



Derandomizing Multivariate Polynomial Factoring for Low Degree Factors

Pranjal Dutta   

School of Computing, National University of Singapore, Singapore

Amit Sinhababu 

Chennai Mathematical Institute, Chennai, India

Thomas Thierauf  

Ulm University, Germany

Abstract

Kaltofen [STOC 1986] gave a randomized algorithm to factor multivariate polynomials given by algebraic circuits. We derandomize the algorithm in some special cases.

For an n -variate polynomial f of degree d from a class \mathcal{C} of algebraic circuits, we design a deterministic algorithm to find all its irreducible factors of degree $\leq \delta$, for *constant* δ . The running time of this algorithm stems from a deterministic PIT algorithm for class \mathcal{C} and a deterministic algorithm that tests divisibility of f by a polynomial of degree $\leq \delta$.

By using the PIT algorithm for constant-depth circuits by Limaye, Srinivasan and Tavenas [FOCS 2021] and the divisibility results by Forbes [FOCS 2015], this generalizes and simplifies a recent result by Kumar, Ramanathan and Saptharishi [SODA 2024]. They designed a subexponential-time algorithm that, given a blackbox access to f computed by a constant-depth circuit, outputs its irreducible factors of degree $\leq \delta$. When the input f is sparse, the time complexity of our algorithm depends on a whitebox PIT algorithm for $\sum_i m_i g_i^{d_i}$, where m_i are monomials and $\deg(g_i) \leq \delta$. All the previous algorithms required a blackbox PIT algorithm for the same class.

Our second main result considers polynomials f , where *each* irreducible factor has degree at most δ . We show that all the irreducible factors with their multiplicities can be computed in polynomial time with blackbox access to f .

Finally, we consider factorization of *sparse* polynomials. We show that in order to compute all the sparse irreducible factors efficiently, it suffices to derandomize irreducibility preserving bivariate projections for sparse polynomials.

2012 ACM Subject Classification Theory of computation \rightarrow Algebraic complexity theory; Computing methodologies \rightarrow Algebraic algorithms

Keywords and phrases algebraic complexity, factoring, low degree, weight isolation, divisibility

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2024.75

Category RANDOM

Funding *Pranjal Dutta*: Funded by the project “Foundation of Lattice-based Cryptography”, by NUS-NCS Joint Laboratory for Cyber Security.

Thomas Thierauf: Supported by DFG grant TH 472/5-2.

1 Introduction

The problem of *multivariate polynomial factorization* asks to find the unique factorization of a given polynomial $f \in \mathbb{F}[x_1, \dots, x_n]$ as a product of distinct irreducible polynomials over \mathbb{F} . The problem reduces to *univariate polynomial factorization* over the same field, for which a deterministic polynomial time algorithm is known over the field \mathbb{Q} . The complexity of multivariate factorization depends on the representation of input and output polynomials. If we use *dense representation* (where all the coefficients are listed including the zero coefficients), deterministic polynomial time algorithms for multivariate factoring



© Pranjal Dutta, Amit Sinhababu, and Thomas Thierauf;
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 75; pp. 75:1–75:20



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

are known [16]. If we use *sparse representation* (where only the nonzero coefficients are listed), only randomized polynomial time (in the total sparsity of input polynomial and the output factors) algorithms are known [41, 21]. There are other standard representations like arithmetic circuits, and blackbox models (that gives the evaluations of the polynomial at any point, but the internal structure of the computation is hidden). Randomized polynomial time factorization algorithms are known in these models due to the classic results of Kaltofen [19] and Kaltofen and Trager [21]. Randomization is naturally required for these models, as the more basic question of *polynomial identity testing* (given a circuit/blackbox, test if it computes the zero polynomial) is not yet derandomized.

Towards derandomization of special cases of multivariate factoring, we are motivated by the following two questions.

► **Question 1.** Given a sparse polynomial f . Can we find all the sparse irreducible factors of f by a deterministic algorithm in polynomial/quasipolynomial/subexponential time?

Note that the factors of a sparse polynomial f might be nonsparse. Bhargava, Saraf and Volkovich [3] showed an upper bound on the sparsity of the factors of a sparse polynomial f . However, the bound is exponential in the degree of f . Therefore, instead of finding *all* the irreducible factors, we want to output only those factors that are sparse.

► **Question 2 ([41, 3]).** Given polynomial $f = \prod_{i=1}^m g_i^{e_i}$ as a blackbox, where polynomials g_i are irreducible polynomials whose sparsities are bounded by s . Can we find the polynomials g_i in *deterministic* time $\text{poly}(s, n, d)$ or time quasi-poly/sub-exponential in s, n, d ?

The second question can be seen as a special case of polynomial factorization, where we are *promised* that *all* the irreducible factors are sparse. To our surprise, we do not know a deterministic *subexponential*-time algorithm even for the special case of Question 2, when the given blackbox computes the product of just *two* irreducible sparse polynomials.

Derandomization of multivariate factoring (whitebox, or blackbox) reduces to (whitebox, or correspondingly blackbox) derandomization of polynomial identity testing (PIT). Kopparty, Saraf and Shpilka [27] showed this reduction in the model of arithmetic circuits. However, we do not know if sparse factorization reduces to sparse PIT or constant-depth arithmetic circuit PIT (the algorithms of [27] reduce to general arithmetic circuit PIT). Recently, there has been some progress on these questions by [28, 29]. Earlier works of Volkovich [39, 40] made progress on several special cases of sparse multivariate factoring.

Multivariate polynomial factoring has various applications, such as *low-degree testing* [1], constructions of pseudorandom generators for low-degree polynomials [6, 8], computational algebraic geometry [14] and many more. blackbox multivariate polynomial factorization is extensively used in *arithmetic circuit reconstruction* [36, 37], and polynomial equivalence testing [22, 23, 33]. Algebraic hardness vs randomness [15] results crucially use multivariate factorization. Special cases of depth-4 polynomial identity testing are related to questions about sparse polynomial factorization [12, 40, 4].

Divisibility testing. In a factorization algorithm, we may want to check if a candidate factor is truly a factor via *divisibility testing*. It asks to test if a polynomial $g(\mathbf{z})$ divides a polynomial $f(\mathbf{z})$. Forbes [9] showed that the divisibility testing question can be efficiently reduced to an instance of a PIT question of a model that relates to both f and g ; see Lemma 9. Currently, we do not know any deterministic polynomial time algorithm even when g and f are both sparse polynomials. When f is a sparse polynomial and g is a linear polynomial, the problem reduces to polynomial identity testing of any-order read-once oblivious branching programs (ROABPs), for which polynomial time whitebox PIT algorithm [34] and quasipolynomial

time blackbox PIT algorithms are known [10, 13, 11]. We do not know a deterministic polynomial time algorithm, even for testing if a quadratic polynomial g divides a sparse polynomial.

1.1 Our results

We show a general result that exhibits properties of a class \mathcal{C} of polynomials, such that we can compute the *constant-degree factors* of polynomials $f \in \mathcal{C}$. The following theorem is an informal statement of Theorem 18.

► **Theorem 1** (Low-degree factors via divisibility). *Let $\delta \in \mathbb{N}$ be a constant and \mathcal{C} be a class of polynomials such that there is an efficient PIT algorithm for \mathcal{C} . For any n -variate polynomial $f \in \mathcal{C}$ of degree d , finding all its irreducible factors of degree $\leq \delta$ reduces to solving polynomially many divisibility questions of whether a given polynomial of degree $\leq \delta$ divides f .*

Arguably, Theorem 1 generalizes and simplifies a recent result by Kumar, Ramanathan and Saptharishi [28] about the factorization of polynomials computed by constant-depth circuits. Importantly, if f is represented in the whitebox setting, then both the required algorithms (PIT and divisibility testing) in Theorem 1 are *whitebox* algorithms, whereas [28] still requires blackbox algorithms. We compare the results in more detail in Section 1.2.

We can apply Theorem 1 in the case of (any-order) ROABPs. There are polynomial time (respectively, quasipolynomial time) whitebox (respectively, blackbox) PIT algorithms for ROABPs. Moreover, using the divisibility techniques by Forbes [9] (see Lemma 9) and the duality trick by Saxena [35], the divisibility testing question of whether a given linear polynomial divides a ROABP can be reduced to a PIT instance of a polynomial-size ROABP.

► **Corollary 2** (Linear factors of ROABPs). *Let f be an n -variate polynomial of degree d , computed by an any-order ROABP of width w . Then one can output all its linear factors, along with the exponents in time $\text{poly}(ndw)$ in the whitebox setting, and in time $\text{poly}(ndw^{\log \log w})$ in the blackbox setting.*

When the input f is s -sparse, this result is already known due to Volkovich [39, Theorem 4]. Note that a sparse polynomial has a trivial ROABP.

Finally, we remark that Theorem 1 can be further generalized to outputting factors from a general class \mathcal{D} (as black box) when in addition to assuming (informally speaking) efficient PIT for \mathcal{C} and the divisibility test for \mathcal{C} by \mathcal{D} , one has to assume efficient derandomization of HIT for \mathcal{D} (in the sense of finding a good bivariate projection preserving irreducibility, see Assumption 1) and an inclusion property (i.e. given $g \in \mathcal{D}$ or not). For simplicity, in the conference version, we only assume that \mathcal{D} is the class of constant-degree polynomials.

Our second result considers the class of polynomials f , where all the irreducible factors of f are promised to have degrees bounded by δ . For this class, there are blackbox PIT algorithms with time complexity $\text{poly}(d, n^\delta)$ for n -variate polynomials f of degree d , see [7, 5]. Hence, by Theorem 1, the factoring problem reduces to a divisibility question. Using techniques of Forbes [9] (see Lemma 9), this can be further reduced to designing a blackbox PIT algorithm for polynomials of the form $\Sigma_i \Pi_j f_{i,j}$, where $\deg(f_{i,j}) \leq \delta$. However, we *do not* know better than subexponential-time PIT algorithms for this model. Thus, Theorem 1 does not yield anything fruitful in this promise setting. We show how to completely avoid divisibility testing and still find all the irreducible factors in polynomial time. The following theorem is an informal statement of Theorem 19.

► **Theorem 3** (Promise low-degree factoring). *Let $\delta \in \mathbb{N}$ be a constant. Given a blackbox access to an n -variate polynomial f of degree d such that all its irreducible factors have degrees at most δ , one can deterministically output all its irreducible factors along with the multiplicities in $\text{poly}(nd)$ time.*

The above theorem can be generalized to polynomials whose irreducible factors are from a class \mathcal{D} (and we will output them as blackbox), for which efficient PIT and derandomization of HIT (in the sense of finding a good bivariate projection preserving irreducibility; see Assumption 1) is known. For simplicity, in the conference version, we only focus on the class of constant-degree polynomials.

Related work on Theorem 3. There have been some works when $\delta = 1$. In this setting, given a promise that $f(\mathbf{z}) = \prod_{i \in [m]} \ell_i^{e_i}$, where $\ell_i(\mathbf{z})$ are mutually co-prime linear polynomials, we have to output ℓ_i . A randomized polynomial time algorithm for this problem follows from the work of Kaltofen and Trager [21]. Recently, Koiran and Ressayre [25] gave *three* different randomized algorithms for the *non-promise* problem that can test if a given f can be completely factored into linear polynomials and output the factorization if it exists. The first algorithm assumes that ℓ_1, \dots, ℓ_m are *linearly independent*, while the last two do not need that assumption. Later, Koiran and Skomra [26] derandomized the first algorithm when ℓ_i are linearly independent. Using a different idea and linear independence of ℓ_i , Medini and Shpilka [32] gave an alternative deterministic polynomial time algorithm. All these works exploited the linearity (and sometimes randomization/linear independence) of the factors, while our algorithm neither requires linearity of the factors nor any linear independence.

Finally, we go back to Question 1 of outputting all the sparse irreducible factors of a given sparse polynomial. Can efficient sparse irreducibility testing lead to an efficient algorithm? For general multivariate factoring, an effective version of Hilbert’s Irreducibility Theorem (HIT) by Kaltofen [17] (Theorem 10) says that with high probability, an irreducible n -variate polynomial remains irreducible if we randomly project it to a bivariate polynomial. This leads to an efficient factoring algorithm, since HIT helps to preserve the *factorization pattern* (the number of distinct irreducible factors and corresponding multiplicities). The hardness of Question 1 stems from the fact that a sparse polynomial may have both sparse and non-sparse irreducible factors. Hence, preserving irreducibility for sparse polynomials will not preserve the factorization pattern, and therefore, it may be hard to get back the actual factor. However, we observe that a deterministic version of HIT for sparse polynomials can indeed solve Question 1. The following theorem is an informal statement of Theorem 20.

► **Theorem 4** (Conditional sparse factoring, Informal). *Suppose there is an efficient algorithm that finds a bivariate projection, making an s -sparse irreducible polynomial both monic (in one variable) and irreducible. Then there is a subexponential-time algorithm that outputs all its irreducible factors with sparsities $\leq s$ along with their multiplicities.*

1.2 Comparison with Kumar, Ramanathan and Saptharishi [28]

Theorem 1 implies [28, Theorem 1.1–1.2]. Let $\Delta \geq 2$ be an arbitrary positive integer. Assume that we have a blackbox access to f , which can be computed by a Δ -depth algebraic circuit of size s . The recent breakthrough result of Limaye, Srinivasan, and Tavenas [31] gives a subexponential time identity testing algorithm for f . Moreover, using the techniques from [9] (see Lemma 9), one can show that whether a given polynomial of degree $\leq \delta$ divides f can be efficiently reduced to PIT for an algebraic circuit of size $\text{poly}(sd)$, of the

form $\sum_i g_i h_i^{d_i}$, where the polynomials g_i are computable by Δ -depth algebraic circuits and $\deg(h_i) \leq \delta$. For a formal proof, see [28, Corollary 2.19]. The main time complexity of [28, Theorem 1.1] is also dictated by the best-known blackbox PIT algorithm for the same model as above, which runs in subexponential time [31]. This algorithm requires the underlying field to have characteristic 0.

When $\Delta = 2$, Theorem 1 gives a quasipolynomial time algorithm to output irreducible factors of degree $\leq \delta$, thus implying [28, Theorem 1.2]. We know a polynomial time identity testing for sparse polynomials, due to Klivans and Spielman [24]. Further, [9, Corollary 7.16] showed that whether a polynomial of degree $\leq \delta$ divides a sparse polynomial, reduces to PIT for $\Sigma m \wedge \Sigma \Pi^{[\delta]}$; this model computes polynomials of the form $\sum_{i=1}^{\text{poly}(sd)} m_i h_i^{d_i}$, where m_i are monomials, and $\deg(h_i) \leq \delta$. The best-known blackbox (and whitebox) PIT algorithm for this model runs in quasipolynomial time [9, Corollary 6.7], and work over fields of characteristics 0 or large.

Whitebox vs. blackbox. Interestingly, if the input f has a whitebox access to it (for example when f is sparse, we can use [24]), then the required PIT algorithms in Theorem 1 are also in the whitebox setting. On the other hand, the factoring algorithm in [28] requires blackbox PIT algorithms. To explain it further, let $f(x, \mathbf{z})$ be a monic polynomial (in x) and computed by a constant-depth circuit. Further, $f = g \cdot h$, where $\deg(g) \leq \delta$ and $\gcd(g, h) = 1$. In the usual factoring algorithm via Hensel lifting/Newton iteration, it is important to find a good starting point $\mathbf{a} \in \mathbb{F}^n$ such that $\gcd(g(x, \mathbf{a}), h(x, \mathbf{a})) = 1$. This step is usually ensured by finding a hitting set for the *Resultant* polynomial $\text{Res}_x(g(x, \mathbf{z}), h(x, \mathbf{z}))$. Once such a point is found, one can project to the univariate $f(x, \mathbf{a})$, factorize it and then do the lifting. [28] observed that $\text{Res}(g, h) = \text{Res}(g, f/g)$. Further, they showed that the polynomials f/g as well as $\text{Res}(g, f/g)$ can be computed by small-size constant-depth algebraic circuits. This was enough to find a good projection using [31], and then find the true factor g via lifting. Since we do not know the factor g a priori, the polynomials f/g and $\text{Res}(g, f/g)$ can be viewed as polynomials computable by small-size constant-depth algebraic circuits *without* having explicit access to them.

1.3 Proof idea

In this section, we give an overview of our algorithms. The overall idea is to project the input polynomial to a trivariate polynomial, factorize it, and recover the original factors via efficient sparse interpolation [24].

Proof ideas of Theorem 1 and Theorem 3. Suppose $f(x, \mathbf{z})$ is an $(n + 1)$ -variate degree d homogeneous polynomial computed by an s -size circuit, which is *monic* in x . The monicness property can be assumed otherwise it is well-known that a random shift can make f monic, and this step can be derandomized assuming PIT for f ; see Lemma 7. We start with the simplest scenario of $\delta = 1$, i.e., given f , we want to output its linear factors.

Suppose $f = \ell^e \cdot g$, with $e \geq 1$, where $\gcd(\ell, g) = 1$ and ℓ is a linear polynomial. Consider the substitution $\phi : z_i \mapsto y^i$, where y is a new variable. Observe that $\phi(f) \in \mathbb{F}[x, y]$ is a nonzero monic polynomial (in x) of total degree at most nd . Further, $\phi(\ell)$, remains an *irreducible* factor of $\phi(f)$ and it is easy to identify ℓ from $\phi(\ell)$, since the monomials $\{z_1, \dots, z_n\}$ are assigned y^i *uniquely*. One can factorize the bivariate polynomial $\phi(f)$ in deterministic $\text{poly}(nd)$ time (see Lemma 11). We can apply the inverse of ϕ to each factor having degree 1 in x , and there could be nd many candidates of linear factors. The actual factor ℓ must be one of them. The divisibility testing makes sure that it always outputs the true linear factors.

When $\delta \geq 2$, we can still find small weights w_i , such that any monomial \mathbf{z}^e of degree $\leq \delta$ gets uniquely mapped to y^i , via $\phi : z_i \mapsto y^{w_i}$; see Lemma 14. Unfortunately, this map may not preserve irreducibility. On the other hand, an effective version of Hilbert's Irreducibility Theorem [20] shows that a monic irreducible polynomial $g(x, \mathbf{z})$ remains irreducible under the substitution $z_i \mapsto \beta_i t + \gamma_i$, where β_i, γ_i are *randomly* chosen from \mathbb{F} ; see Theorem 10. Further, this step can be derandomized when g is a low-degree polynomial. We combine these two ideas to get small weights w_i and w'_i such that the projection $\Psi : z_i \mapsto y^{w_i} t + y^{w'_i}$ preserves the irreducibility of *any* polynomial of degree $\leq \delta$, and further it is *uniquely recoverable* from the projected trivariate polynomial; see Corollary 16. Therefore, it suffices to factor the trivariate polynomial, find all its irreducible factors, and recover the original factors.

The proof of Theorem 3 uses the same trivariate projection as above. In this case, we can *avoid* divisibility because the trivariate projection preserves the factorization pattern, and one can recover the original factors uniquely from the projected ones.

Proof idea of Theorem 4. Kaltofen and Trager [21] gave an efficient blackbox factoring algorithm, that given a blackbox access to a polynomial f , and an arbitrary point, outputs evaluations at that point of all its irreducible factors. For simplicity, consider a monic polynomial $f(x, \mathbf{z})$. To get the evaluations at $(\alpha, \mathbf{c}) \in \mathbb{F}^{n+1}$, consider a trivariate projection $\eta : \mathbf{z} \mapsto \beta t_1 + (\mathbf{c} - \gamma)t_2 + \gamma$ and $x \mapsto x$, for new variables t_1 and t_2 . Here $\beta, \gamma \in \mathbb{F}^n$ was chosen such that $\mathbf{z} \mapsto \beta t + \gamma$ *preserves* the irreducibility *all* the irreducible factors of f ; such a projection exists using Theorem 10. The map η preserves the factorization pattern, and hence one can find the evaluations by factoring $\eta(f)$, and evaluating the irreducible factors at $x = \alpha, t_1 = 0, t_2 = 1$.

Our algorithm is a simple adaptation of their algorithm, with the following observation. Let $g(x, \mathbf{z})$ be an irreducible sparse factor of $f(x, \mathbf{z})$ and let $\beta, \gamma \in \mathbb{F}^n$ be such that $g(x, \beta t + \gamma)$ remains irreducible. Although the map η does not preserve the factorization pattern, $\eta(g)$ remains an irreducible factor of $\eta(f)$. Therefore, $g(\alpha, \mathbf{c})$ can be efficiently found, via evaluating the *right* trivariate factor. For finding the right factor, one can observe that there is a unique correspondence between the bivariate $g(x, \beta t + \gamma)$ and trivariate $\eta(g)$. Since, g is sparse, one can use sparse interpolation [24] to explicitly reconstruct the polynomial g , from its evaluations. Finally, whether a sparse polynomial g divides the input sparse polynomial f can be solved in deterministic subexponential time, via divisibility-to-PIT reduction of [9] (see Lemma 9) and the blackbox PIT algorithm for constant-depth circuits of [31].

2 Preliminaries

We take $\mathbb{F} = \mathbb{Q}$ as the underlying field throughout the paper, although the results hold over large characteristics.

Let $\mathcal{P}(n, d)$ be the set of n -variate polynomials of degree at most d , with variables $\mathbf{z} = (z_1, z_2, \dots, z_n)$. For an exponent vector $\mathbf{e} = (e_1, e_2, \dots, e_n)$, we denote the monomial $\mathbf{z}^{\mathbf{e}} = (z_1^{e_1}, z_2^{e_2}, \dots, z_n^{e_n})$. Its degree is $\|\mathbf{e}\|_1 = \sum_{i=1}^n e_i$.

For $\mathbf{a} \in \mathbb{F}^n$, we also denote $\|\mathbf{a}\|_0 = |\{i \mid a_i \neq 0\}|$.

$\text{sp}(f)$ denotes the *sparsity*, i.e., the number of monomials with nonzero coefficients in f .

$\text{Hom}_k[f]$ denotes the homogeneous component of f of degree equal to k .

A polynomial f is called *irreducible*, if it cannot be factored into the product of two non-constant polynomials. Polynomial f is called *square-free*, if for any non-constant factor g , the polynomial g^2 is not a factor of f .

By $\deg(f)$ we denote the total degree of f . Let x and $\mathbf{z} = (z_1, \dots, z_n)$ be variables and $f(x, \mathbf{z})$ be a $(n + 1)$ -variate polynomial. Then we can view f as a univariate polynomial $f = \sum_i a_i(\mathbf{z}) x^i$ over $\mathbb{K}[x]$, where $\mathbb{K} = \mathbb{F}[\mathbf{z}]$. The x -degree of f is denoted by $\deg_x(f)$. It is the highest degree of x in f . Polynomial f is called *monic in x* , if the coefficient $a_{d_x}(\mathbf{z})$ is the constant 1 polynomial, i.e. $a_{d_x}(\mathbf{z}) = 1$, where $d_x = \deg_x(f)$.

An algorithm runs in *subexponential time*, if its running time on inputs of length n is bounded by 2^{n^ϵ} , for any $\epsilon > 0$.

2.1 Computational problems, complexity measures and closure properties

For classes \mathcal{P}, \mathcal{Q} of multivariate polynomials, we define the following computational problems.

- PIT(\mathcal{P}): given $p \in \mathcal{P}$, decide whether $p \equiv 0$.
- Factor($\mathcal{P}|\mathcal{Q}$): given $p \in \mathcal{P}$, compute all its irreducible factors in \mathcal{Q} with their multiplicities.
- Div($\mathcal{P}|\mathcal{Q}$): given $p \in \mathcal{P}$ and $q \in \mathcal{Q}$, decide whether $q|p$.

The time complexity to solve these problems we denote by $T_{\text{PIT}(\mathcal{P})}$, $T_{\text{Factor}(\mathcal{P}|\mathcal{Q})}$, and $T_{\text{Div}(\mathcal{P}|\mathcal{Q})}$, respectively.

► **Remark.** Note that a decision algorithm for PIT(\mathcal{P}) also yields an algorithm that computes a point $\mathbf{a} \in (\mathbb{F} \setminus \{0\})^n$ such that $p(\mathbf{a}) \neq 0$, in case when $p \not\equiv 0$. In the blackbox case, the queries of the decision algorithm on the input of the zero-polynomial yield a *hitting set*¹ for the whole class \mathcal{P} . In the whitebox case, one can search for \mathbf{a} by assigning values successively to the variables and do kind of a *self-reduction*. For each variable, one tries at most d values from $\{1, 2, \dots, d\}$ for a polynomial of degree d . If they all give 0, definitely $d + 1$ works because it cannot be zero at $(d + 1)$ many values. With n variables, this amounts to nd calls to the PIT-decision algorithm. The final desired point $\mathbf{a} \in \{1, \dots, d + 1\}^n$, which is very explicit. The running time to compute \mathbf{a} is therefore bounded by $nd \cdot T_{\text{PIT}(\mathcal{P})}$.

For time complexity, we assume that the polynomials are given in some model of computation, such as circuits, branching programs, or formulas. With each model, we associate a complexity measure $\mu : \mathbb{F}[\mathbf{z}] \rightarrow \mathbb{N}$. For example, let $f \in \mathbb{F}[\mathbf{z}]$, some of the commonly used measures in the literature are:

- $\mu(f) = \text{sp}(f)$, the number of monomials with nonzero coefficients,
- $\mu(f) = \text{size}_\Delta(f)$, the size of the *smallest* depth- Δ algebraic circuit computing f ,
- $\mu(f) = \text{size}_{\text{ROABP}}(f)$, the *width* of the *smallest* any-order read read-once oblivious branching program (ROABP) computing f .

We define classes of polynomials of bounded measure,

$$\mathcal{C}_\mu(s, n, d) := \{f \in \mathcal{P}(n, d) \mid \mu(f) \leq s\}. \tag{1}$$

We generally assume that all polynomials we deal with can be *efficiently evaluated* at any point $\mathbf{a} \in \mathbb{F}^n$ within the respective measure, where we consider the unit-cost model for operations over \mathbb{F} . This holds for all the computational models usually considered in the literature.

► **Definition 5 (Closure under derivatives).** *Class $\mathcal{C}_\mu(s, n, d)$ is closed under derivatives, if for any $f \in \mathcal{C}_\mu(s, n, d)$, a variable $z \in \{z_1, \dots, z_n\}$, and $e \in \mathbb{N}$, the size of the derivative $\mu(\partial^e f / \partial z^e) = \text{poly}(snd)$, and further it can be computed in $\text{poly}(snd)$ time from f .*

¹ $H \subseteq \mathbb{F}^n$ is a hitting set for a class \mathcal{P} , if for every nonzero $f \in \mathcal{P}$, there exists $\mathbf{a} \in H$, such that $f(\mathbf{a}) \neq 0$.

► **Definition 6** (Closure under highest degree). Let $f = \sum_{k=0}^d f_k(\mathbf{z}) \in \mathcal{C}_\mu(s, n, d)$, where $f_k(\mathbf{z}) = \text{Hom}_k[f]$, the homogeneous component of f of degree k . We say that $\mathcal{C}_\mu(s, n, d)$ is closed under highest degree component if in the above, $\mu(f_d) \leq \text{poly}(snd)$, and further it can be computed in $\text{poly}(snd)$ time from f .

For example, if the class contains polynomials where each is a product of constant-degree polynomials, then it is *not* closed under homogenization (i.e. $\mu(f_k) \leq \text{poly}(snd)$ and it can be computed in polynomial time). However, the highest-degree component is still a product of constant-degree polynomials. Other classes that are closed under the highest degree are sparse polynomials, polynomials computed by polynomial size any-order ROABPs, or by constant-depth circuits.

Finally, we define $\text{Hom}[\mathcal{C}_\mu(s, n, d)]$, as follows:

$$\text{Hom}[\mathcal{C}_\mu(s, n, d)] := \{ f \in \mathcal{P}(n, d) \mid g \in \mathcal{C}_\mu(s, n, d) \text{ and } f = \text{Hom}_d[g] \}. \quad (2)$$

When the context and the parameters are clear, we will simply denote the classes as \mathcal{C} and $\text{Hom}[\mathcal{C}]$, without explicitly writing the parameters.

2.2 Transformation to a monic polynomial

Algorithms for factoring polynomials often assume that the given polynomial is monic. If this is not the case for the given polynomial f , we apply a transformation τ to f that yields a monic polynomial $\tau(f)$ that we can factor. From the factors of $\tau(f)$ we can then reveal the factors of f . Although this is standard in the literature, we state it in terms of the symbols that we introduced above.

► **Lemma 7** (Transformation to monic). Let \mathcal{C} be a class of polynomials that is closed under highest degree component. Let $f(\mathbf{z}) \in \mathcal{C}$ be n -variate of degree d and size s . For a new variable x , and $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n) \in (\mathbb{F} \setminus \{0\})^n$, define a linear transformation τ_α on the variables z_i :

$$\tau_\alpha : z_i \mapsto \alpha_i x + z_i,$$

where $\alpha_i \neq 0$, for $i = 1, 2, \dots, n$. Let $f_\alpha(x, \mathbf{z})$ be the resulting polynomial.

Then we can compute $\boldsymbol{\alpha}$ such that $\frac{1}{f_d(\boldsymbol{\alpha})} f_\alpha(x, \mathbf{z})$ is monic in x in time $nd \cdot \text{T}_{\text{PIT}}(\text{Hom}[\mathcal{C}]) + \text{poly}(snd)$, where $f_d = \text{Hom}_d[f]$.

Proof. Let $f(\mathbf{z}) \in \mathcal{C}$ be a polynomial of degree d with n variables $\mathbf{z} = (z_1, \dots, z_n)$.

To see what the transformation does, let

$$f = f_0 + f_1 + \dots + f_d,$$

where $f_k = \text{Hom}_k[f]$, the homogeneous degree- k component of f . Consider the degree- d component,

$$f_d(\mathbf{z}) = \sum_{|\boldsymbol{\beta}|_1=d} c_\beta \mathbf{z}^\beta.$$

Then, for f_α , we have $\deg_x(f_\alpha) = d$ and the coefficient of the leading x -term x^d in f_α is $f_d(\boldsymbol{\alpha}) = \sum_{|\boldsymbol{\beta}|_1=d} c_\beta \boldsymbol{\alpha}^\beta$.

Hence, the PIT algorithm for the homogeneous component f_d of f yields an $\boldsymbol{\alpha} \in (\mathbb{F} \setminus \{0\})^n$ such that $f_d(\boldsymbol{\alpha}) \neq 0$. Then the polynomial $\frac{1}{f_d(\boldsymbol{\alpha})} f_\alpha(x, \mathbf{z})$ is monic in x . For simplicity of notation, assume in the following that $f_d(\boldsymbol{\alpha}) = 1$, so that $f_\alpha(x, \mathbf{z})$ is monic in x . ◀

Since we work with the shifted polynomial, we need to ensure that the shift of variables does not affect the irreducibility of the factors; this is guaranteed by the following lemma. This is quite standard in the literature; for a nice proof, see [29, Lemma B7].

► **Lemma 8.** *Let $f(\mathbf{z}) \in \mathbb{F}[\mathbf{z}]$ be an n -variate irreducible polynomial. Then, for every $\mathbf{a} \in \mathbb{F}^n$, the polynomial $f(\mathbf{a}\mathbf{x} + \mathbf{z})$ is also irreducible.*

2.3 Divisibility testing reduces to PIT

Strassen [38] showed that if $g \mid f$, where both f and g can be computed by size s circuits, then $h := f/g$ can also be computed by a circuit of size $\text{poly}(sd)$, where $d = \deg(h)$. Forbes [9] observed that this procedure can still be done, even when $g \nmid f$, and we will get a small size circuit computing a polynomial \tilde{h} . We can then argue that g divides f if and only if $f = g \cdot \tilde{h}$. The latter question is a PIT question.

► **Lemma 9** (Divisibility reduces to PIT, [9, Corollary 7.10]). *Let $g(\mathbf{z})$ and $f(\mathbf{z})$ be two polynomials of degree at most d . Let $S \subseteq \mathbb{F}$ be a $\text{poly}(d)$ -explicit set such that $|S| = 2d^2 + 1$. Further let $\alpha \in \mathbb{F}^n$ such that $g(\alpha) \neq 0$. Then there are $\text{poly}(d)$ -explicit constants $\{c_{\beta,i}\}_{\beta \in S, 0 \leq i \leq d}$, such that*

$$g(\mathbf{z}) \mid f(\mathbf{z}) \iff f(\mathbf{z} + \alpha) - g(\mathbf{z} + \alpha) \cdot \sum_{\beta \in S} f(\beta\mathbf{z} + \alpha) \sum_{0 \leq i \leq d} c_{\beta,i} \cdot g(\beta\mathbf{z} + \alpha)^i = 0$$

2.4 Effective Hilbert’s Irreducibility Theorem

An effective version of Hilbert’s Irreducibility Theorem due to Kaltofen and von zur Gathen shows how to project a multivariate irreducible polynomial down to two variables, such that the projected bivariate polynomial stays irreducible. The proof shows the existence of an *irreducibility certifying polynomial* $G(\mathbf{a}, \mathbf{b})$ in $2n$ variables corresponding to the irreducible polynomial $g(x, \mathbf{z})$. The nonzeroness of G proves the irreducibility of $g(x, \mathbf{z})$ and also gives a way to find an irreducibility-preserving projection to bivariate (see [17, 20, 27]).

► **Theorem 10.** *Let $g(x, \mathbf{z})$ be an irreducible polynomial of total degree δ with $n + 1$ variables that is monic in x . There exists a nonzero polynomial $G(\mathbf{a}, \mathbf{b})$ of degree $2\delta^5$ in $2n$ variables such that for $\alpha, \beta \in \mathbb{F}^n$,*

$$G(\alpha, \beta) \neq 0 \implies \hat{g}(x, t) = g(x, \alpha_1 t + \beta_1, \dots, \alpha_n t + \beta_n) \text{ is irreducible.}$$

The certifying polynomial G immediately yields a randomized algorithm to construct the irreducible projection \hat{g} via PIT. The derandomization of Hilbert’s Irreducibility Theorem is a challenging open problem in general. We observe that it can be derandomized for constant degree polynomials.

2.5 Basics of factoring and interpolation

Berlekamp [2] and Lenstra, Lenstra and Lovász [30] gave efficient factorization algorithms for *univariate* polynomials over finite fields and \mathbb{Q} , respectively. Kaltofen [18] showed how to reduce the factorization of *bivariate* polynomials to univariate polynomials. In fact, the reduction works for k -variate polynomials, for any constant k . In our case, we use it for the case $k = 3$.

Via standard interpolation, one can assume that the input is given as a dense representation.

75:10 Derandomizing Multivariate Polynomial Factoring for Low Degree Factors

► **Lemma 11** (Trivariate Factorization). *Let $f(x, y, z)$ be a trivariate polynomial of degree d . Then there exists an algorithm that outputs all its irreducible factors and their multiplicities in time $\text{poly}(d)$.*

The following lemma shows how to find the multiplicity of an irreducible factor g of a polynomial f . It holds when $\text{char}(\mathbb{F}) = 0$, or, large. For a concise proof, see [28, Lemma 4.1].

► **Lemma 12** (Factor multiplicity). *Let $f(\mathbf{z}), g(\mathbf{z}) \in \mathbb{F}[\mathbf{z}]$ be non-zero polynomials and let $z \in \{z_1, \dots, z_n\}$ be such that $\partial_z(g) \neq 0$ and g is irreducible. Then the multiplicity of g in f is the smallest non-negative integer e such that $g \nmid \frac{\partial^e f}{\partial z^e}$.*

Klivans and Spielman [24] derandomized the isolation lemma for PIT of sparse polynomials. Their algorithm works over fields of 0 or large characteristics.

► **Lemma 13** (Sparse PIT and interpolation). *Given an n -variate s -sparse polynomial f of degree d via blackbox access, PIT for f works in time $\text{poly}(snd)$. Furthermore, if f is nonzero, one can find the monomials in f with nonzero coefficients in time $\text{poly}(snd)$.*

► **Remark.** Let $\mathbb{F} = \mathbb{Q}$, for the simplicity. The interpolation algorithm in [24] works by projecting f to a univariate polynomial in y via the map $z_i \mapsto p_i y^{w_i}$, for p_i are distinct primes, and weights w_i . They used univariate interpolation, to find the coefficients and the exponents. If the input polynomial f is not s -sparse, one can still run the algorithm. If at any moment while doing the univariate interpolation, it detects more than s many nonzero coefficients, it stops, otherwise it will continue, and indeed at the end, output a wrong s -sparse polynomial \hat{f} , such that (unfortunately) it matches at all the interpolating values. Given s, n, d , the evaluation points on which the interpolation algorithm can be thought as coming from a fixed set.

2.6 Isolation

Let M_δ be the set of monomials in n variables $\mathbf{z} = (z_1, z_2, \dots, z_n)$ of degree bounded by δ ,

$$M_\delta = \{ \mathbf{z}^e \mid \|e\|_1 \leq \delta \}. \quad (3)$$

Note that M_δ is polynomially bounded, for constant δ ,

$$|M_\delta| \leq \binom{n + \delta}{\delta} \leq (n + \delta)^\delta \leq (\delta + 1) n^\delta = O(n^\delta). \quad (4)$$

There is a standard way to map the multivariate monomials in M_δ in a injective way to univariate monomials of polynomial degree. For completeness, we describe the details.

Consider the standard *Kronecker substitution* on M_δ . Define

$$\varphi : z_i \mapsto y^{(\delta+1)^{i-1}}. \quad (5)$$

By extending φ linearly to monomials $\mathbf{z}^e \in M_\delta$, we get

$$\varphi : \mathbf{z}^e \mapsto y^{\sum_{i=1}^n e_i (\delta+1)^{i-1}}, \quad (6)$$

Clearly, φ is injective on M_δ . However, the degree of y can be exponentially large, up to $(\delta + 1)^n$. A way around is to take the exponents modulo some small prime number p . We have to determine p in a way to keep the mapping injective on M_δ . Hence, for any two terms $y^e, y^{e'}$ we get from φ , we have to ensure that $e \not\equiv e' \pmod{p}$. Equivalently $p \nmid (e - e')$.

We have $|e - e'| \leq (\delta + 1)^n$ and, by (4), there are $(\delta + 1)^2 n^{2\delta}$ many pairs e, e' we get from M_δ via φ . Prime p should not divide any of these differences, and hence, p should not divide their product P . The product P is bounded by

$$P \leq ((\delta + 1)^n)^{(\delta+1)^2 n^{2\delta}} = (\delta + 1)^{(\delta+1)^2 n^{2\delta+1}}. \quad (7)$$

Hence, P has at most $\log P \leq R = (\delta + 1)^3 n^{2\delta+1}$ many prime factors. By the Prime Number Theorem, there are more than $\log P$ primes in the set $[R^2]$. Hence, we can find an appropriate prime $p \leq R^2 = n^{O(\delta)}$.

► **Lemma 14.** *There is a prime $p = n^{O(\delta)}$ such that the linear extension of*

$$\varphi_p : z_i \mapsto y^{w_i}, \quad \text{where } w_i = (\delta + 1)^{i-1} \bmod p, \quad \text{for } i = 1, 2, \dots, n, \quad (8)$$

to monomials is injective on M_δ . Moreover, we can find such a p in time $n^{O(\delta)}$ and compute and invert φ_p in time $n^{O(\delta)}$.

Proof. We already argued about the existence of prime p . For the running time, recall that $|M_\delta| = O(n^\delta)$. Therefore we can search for p and check whether it works on M_δ in time $n^{O(\delta)}$. At the same time we can compute pairs of exponents (e, k) such that $\varphi_p(z^e) = y^k$. These pairs can be used to invert φ_p . ◀

The mapping φ_p in Lemma 14 maintains factors of degree δ of a polynomial in the following sense.

► **Lemma 15.** *Let polynomial $f(\mathbf{z})$ factor as $f = gh$, where $g(\mathbf{z})$ has degree δ . Let φ_p be the map from Lemma 14. Then we have $\varphi_p(f) = \varphi_p(g)\varphi_p(h)$, and g can be recovered from $\varphi_p(g)$ in time $n^{O(\delta)}$.*

Note that in Lemma 15, we do *not* claim that irreducibility is maintained: when g is irreducible, still $\varphi_p(g)$ might be reducible. Consider the example $n = \delta = 2$. The weights $\{1, 3\}$ make sure that each monomial $z_1^2, z_1 z_2, z_2^2$ gets mapped to a distinct power in y . Let $g(x, z) = x^2 - z_1 z_2$. Observe that g is irreducible, however $g(x, y, y^3) = (x - y^2)(x + y^2)$ is reducible.

We combine Lemma 14 and Theorem 10 to obtain a projection of a multivariate polynomial to a 3-variate polynomial that maintains irreducibility of polynomials up to degree δ .

► **Corollary 16.** *Let $g(x, \mathbf{z})$ be an irreducible polynomial of constant degree δ with $n + 1$ variables that is monic in x . There exists $\mathbf{w}, \mathbf{w}' \in \mathbb{F}^n$ with $w_i, w'_i = n^{\text{poly}(\delta)}$ such that*

$$\Psi(g) = g(x, y^{w_1}t + y^{w'_1}, \dots, y^{w_n}t + y^{w'_n}) \in \mathbb{F}[x, y, t] \quad (9)$$

is irreducible. Moreover, we can compute and invert $\Psi(g)$ in time $n^{\text{poly}(\delta)}$.

Proof. Let $G(\mathbf{a}, \mathbf{b})$ be the polynomial of degree $2\delta^5$ in $2n$ variables provided by Theorem 10 for g . Let $\mathbf{w}, \mathbf{w}' \in \mathbb{F}^n$ with $w_i, w'_i = n^{\text{poly}(\delta)}$ be the exponents we get from Lemma 14 for G . That is,

$$\widehat{G}(y) = G(y^{w_1}, \dots, y^{w_n}, y^{w'_1}, \dots, y^{w'_n}) \neq 0.$$

Now, suppose that $\Psi(g)$ is reducible. Then it would also be reducible at a point $y = \alpha$, where $\widehat{G}(\alpha) \neq 0$. But then $\widehat{g}(x, t) = \Psi(g)(x, \alpha, t)$ would be reducible too, and this would contradict Theorem 10. We conclude that $\Psi(g)$ is irreducible.

75:12 Derandomizing Multivariate Polynomial Factoring for Low Degree Factors

For the complexity, we first determine prime p from Lemma 14 and then get the weights \mathbf{w}, \mathbf{w}' from above. For a given $g(x, \mathbf{z}) = \sum_{k, \mathbf{e}} c_{k, \mathbf{e}} x^k \mathbf{z}^{\mathbf{e}}$, we can compute $\Psi(g)$ in time $n^{\text{poly}(\delta)}$. For a monomial of g , the mapping looks as follows:

$$c_{k, \mathbf{e}} x^k \mathbf{z}^{\mathbf{e}} \mapsto c_{k, \mathbf{e}} x^k \prod_{i=1}^n (y^{w_i} t + y^{w'_i})^{e_i}. \quad (10)$$

To compute g from $\Psi(g)$, set $t = 0$, i.e. consider $\Psi(g)(x, y, 0)$. From (10) we see that monomials then have the form

$$c_{k, \mathbf{e}} x^k y^{\sum_{i=1}^n e_i w'_i}.$$

From these we get the exponents k and \mathbf{e} similar as in the proof of Lemma 14. \blacktriangleleft

► **Remark.** In Corollary 16, when we say that we *invert* Ψ , it means that for a given $h \in \mathbb{F}[x, y, t]$ which is monic in x with x -degree $\leq \delta$, we either detect that h is not in the codomain of Ψ , or we compute $g \in \mathbb{F}[x, \mathbf{z}]$ such that $\Psi(g) = h$ in time $n^{\text{poly}(\delta)}$.

The inversion can be done similarly as described in the proof of Corollary 16. One can evaluate $t = 0$, and then for every monomial $x^k y^j$, try to find $x^k \mathbf{z}^{\mathbf{e}}$ that would map to such a monomial at $t = 0$. By the property of the map, while mapping the y -degrees, \mathbf{z} -degree could be at most δ , i.e. $\deg(x^k \mathbf{z}^{\mathbf{e}}) \leq \delta$. We will, of course, return empty if the degree of any such monomial, after inverting, becomes $> \delta$. Finally, once we have got a candidate g of degree δ , we still have to check whether $\Psi(g) = h$, because the inversion procedure ignores the variable t . The last step can also be efficiently checked.

The polynomial g of degree δ we considered so far can be thought to be a constant-degree factor of a given polynomial f of degree d . Our goal would be to compute g . It is now easy to extend the above results to hold for all degree- δ factors of f simultaneously.

► **Corollary 17.** *Let $f(x, \mathbf{z})$ be a polynomial of degree d with $n + 1$ variables that is monic in x , and let δ be a constant. There exists $\mathbf{w}, \mathbf{w}' \in \mathbb{F}^n$ with $w_i, w'_i = dn^{\text{poly}(\delta)}$ such that for any irreducible factor g of degree δ of f , we have that $\Psi(g)$ is an irreducible factor of $\Psi(f)$.*

Proof. The proof goes along the lines of Corollary 16, but we choose the weights slightly larger so that the $\widehat{G}(y)$ polynomials for *all* the degree- δ factors g of f are non-zero simultaneously. That is, we choose prime p in Lemma 14 as $p = dn^{\text{poly}(\delta)}$. \blacktriangleleft

Finally, we conclude this subsection by a general remark that whenever n and δ are fixed, these weights are fixed and can be found efficiently.

3 Computing the low-degree factors

For a size measure μ , we consider a class of polynomials $\mathcal{C} = \mathcal{C}_\mu(s, n, d) \subseteq \mathcal{P}(n, d)$ such that \mathcal{C} is closed under derivatives. Many classes \mathcal{C} in the literature fulfill this condition. Useful for us are in particular sparse polynomials, polynomials computed by poly-size any-order ROABPs or by constant-depth circuits. For a constant $\delta \in \mathbb{N}$, let $\mathcal{D} = \mathcal{P}(n, \delta)$.

Our first theorem shows that for any polynomial $f \in \mathcal{C}$, all the factors of f that are in \mathcal{D} can be computed in polynomial time with oracles for PIT for $\text{Hom}[\mathcal{C}]$ and divisibility testing \mathcal{C} by \mathcal{D} .

► **Theorem 18.** *Factor($\mathcal{C}|\mathcal{D}$) can be solved deterministically in time*

$$nd \cdot \text{T}_{\text{PIT}(\text{Hom}[\mathcal{C}])} + d^2 n^{\text{poly}(\delta)} \cdot \text{T}_{\text{Div}(\mathcal{C}/\mathcal{D})} + \text{poly}(s, n^{\text{poly}(\delta)}, d).$$

Proof. Let $f(\mathbf{z}) \in \mathcal{C}$. To compute the factors of degree δ of f , we first do some transformations. The first step is to make f monic in a new variable x via Lemma 7. Let $f_\alpha(x, \mathbf{z})$ be monic.

Then we apply Corollary 17 to $f_\alpha(x, z)$. That is we compute the weights $\mathbf{w}, \mathbf{w}' \in \mathbb{F}^n$ bounded by $dn^{\text{poly}(\delta)}$ and explicitly compute $\Psi(f_\alpha) \in \mathbb{F}[x, y, t]$ of degree at most $\tilde{d} = d^2 n^{\text{poly}(\delta)}$. Note that the x -degree of f_α has not changed by mapping Ψ . In the blackbox case, we can interpolate and reconstruct the polynomial in time $\text{poly}(s\tilde{d})$.

The next step is to factor 3-variate $\Psi(f_\alpha)$. This can be done efficiently Lemma 11. Finding and listing all the irreducible factors takes time $\text{poly}(\tilde{d})$.

Having the factors of $\Psi(f_\alpha)$ in hand, we invert transformations Ψ and τ_α on the factors. Let \tilde{g} be a factor of $\Psi(f_\alpha)$. By Corollary 17, if g indeed corresponds to a δ -degree factor of f , then $\tau_\alpha(g)$ corresponds to a δ -degree factor of $\tau_\alpha(f)$. Therefore, the inverse transformations will yield g . Formally, the factor is $g = \tau_\alpha^{-1}(\Psi^{-1}(\tilde{g}))$.

However \tilde{g} might also *not* correspond to a degree- δ factor of f . In this case, either the inverse transformation does not go through properly, or the degree we get is larger than δ . In these cases, we can immediately throw away \tilde{g} ; see the remark after Corollary 16. But it could also be that we actually obtain a polynomial g of degree δ , just that it is not a factor of f . For that reason, we finally do a divisibility check whether $g|f$. That way we will compute all factors of f of degree δ .

For the time complexity of the factoring algorithm, we have $nd \cdot \text{T}_{\text{PIT}(\text{Hom}[\mathcal{C}]})$ for transforming f to monic f_α by Lemma 7. Time $\text{poly}(dn^{\text{poly}(\delta)})$ is used for map Ψ and the factoring of $\Psi(f_\alpha)$. Similar time is taken to invert and get the candidate factors. Finally, we have at most $d^2 n^{\text{poly}(\delta)}$ candidate polynomials g for which we test divisibility of $g|f$. ◀

► **Remark.** Theorem 18 can be applied in different algebraic models. Furthermore, if a class is closed under highest degree, one can simply assume PIT for \mathcal{C} . In particular, if we work with algebraic formulas, or algebraic branching programs (ABPs), then the above theorem along with the divisibility lemma Lemma 9 implies that we need PIT for the *same* class, to deterministically find the constant-degree factors.

The following pseudo-code summarizes the algorithm given in the proof of Theorem 18.

■ **Algorithm 1** Computing factors of degree $\leq \delta$.

Input : $f(z)$, s , and δ , where f is an n -variate polynomial of degree d such that $\mu(f) \leq s$, and δ is a constant.

Output : A list of irreducible polynomials of degree $\leq \delta$, which are factors of f , along with the factor-multiplicities.

- 1 Set the output list $L = \emptyset$. Set the intermediate candidates list $L' = \emptyset$.
- 2 Make a monic transformation $\tau_\alpha : z_i \mapsto \alpha_i x + z_i$, according to Lemma 7. Let $f_\alpha(x, z) := \tau_\alpha(f)$.
- 3 Find weights \mathbf{w}, \mathbf{w}' bounded by $dn^{\text{poly}(\delta)}$ according to Corollary 17 and compute $\Psi(f_\alpha) \in \mathbb{F}[x, y]$ in dense representation.
- 4 Factorize the trivariate polynomial $\Psi(f_\alpha)$ according to Lemma 11. Let S be the set of all $\leq \delta$ degree factors in x of $\Psi(f_\alpha)$ in dense representation.
- 5 **for** $\tilde{g} \in S$ **do**
 - 6 /* Computing candidate factors via divisibility */
 - 7 Compute $\hat{g} = \Psi^{-1}(\tilde{g})$ (if the inverse exists) of degree $\leq \delta$ by Corollary 16.
 - 8 Compute $g = \tau_\alpha^{-1}(\hat{g})$.
 - 9 If $g|f$ then add g to L' .
- 9 **for** $g \in L'$ **do**
 - 10 /* Computing multiplicities via Lemma 12 */
 - 11 Let $z \in \{z_1, \dots, z_n\}$ be any variable that g depends on, so that $\partial_z(g) \neq 0$.
 - 12 Find the smallest $e \geq 0$ such that $g \nmid \frac{\partial^e f}{\partial z^e}$.
 - 13 **if** $e > 1$ **then** add (g, e) to the list L .
- 13 **return** L

4 Computing the factors of a constant degree product

Now, we want to output the constant-degree irreducible factors in the promise case.

It is still an open question to test in deterministic polynomial time if a quadratic polynomial (or any non-linear polynomial whose total degree is upper bounded by a constant) divides another polynomial f , even when f is a sparse polynomial. In the above theorem f may not be sparse and we have only blackbox access to it. Thus, the proof for Theorem 2 is a bit different from Theorem 1.

► **Theorem 19.** *Given blackbox access to an n -variate degree- d polynomial $f = \prod_{i=1}^s g_i^{e_i}$, where g_i are irreducible polynomials with $\deg(g_i) \leq \delta$, one can deterministically output all (g_i, e_i) in time $\text{poly}(dn^{\text{poly}(\delta)})$.*

Proof. The first step of the algorithm is to make $f(\mathbf{z})$ monic in a new variable x via Lemma 7. One can find an $\alpha \in (\mathbb{F} \setminus \{0\})^n$ in $\text{poly}(d, n^\delta)$ time by using the hitting set for polynomials of degree $\leq \delta$; see [7, 5].

Next step is to apply Corollary 16 to $f_\alpha(x, \mathbf{z})$. Find the weights $\mathbf{w}, \mathbf{w}' \in \mathbb{F}^n$, each bounded by $dn^{\text{poly}(\delta)}$. Observe that $\Psi(f_\alpha) \in \mathbb{F}[x, y, t]$, of degree at most $\tilde{d} := d^2 n^{\text{poly}(\delta)}$. By the same lemma, we know that for any irreducible factor g of f , we have that $\Psi(g)$ is an irreducible factor of $\Psi(f)$.

The next step is to explicitly compute the trivariate polynomial $\Psi(f_\alpha)$. Since, the degree of the polynomial is at most \tilde{d} , one can interpolate and reconstruct the polynomial, from its blackbox access, in time $\text{poly}(s\tilde{d})$. Note that $s \leq d$.

The next step is to factorize $\Psi(f_\alpha)$. This can be done efficiently using Lemma 11. Finding and listing all the irreducible factors takes time $\text{poly}(\tilde{d})$.

The next step is to invert the transformation Ψ^{-1} on the factors computed in the previous step. This can be done efficiently in time $\text{poly}(dn^{\text{poly}(\delta)})$; see Corollary 16 and its remark. Let \tilde{g} be a factor of $\varphi(f_\alpha)$. Output $g = \tau_\alpha^{-1}(\Psi^{-1}(\tilde{g}))$.

■ **Algorithm 2** Promise factors of degree $\leq \delta$.

Input : An n -variate, degree d polynomial $f(\mathbf{z})$, and a constant δ , and a promise that all its irreducible factors have degree $\leq \delta$.

Output : All the irreducible factors of f , along with the multiplicities.

- 1 Set the output list $L = \emptyset$.
 - 2 Make a monic transformation $\tau_\alpha : z_i \mapsto \alpha_i x + z_i$, according to Lemma 7. Let $f_\alpha(x, \mathbf{z}) := \tau_\alpha(f)$.
 - 3 Find weights \mathbf{w}, \mathbf{w}' bounded by $dn^{\text{poly}(\delta)}$ according to Corollary 17 and compute $\Psi(f_\alpha) \in \mathbb{F}[x, y]$ in dense representation.
 - 4 Factorize the trivariate polynomial $\Psi(f_\alpha)$ according to Lemma 11. Let S be the set of irreducible factors of $\Psi(f_\alpha)$ along with its multiplicities as a tuple.
 - 5 **for** $(\tilde{g}, e) \in S$ **do**
 - 6 /* Computing the irreducible factors via inversion */
 - 6 Compute $\hat{g} = \Psi^{-1}(\tilde{g})$, by Corollary 16 and its remark.
 - 7 Compute $g = \tau_\alpha^{-1}(\hat{g})$, and add (g, e) to L .
 - 8 **return** L
-

Now, we discuss the correctness of the output. By assumption, $f = g_1^{e_1} \cdots g_s^{e_s}$ where $g_i \in \mathcal{D}$. Therefore, $\Psi(\tau_\alpha(f)) = \Psi(\tau_\alpha(g_1))^{e_1} \cdots \Psi(\tau_\alpha(g_s))^{e_s}$. Furthermore, by choice of \mathbf{w}, \mathbf{w}' , we know the following facts.

1. The polynomials $\Psi(\tau_\alpha(g_i)) \in \mathbb{F}[x, y, t]$ are irreducible over \mathbb{F} , for all $i = 1, \dots, s$.
2. $\Psi(\tau_\alpha(g_i)) \neq \Psi(\tau_\alpha(g_j))$, for $i \neq j$.
3. One can uniquely recover g_i from $\Psi(\tau_\alpha(g_i))$, using Corollary 16 and its remark.

This implies that the factoring pattern of f and $\Psi(\tau_\alpha(f))$ remain the same, and can be recovered by simply looking at the factorization of $\Psi(\tau_\alpha(f))$.

Time complexity. To make f monic, we find vector α in time $\text{poly}(dn^\delta)$. Interpolation and trivariate factorization of a degree- \tilde{d} polynomial takes time $\text{poly}(\tilde{d}) = \text{poly}(dn^{\text{poly}(\delta)})$, see Lemma 11. Inverting each of them to get the original factor also takes time $\text{poly}(dn^{\text{poly}(\delta)})$. \blacktriangleleft

5 Finding sparse factors reduces to sparse irreducibility

Let f be an n -variate irreducible polynomial of degree d . Let $\alpha \in (\mathbb{F} \setminus \{0\})^n$, such that

$$f_d(\alpha) \neq 0, \text{ where } f_d = \text{Hom}_d[f] \text{ is the homogeneous degree } -d \text{ component of } f.$$

Using Lemma 7 and Lemma 8, one can conclude that $\hat{f}(x, \mathbf{z}) := f(\alpha x + \mathbf{z})$ is a monic irreducible $(n+1)$ -variate polynomial of degree d .

On the other hand Theorem 10 shows that for a *random* $\beta, \gamma \in \mathbb{F}^n$, the bivariate polynomial

$$f(\alpha x + \beta t + \gamma) = \hat{f}(x, \beta t + \gamma) \in \mathbb{F}[x, t],$$

remains irreducible. When the degree of f is a constant, such an irreducibility preserving reduction can be efficiently derandomized; see Corollary 16. Formally, we should think of it as a derandomization for the class of constant degree polynomials. In Corollary 16, one can think of evaluating y at polynomially many points to find the right β, γ , while the point α comes from an efficient PIT algorithm for f_d . Note that *any* $\alpha \in (\mathbb{F} \setminus \{0\})^n$ suffices as long as $f_d(\alpha) \neq 0$.

Let $\alpha \in (\mathbb{F} \setminus \{0\})^n$. We define a set $S_\alpha(s, n, d)$ as follows.

$$S_\alpha(s, n, d) := \{f \in \mathcal{P}(n, d) \mid f \text{ is irreducible, } \text{sp}(f) \leq s, \text{ and } \text{Hom}_{\text{deg}(f)}[f](\alpha) \neq 0\}. \quad (11)$$

Motivated by the efficient derandomization of irreducibility preserving bivariate projections for constant degree polynomials, we assume the following.

► **Assumption 1 (Sparse irreducible projection).** Let $\alpha \in (\mathbb{F} \setminus \{0\})^n$, and $S_\alpha(s, n, d) \subseteq \mathcal{P}(n, d)$ as defined in Equation (11). Then, there is a deterministic subexponential time algorithm to find an explicit set $\mathcal{H}_\alpha \subseteq \mathbb{F}^{2n}$ of size subexponential, such that for any $f \in S_\alpha(s, n, d)$, there exists $(\beta, \gamma) \in \mathcal{H}_\alpha$ such that $f(\alpha x + \beta t + \gamma)$ remains irreducible.

► **Remark.** Assuming the above, one can decide whether an s -sparse degree- d polynomial f is irreducible or not in subexponential time. To do this, one can find $\alpha \in (\mathbb{F} \setminus \{0\})^n$ such that $\text{Hom}_d[f](\alpha) \neq 0$, in time $\text{poly}(snd)$, using [24]. Hence, one can find a set \mathcal{H}_α such that $f(\mathbf{z})$ is reducible if and only if $f(\alpha x + \beta t + \gamma)$ is reducible, for every $(\beta, \gamma) \in \mathcal{H}_\alpha$. Whether a bivariate polynomial is reducible can be checked in time $\text{poly}(d)$ (Lemma 11).

75:16 Derandomizing Multivariate Polynomial Factoring for Low Degree Factors

Using Assumption 1, one can design an efficient deterministic algorithm to output sparse irreducible factors of a sparse polynomial.

► **Theorem 20** (Efficient sparse factoring). *Let $f \in \mathbb{F}[z]$ be an s -sparse polynomial of degree d . If Assumption 1 holds, then there is a deterministic subexponential time algorithm that outputs all its irreducible factors with sparsities $\leq s$, along with the multiplicities.*

Proof. Assume that g is a s -sparse irreducible factor of f with multiplicity e , i.e., $f = g^e \cdot R$, where $\gcd(g, R) = 1$. Assume that $\deg(g) = d_1$, and $\deg(R) = d_2$. We will argue that Algorithm 3 correctly outputs (g, e) . Let $\alpha \in (\mathbb{F} \setminus \{0\})^n$, such that $\text{Hom}_d[f](\alpha) \neq 0$. Observe that $\text{Hom}_d[f] = (\text{Hom}_{d_1}[g])^e \cdot \text{Hom}_{d_2}[R]$. Therefore, it must hold that $\text{Hom}_{d_1}[g](\alpha) \neq 0$, implying $g \in S_\alpha(s, n, d)$.

By Assumption 1, we know that there exists a subexponential time algorithm to find a set \mathcal{H}_α such that $g(\alpha x + \beta t + \gamma)$ remains irreducible for some $(\beta, \gamma) \in \mathbb{F}^{2n}$. We call (β, γ) a “good” point for g .

For such a good point, the set S in Line 6 of Algorithm 3 must contain $g(\alpha x + \beta t + \gamma)$. Pick any $c \in \mathbb{F}^n$. Observe that $\tilde{g}(x, t_1, t_2) := \phi_c(g) = g(\alpha x + \beta t_1 + (c - \gamma)t_2 + \gamma)$ is a monic polynomial in x . Further, \tilde{g} remains irreducible, since $\tilde{g}(x, t, 0) = g(\alpha x + \beta t + \gamma)$ is irreducible by the choice of a good point (β, γ) . Hence, S' in Line 10 must contain the polynomial \tilde{g} .

One can find the corresponding factor in Line 12, and suppose the corresponding index is j . Note that $\tilde{g}(0, 0, 1) = g(c)$.

From the above, one can conclude that the L'_j contains $(j, c, g(c))$. One can now do the sparse interpolation using [24], to reconstruct g ; see Lemma 13. Since $g \mid f$, this is correctly detected and added to the list L' . This is being discussed in Line 15-16.

Further, by our choice of a good point $(\beta, \gamma) \in \mathcal{H}_\alpha$, the bivariate $g(\alpha x + \beta t + \gamma)$ remains irreducible, and hence it successfully passes Line 19. Since g is a nontrivial polynomial, one can find a variable $z \in \{z_1, \dots, z_n\}$ such that $\partial_z(g) \neq 0$ in Line 20. Using Lemma 12, one can conclude that indeed Line 22 adds (g, e) to the list L .

From the above analysis, we know that Algorithm 3 always outputs g with the corresponding multiplicity. On the other hand, Line 15 makes sure that the candidate polynomial is indeed at most s -sparse (see Lemma 13) and Line 16, by the divisibility testing, makes sure it detects a wrong sparse factor. What could have happened is L' contains factors which are s -sparse reducible polynomials dividing f .

Line 19 again uses Assumption 1 to check if it is indeed irreducible or not. Finally, Line 20-22 make sure that the Algorithm 3 never outputs the correct multiplicity. This finishes the correctness of the algorithm.

Running time analysis. Since f is s -sparse, one can find an $\alpha \in \mathbb{F}^n$, such that $\text{Hom}_d[f](\alpha) \neq 0$, in $\text{poly}(snd)$ time [24].

Line 6-10 take $\text{poly}(d)$ time, since bivariate/trivariate interpolation and factorization can be done efficiently Lemma 11.

Line 15 can be done using the sparse interpolation algorithm in $\text{poly}(snd)$ time [24]. Line 16 requires whether a given s -sparse polynomial P_j divides another s -sparse polynomial f or not. Using the techniques from [9] (Lemma 9), this divisibility question can be reduced to a PIT instance of a constant-depth circuit, for which there is a subexponential time algorithm [31].

Line 19 is again bivariate factorization Lemma 11 which can be done in $\text{poly}(d)$ time.

Further, $\frac{\partial^e f}{\partial z^e}$ is at most s -sparse. Hence, the divisibility question of whether $P \mid \frac{\partial^e f}{\partial z^e}$ in Line 21, can be similarly done in subexponential time. Finally, since $|\mathcal{H}_\alpha|$ is subexponentially large, the overall running time remains subexponential. ◀

■ **Algorithm 3** Computing s -sparse factors.

Input : An n -variate, degree d polynomial s -sparse polynomial $f(\mathbf{z})$.

Output : A list of irreducible s -sparse polynomials which are factors of f , along with the factor-multiplicities.

```

1 Set the output list  $L = \emptyset$ . Set the intermediate candidate list  $L' = \emptyset$ .
2 Use Lemma 13 to find an  $\alpha \in (\mathbb{F} \setminus \{0\})^n$  such that  $\text{Hom}_d[f](\alpha) \neq 0$ .
3 Use Assumption 1 to find a set  $\mathcal{H}_\alpha$ .
4 for each  $(\beta, \gamma) \in \mathcal{H}_\alpha$  do
5   Let  $\phi : z_i \mapsto \alpha_i x + \beta_i t + \gamma_i$ . Compute  $\hat{f} := \phi(f) \in \mathbb{F}[x, t]$ , as a dense
   representation.
6   Factorize the bivariate polynomial  $\hat{f}$  over  $\mathbb{F}$ . Let  $S = \{g_1(x, t), \dots, g_m(x, t)\}$  be
   the set of its irreducible factors.
7   Set  $m$  many interpolating lists  $L'_j = \emptyset$ , for  $j \in [m]$ .
8   Fix  $\mathbf{c} = (c_1, \dots, c_n) \in \mathbb{F}^n$ . /* These points are the evaluation points
   for which  $s$ -sparse interpolation succeeds */
9   For new variables  $t_1$  and  $t_2$ , define a new map
    $\phi_{\mathbf{c}} : z_i \mapsto \alpha_i x + \beta_i t_1 + (c_i - \gamma_i)t_2 + \gamma_i$ . Compute  $\tilde{f}_{\mathbf{c}} := \phi_{\mathbf{c}}(f) \in \mathbb{F}[x, t_1, t_2]$ , as a
   dense representation.
10  Factorize the trivariate  $\tilde{f}_{\mathbf{c}}$  over  $\mathbb{F}$ . Let  $S' = \{h_1(x, t_1, t_2), \dots, h_r(x, t_1, t_2)\}$  be the
   set of its irreducible factors.
11  for  $h_i \in S$  do
12    /* Computing the unique correspondence between bivariate and
   trivariate factors, and the evaluations */
13    Find the unique  $j \in [m]$  such that  $h_i(x, t, 0) = g_j(x, t)$ , if exists, otherwise go
   to the next factor in  $S$ .
14    Evaluate  $h_i(0, 0, 1)$ , and add  $(j, \mathbf{c}, h_i(0, 0, 1))$  to  $L'_j$ .
15  for  $j \in [m]$  do
16    /* Computing candidate sparse factors via interpolation and
   divisibility */
17    Use sparse interpolation algorithm (Lemma 13) to find an  $s$ -sparse polynomial
    $P_j$ , if exists, such that  $P_j(\mathbf{c}) = \theta$  where  $(j, \mathbf{c}, \theta) \in L'_j$ .
18    Check if  $P_j \mid f$ , or not. If yes, then update  $L' = \{P_j\} \cup L'$ .
19  return  $L'$ 
20 for each  $P \in L'$  do
21   /* Deciding irreducibility and computing multiplicities via
   Lemma 12 */
22   Check if  $P$  is irreducible, using Assumption 1 and its remark. If it is not
   irreducible, STOP, and go to the next polynomial in  $L'$ .
23   Otherwise, let  $z \in \{z_1, \dots, z_n\}$  be any variable that  $P$  depends on, so that
    $\partial_z(P) \neq 0$ .
24   Find the smallest  $e \geq 1$  such that  $P \nmid \frac{\partial^e f}{\partial z^e}$  and add  $(P, e)$  to the list  $L$ .
25 return  $L$ 

```

► Remarks.

1. Assumption 1 is true for constant degree polynomials and can be solved in polynomial time Corollary 16. Therefore, Algorithm 3 can be used to give an alternative proof of Theorem 18. This is because for $f \in \mathcal{C}$, we need to find α , such that $\text{Hom}_d[f](\alpha) \neq 0$, which is given by the PIT oracle for \mathcal{C} . Once the α is found, \mathcal{H}_α can be found for the constant-degree polynomials, that preserves irreducibility; see Corollary 16. Additionally, the algorithm requires some divisibility testing by the candidate constant-degree polynomials, which is done using the divisibility testing $\text{Div}(\mathcal{C}/\mathcal{D})$.
2. Theorem 20 can be generalized to the input and output polynomials being computed by constant-depth circuits, by analogously changing the Assumption 1 for constant-depth circuits. In this case, we will only be able to output the factors as blackbox (because efficient reconstruction for constant-depth circuits is still unknown). Note that whether a constant-depth circuit divides another constant-depth circuit can be deterministically decided in subexponential time.

6 Conclusion

We conclude with some open questions.

1. Can we decide whether a given sparse polynomial is irreducible in deterministic subexponential time? The proof may already give a good bivariate projection that preserves irreducibility. Then Theorem 20 would give us a deterministic subexponential-time algorithm to find irreducible sparse factors of a sparse polynomial.
2. Can we find *bounded individual degree* sparse factors of a sparse polynomial (without any bound on the individual degree) in deterministic quasipolynomial time? Volkovich asked if multilinear factors of a sparse polynomial can be found in deterministic polynomial time [39].
3. Can one compute all the factors of a sparse polynomial/constant depth circuit by constant depth circuits of small size? At least, can one find all the factors that are computable in constant depth? The recent result in [29] gives a deterministic subexponential-time algorithm that outputs a list of circuits (of unbounded depth and possibly with division gates) that includes all such factors.
4. Given a blackbox computing the product of sparse irreducible polynomials f_i with bounded individual degree, find f_i 's in deterministic polynomial time. [3] gives a quasipolynomial time algorithm, when the input is sparse with constant individual degree and the factors are all sparse (polynomially upper bounded with respect to input polynomial's sparsity).

References

- 1 Sanjeev Arora and Madhu Sudan. Improved low-degree testing and its applications. *Combinatorica*, 3(23), 2003.
- 2 Elwyn R Berlekamp. Factoring polynomials over large finite fields. *Mathematics of computation*, 24(111):713–735, 1970.
- 3 Vishwas Bhargava, Shubhangi Saraf, and Ilya Volkovich. Deterministic factorization of sparse polynomials with bounded individual degree. *Journal of the ACM (JACM)*, 67(2):1–28, 2020.
- 4 Pranav Bisht and Ilya Volkovich. On solving sparse polynomial factorization related problems. In *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2022)*. Schloss-Dagstuhl – Leibniz-Zentrum für Informatik, 2022.
- 5 Markus Bläser and Anurag Pandey. Polynomial identity testing for low degree polynomials with optimal randomness. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

- 6 Andrej Bogdanov. Pseudorandom generators for low degree polynomials. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 21–30, 2005.
- 7 Nader Bshouty. Testers and their applications. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 327–352, 2014.
- 8 Ashish Dwivedi, Zeyu Guo, and Ben Lee Volk. Optimal pseudorandom generators for low-degree polynomials over moderately large fields. *arXiv preprint*, 2024. [arXiv:2402.11915](https://arxiv.org/abs/2402.11915).
- 9 Michael A Forbes. Deterministic divisibility testing via shifted partial derivatives. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 451–465. IEEE, 2015.
- 10 Michael A Forbes, Ramprasad Saptharishi, and Amir Shpilka. Hitting sets for multilinear read-once algebraic branching programs, in any order. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 867–875, 2014.
- 11 Zeyu Guo and Rohit Gurjar. Improved explicit hitting-sets for roabps. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2020.
- 12 Ankit Gupta. Algebraic geometric techniques for depth-4 pit & sylvester-gallai conjectures for varieties. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 21, 2014.
- 13 Rohit Gurjar, Arpita Korwar, Nitin Saxena, and Thomas Thierauf. Deterministic identity testing for sum of read-once oblivious arithmetic branching programs. *computational complexity*, 26:835–880, 2017.
- 14 Ming-Deh Huang and Yiu-Chung Wong. Extended hilbert irreducibility and its applications. *Journal of Algorithms*, 37(1):121–145, 2000.
- 15 Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. In *Proceedings of the 35th ACM Symposium on Theory of Computing (STOC)*, pages 355–364. ACM, 2003.
- 16 Erich Kaltofen. Computing with polynomials given by straight-line programs I: greatest common divisors. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 131–142, 1985.
- 17 Erich Kaltofen. Effective hilbert irreducibility. *Information and Control*, 66(3):123–137, 1985.
- 18 Erich Kaltofen. Polynomial-time reductions from multivariate to bi-and univariate integral polynomial factorization. *SIAM Journal on Computing*, 14(2):469–489, 1985.
- 19 Erich Kaltofen. Factorization of polynomials given by straight-line programs. *Randomness and Computation*, 5:375–412, 1989.
- 20 Erich Kaltofen. Effective noether irreducibility forms and applications. *Journal of Computer and System Sciences*, 50(2):274–295, 1995.
- 21 Erich Kaltofen and Barry M. Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *Journal of Symbolic Computation*, 9(3):301–320, 1990.
- 22 Neeraj Kayal. Efficient algorithms for some special cases of the polynomial equivalence problem. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1409–1421. SIAM, 2011. [doi:10.1137/1.9781611973082.108](https://doi.org/10.1137/1.9781611973082.108).
- 23 Neeraj Kayal. Affine projections of polynomials. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 643–662, 2012.
- 24 Adam R Klivans and Daniel Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 216–223, 2001.
- 25 Pascal Koiran and Nicolas Ressayre. Orbits of monomials and factorization into products of linear forms. *arXiv preprint*, 2018. [arXiv:1807.03663](https://arxiv.org/abs/1807.03663).
- 26 Pascal Koiran and Mateusz Skomra. Derandomization and absolute reconstruction for sums of powers of linear forms. *Theoretical Computer Science*, 887:63–84, 2021.

- 27 Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka. Equivalence of polynomial identity testing and polynomial factorization. *computational complexity*, 24(2):295–331, 2015.
- 28 Mrinal Kumar, Varun Ramanathan, and Ramprasad Saptharishi. Deterministic algorithms for low degree factors of constant depth circuits. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3901–3918. SIAM, 2024.
- 29 Mrinal Kumar, Varun Ramanathan, Ramprasad Saptharishi, and Ben Lee Volk. Towards deterministic algorithms for constant-depth factors of constant-depth circuits. *arXiv preprint*, 2024. [arXiv:2403.01965](https://arxiv.org/abs/2403.01965).
- 30 Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische annalen*, 261:515–534, 1982.
- 31 Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. Superpolynomial lower bounds against low-depth algebraic circuits. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 804–814. IEEE, 2021.
- 32 Dori Medini and Amir Shpilka. Hitting sets and reconstruction for dense orbits in $vp_{\{e\}}$ and $\sigma\pi\sigma$ circuits. In *36th Computational Complexity Conference (CCC 2021)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2021.
- 33 C Ramya and BV Raghavendra Rao. Linear projections of the vandermonde polynomial. *Theoretical Computer Science*, 795:165–182, 2019.
- 34 Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *computational complexity*, 14:1–19, 2005.
- 35 Nitin Saxena. Diagonal circuit identity testing and lower bounds. In *Automata, Languages and Programming: 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I 35*, pages 60–71. Springer, 2008.
- 36 Amir Shpilka. Interpolation of depth-3 arithmetic circuits with two multiplication gates. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 284–293, 2007.
- 37 Gaurav Sinha. Reconstruction of real depth-3 circuits with top fan-in 2. In *31st Conference on Computational Complexity (CCC 2016)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2016.
- 38 Volker Strassen. Vermeidung von divisionen. *Journal für die reine und angewandte Mathematik*, 264:184–202, 1973.
- 39 Ilya Volkovich. Deterministically factoring sparse polynomials into multilinear factors and sums of univariate polynomials. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015.
- 40 Ilya Volkovich. On some computations on sparse polynomials. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017.
- 41 Joachim von zur Gathen and Erich Kaltofen. Factoring sparse multivariate polynomials. *Journal of Computer and System Sciences*, 31(2):265–287, 1985.