- **Step 2:** Factorize the normalized polynomial $\hat{f}$ into irreducibles
- **Step 3:** Recursively factor $f_t(\bar{x})$

[Remember $f = \sum_{i=0}^{t} f_i(\bar{x}) Y^i$]

[BSV'20] Algorithm

- **Input:** $f \to$ s-sparse polynomial of bounded individual degree $\leq d$
- **Step 0:** Prove structural lemma about sparsity of factors of f.
- **Step 1:** Normalize the Polynomial
- **Step 2:** Factorize the normalized polynomial $\hat{f}$ into irreducibles
- **Step 3:** Recursively factor $f_t(\bar{x})$
- **Step 4:** Using factorization of $\hat{f}$ and f_t, find irreducible factorization of f. [This reduces to a sparse PIT]

[CS'26] Optimizations

- Uses the same Step 0 - Step 4 ideas as [BSV'20]. However [CS'26] makes some changes in their implementation.

[CS'26] Optimizations

- Step 0: Same as [BSV'20]

[CS'26] Optimizations

- **Step 0:** Same as [BSV'20]

- **Step 1:** Reverse normalize f.

$$f(\vec{x}, Y) : \sum_{i=0}^{t} f_i(\vec{x}) Y^i \quad \text{where} \quad f_0(\vec{x}) \neq 0$$

$$\tilde{f}(\vec{x}, Y) : f(\vec{x}, Y \cdot f_0) / f_0$$

↑ Reverce normalization of f

$\|\tilde{f}\| = s^{O(d)}$ and $\tilde{f}$ has individual deg $2d$.

[CS'26] Optimizations

- Step 0: Same as [BSV'20]
- Step 1: Reverse normalize f.
- Step 2: Find all sparse factors of $\tilde{f}$.

  The projection to a low dimensional space is implemented differently.

[CS'26] Optimizations

- **Step 0:** Same as [BSV'20]
- **Step 1:** Reverse normalize f.
- **Step 2:** Find all sparse factors of $\tilde{f}$.
- **Step 3:** Recursively factor f_0
- **Step 4:** Using factorization of $\hat{f}$ and f_0, recombine them appropriately to find all sparse factors of f.

Our Algorithm

- **Step 0:** Same as [BSV'20]

Our Algorithm

- **Step 0:** Same as [BSV'20]

- **Step 1:** Depending on whether $\|f_{\mathrm{even}}\|$ is small or $\|f_{\mathrm{odd}}\|$ is small normalize or reverse-normalize the polynomial.

CS's algorithm might need n layers of recursion but we are cutting it down to $\log(s)$.

Our Algorithm

- **Step 0:** Same as [BSV'20]

- **Step 1:** Depending on whether $\|f_{\le l}\|$ is small or $\|f_{>l}\|$ is small normalize or reverse-normalize the polynomial.

- **Step 2:** Factorize the normalized (or reverse-normalized) polynomial.

Here we again project down to a low dimensional space, but use Hensel lifting / Rustenberg to find all factors.

Earlier algorithms used sparse reconstruction.

Our Algorithm

- **Step 0:** Same as [BSV'20]
- **Step 1:** Depending on whether $\|f_e\|$ is small or $\|f_o\|$ is small normalize or reverse-normalize the polynomial.
- **Step 2:** Factorize the normalized (or reverse-normalized) polynomial.
- **Step 3:** Recursively factor f_e (or f_o)
- **Step 4:** Using factors of normalized (or reverse normalized) polynomial and f_e (or f_o) reconstruct all factors of f_o.

Final Remarks

- The unbounded individual degree case also has same algorithmic structure but one major problem we face is that normalization can blow up the sparsity. We use ideas from [CS'26] to deal with the case.

Final Remarks

- The general sparse polynomial case also has same algorithmic structure but one major problem we face is that normalization can blow up the sparsity. We use ideas from [CS'26] to deal with the case.
- Our algorithm recovers the bounded ind.-deg results from [CS'26] and [BSV'20].
- Our algorithm recovers the result of finding constant degree factors for general sparse polynomial from [KRS'24], [DST'24]. We prune our list using divicibility testing from [Forbes'15]

Open Problems

- Can we prune our list of factors efficiently? In particular for an s-sparse polynomial f and multilinear polynomial g, test whether $g \mid f$?

- Our algorithm does not work over low characteristic. Can we extend it to that setting?