Electronic Colloquium on Computational Complexity, Report No. 36 (2026)

On Factorization of Sparse Polynomials of Bounded Individual Degree

Aminadav Chuyoon[*]
chuyoon1@mail.tau.ac.il

Amir Shpilka[*]
shpilka@tauex.tau.ac.il

Abstract

We study sparse polynomials with bounded individual degree and the class of their factors. In particular we obtain the following algorithmic and structural results:

(1) A deterministic polynomial-time algorithm for finding *all the sparse divisors* of a sparse polynomial with bounded individual degree. As part of this, we establish the first upper bound on the number of non-monomial irreducible factors of such polynomials.

(2) A $\mathrm{poly}(n, s^{d \log \ell})$-time algorithm for recovering ℓ irreducible s-sparse polynomials of bounded individual degree d from blackbox access to their product (which is not necessarily sparse). This partially resolves a question posed in [DST24]. In particular, when $\ell = O(1)$, the algorithm runs in polynomial time.

(3) Deterministic algorithms for factoring a *product* of s-sparse polynomials of bounded individual degree d from blackbox access. Over fields of characteristic zero or sufficiently large, the algorithm runs in $\mathrm{poly}(n, s^{d^3 \log n})$-time; over arbitrary fields it runs in $\mathrm{poly}(n, (d^2)!, s^{d^5 \log n})$-time. This improves upon the algorithm of [BSV20], which runs in $\mathrm{poly}(n, s^{d^7 \log n})$-time and applies only to a single sparse polynomial of bounded individual degree. In the case where the input is a single sparse polynomial, we give an algorithm that runs in $\mathrm{poly}(n, s^{d^2 \log n})$-time.

(4) Given blackbox access to a *product* of (not necessarily sparse or irreducible) *factors of sparse polynomials of bounded individual degree*, we give a deterministic polynomial-time algorithm for finding all irreducible sparse multiquadratic factors of it (along with their multiplicities). This generalizes the algorithms of [Vol15] and [Vol17]. We also show how to decide whether such a product is a complete power (in case it is defined over a field of zero or large enough characteristic), extending the algorithm of [BV25].

Our algorithms most naturally apply over fields of zero or sufficiently large characteristic. To handle arbitrary fields, we introduce the notion of *primitive divisors* for a class of polynomials, which may be of independent interest. This notion enables us to adapt ideas of [BV25] and remove characteristic assumptions from most of our algorithms.

[*]School of Computer Science and AI, Tel Aviv University. This research was funded by the European Union (ERC, EACTP, 101142020). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

ISSN 1433-8092

Contents

1 Introduction

1.1 Background and Motivation

This paper is motivated by numerous open problems concerning the factorization and complexity of factors of multivariate sparse polynomials. These are n-variate polynomials of degree $\mathrm{poly}(n)$ whose number of monomials is also $\mathrm{poly}(n)$. In particular, this is the class of polynomials that have polynomial size $\Sigma\Pi$ circuits, which is one of the simplest classes from the algebraic complexity point of view.

Despite being structurally simple, sparse polynomials form a rich and non-trivial class of polynomials that has been extensively studied. Efficient deterministic algorithms for polynomial identity testing and reconstruction were developed for this class [GK87, BOT88, GKS90, CDGK91, KS01]. In particular, the work of Klivans and Spielman [KS01] introduced a special reduction from multivariate sparse polynomials to univariate polynomials of polynomial degree, over any field, that has since found many applications, mainly for questions concerning polynomial identity testing and reconstruction. See the surveys [Sax09, SY10, Sax14, DG24].

In this work we are mostly interested in understanding factors of sparse polynomials. The seminal work of Kaltofen [Kal85a] shows that factors of sparse polynomials have polynomial-size algebraic circuits. A recent breakthrough result of [BKR+25] implies that they even have polynomial-size bounded-depth circuits.

However, an intriguing question that remains unsolved for over forty years concerns the complexity of factors of sparse polynomials in terms of their number of monomials. In other words, how many monomials can a factor of a sparse polynomial have. The following example of von zur Gathen and Kaltofen demonstrates that the sparsity of irreducible factors of polynomials may be super-polynomial.

Example 1.1 (Example of [VZGK85])**.** *Consider*

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

It is not hard to prove that h is irreducible over $\mathbb{Q}$. Let

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

Then, f has $(2^{n+1} - 1)$ monomials, and an irreducible factor h of sparsity n^n, which is super-polynomial in the sparsity of f.

This led them to pose the following question.

Question 1.2 ([VZGK85])**.** *Can the output size for the factoring problem actually be more than quasi-polynomial in the sparsity of the input?*

This question is still wide open with only several special cases solved. In [Vol15] Volkovich proved that multilinear factors of sparse polynomials are sparse. In a follow-up work [Vol17] he proved that all factors of multiquadratic sparse polynomials are themselves sparse. This result was extended to higher individual degrees by Bhargava, Saraf, and Volkovich [BSV20], who considered sparse polynomials with bounded individual degree d, and proved a quasi-polynomial upper bound on the sparsity of factors of such polynomials. In particular, if f is an s-sparse

n-variate polynomial with individual degrees bounded by d, then every factor of f has at most $s^{O(d^2 \log n)}$ monomials (see Theorem 2.14). This is the first nontrivial sparsity bound for factors in the regime $d > 2$. While the result of [BSV20] gives a quasi-polynomial upper bound on sparsity of factors of polynomials with bounded individual degree, it is believed that the "real" bound should be polynomial. In [BV25], the authors provide some evidence for that by giving polynomial-time algorithms for various problems for which the existence of such algorithms is implied by the conjectured polynomial upper bound on sparsity of factors.

In their work von zur Gathen and Kaltofen gave a randomized algorithm for factorization of sparse polynomials and noted another obstacle towards obtaining deterministic factorization algorithms:

"Also note that there does not seem to be a feasible way of checking deterministically whether the output is correct, i.e., whether $f = \prod_i {f_i}^{e_i}$" [VZGK85].

In particular, the following question is open.

Question 1.3 ([DST24, Question 1])**.** *Can we find the factorization of polynomials whose irreducible factors are sparse?*

In fact, Dutta et al. also asked the following special case of Question 1.3.

Question 1.4 ([DST24, Section 5])**.** *Given a blackbox computing the product of sparse irreducible polynomials f_i with bounded individual degree, find the f_i's in deterministic polynomial time.*

The aforementioned results of [Vol15, Vol17, BSV20] also have consequences for Question 1.3 and Question 1.4. In [Vol15] Volkovich gave a deterministic polynomial-time algorithm for factorization of sparse polynomials whose irreducible factors are all multilinear (hence sparse) polynomials. In [Vol17] he gave an efficient deterministic algorithm for factoring multiquadratic sparse polynomials. In [BSV20] the authors combined their sparsity bound with techniques similar to [Vol15, Vol17] to obtain a deterministic quasi-polynomial-time algorithm for factoring sparse polynomials of bounded individual degree. Dutta, Sinhababu, and Thierauf [DST24] gave efficient deterministic reconstruction of constant degree factors of sparse polynomials.

One notable limitation of the algorithm of [BSV20] is that it does not give an efficient way to find the *sparse factors* of the input polynomial, even if such factors exist. This problem is closely related to the following question asked by Dutta, Sinhababu and Thierhauf.

Question 1.5 ([DST24, Question 2])**.** *Design a better-than-exponential time deterministic algorithm that outputs all the sparse irreducible factors of a sparse polynomial.*

Another outstanding open problem concerns the reconstruction of rational functions whose numerator and denominator are sparse polynomials. Surprisingly, the best known deterministic algorithm for this question runs in exponential time [GKS94]. While seemingly unrelated, special cases of this question lie at the heart of our algorithms, and we solve it in two special cases where more information is provided.

Many more open problems were asked for sparse polynomials and the reader is referred to [DST24] and [FS15].

1.2 Our Results

Before presenting our results, let us introduce the following notation.

Definition 1.6. We denote the class of n-variate, s-sparse polynomials of bounded individual degree d, by $\mathcal{P}(n, s, d)$. We refer to polynomials in this class as (n, s, d)-sparse polynomials. It turns out that in many cases it makes sense to talk about the class of *divisors* of (n, s, d)-sparse polynomials; we denote this class of polynomials by $\mathcal{P}_{\text{Factors}}(n, s, d)$.

We first summarize our main contributions informally as follows:

(1) We prove a sharp upper bound on the number of irreducible factors of (n, s, d)-sparse polynomials (Theorem 1.8).

(2) Using this bound, we give the first deterministic polynomial-time algorithm that constructs *all* sparse divisors of (n, s, d)-sparse polynomials (Theorem 1.9).

(3) We give a deterministic blackbox factorization algorithm for products of sparse (n, s, d)-sparse polynomials in quasi-polynomial time in the number of terms. This solves Question 1.4 when the number of terms is constant (Theorem 1.10). We note that the running time of this algorithm is polynomial in the number of variables n.

Remark 1.7. In stating our results we suppress the precise dependence on the field (beyond its characteristic) and on bit complexity. As usual in factorization algorithms, an additional factor reflecting the complexity of univariate factorization over the field should be included. These bounds are stated in Section 2.2.

We now give the formal statement of our results. Our first main result gives an upper bound for the number of sparse divisors of (n, s, d)-sparse polynomials. While its proof is not difficult, it is instrumental in our algorithms and demonstrates the strong structural limitations that sparsity imposes.

Theorem 1.8 (Divisor bound for (n, s, d)-sparse polynomials). *Let $f \in \mathcal{P}(n, s, d)$. The number of non-monomial irreducible factors of f, counted with multiplicities, is at most $d \log s$. In particular, the number of divisors of an s-sparse polynomial f with bounded individual degree d, that are not divisible by any monomial, is at most $2^{d \log s} = s^d$.*

Our first main algorithmic result shows how to deterministically output all sparse divisors of such polynomials in polynomial time. Earlier, only a quasi-polynomial-time algorithm was known [BSV20].

Theorem 1.9 (Finding all sparse divisors of (n, s, d)-sparse polynomials). *There is a deterministic, $\text{poly}(n, d!, s^d)$-time algorithm which, given $f \in \mathcal{P}(n, s, d)$, outputs all the s-sparse divisors of f that are not divisible by any monomial, as well as the monomial of maximal degree dividing f.*

Our second main algorithmic result answers Question 1.4 when there are constantly many polynomials in the product, and in the general case it provides a quasi-polynomial-time algorithm.

Theorem 1.10 (Factoring products of (n, s, d)-sparse polynomials). *Let $\mathbb{F}$ be a field of characteristic zero or larger than $2d$. There is a deterministic $\text{poly}(n, d^d, s^{d \log \ell}, \ell^d)$-time algorithm which, given blackbox access to a product $f = \prod_{i=1}^{\ell} \phi_i$, where $\phi_1, \ldots, \phi_\ell \in \mathcal{P}(n, s, d)$ are all (not necessarily different) irreducible, returns the ϕ_i-s. Moreover, when the assumption on the field is removed, there is an algorithm that solves this problem in $\text{poly}(n, (d^2)!, s^{d \log \ell + d^3}, \ell^d)$-time.*

This theorem will follow as a consequence of the next, more general, result.

Theorem 1.11 (Factoring products of factors of sparse polynomials). *Let $\mathbb{F}$ be a field with* $\mathrm{char}(\mathbb{F}) = 0$ *or* $\mathrm{char}(\mathbb{F}) > 2d$*. There is a deterministic* $\mathrm{poly}(n, d^d, s^{d \log \ell}, \ell^d)$*-time algorithm which, given blackbox access to a product* f *of* ℓ *polynomials in* $\mathcal{P}_{Factors}(n, s, d)$ *over the field* $\mathbb{F}$*, returns all the* s*-sparse divisors of bounded individual degree* d *of* f *that are not divisible by any monomial, as well as the multiplicity* k_i *of each* x_i *as a factor of* f*. Moreover, if* f *was in fact a product of polynomials in* $\mathcal{P}(n, s, d)$*, there is such an algorithm that works over arbitrary fields, whose running time is* $\mathrm{poly}(n, (d^2)!, s^{d \log \ell + d^3}, \ell^d)$*.*

Observe that this result is slightly weaker for fields of small characteristic. Indeed, in Section 4 we will see that some of our methods only work for zero characteristic fields, or for fields with large enough characteristic.

In addition to our main results we provide several improvement and generalization of earlier work. We start by stating results concerning factorization of sparse polynomials of bounded individual degree into irreducible factors. The first is an improvement of the best known factorization algorithm of such polynomials [BSV20].

Theorem 1.12. *There is a deterministic* $\mathrm{poly}(n, s^{d^2 \log n})$*-time algorithm which, given an* s*-sparse polynomial of bounded individual degree* d *on* n*-variables, returns its factorization into irreducible factors.*

The second theorem generalizes the result of [BSV20] to a significantly larger class of polynomials, while achieving improved running time compared to their algorithm.

Theorem 1.13. *Let* $\mathbb{F}$ *be a field with* $\mathrm{char}(\mathbb{F}) = 0$ *or* $\mathrm{char}(\mathbb{F}) > 2d$*. There is a deterministic* $\mathrm{poly}(n, s^{d^3 \log n}, (d\ell)^d)$*-time algorithm which, given blackbox access to a product* f *of* ℓ *polynomials in* $\mathcal{P}_{Factors}(n, s, d)$ *over* $\mathbb{F}$*, returns its factorization into irreducible factors. Moreover, there is an algorithm that works over arbitrary fields, whose running time is* $\mathrm{poly}(n, (d^2)!, s^{d^5 \log n}, \ell^d)$*.*

Note that this last result considers the class of *factors of sparse polynomials of bounded individual degree*. Indeed, we will see that it is more natural to consider this class in many cases, and several results of ours actually hold for this class rather than just for the class of sparse polynomials.

Finally, we generalize results of [Vol15, Vol17, BV25] in a similar manner to our generalization of the results of [BSV20].

Our next algorithm finds all the multiquadratic irreducible factors of a product of polynomials from $\mathcal{P}_{\mathrm{Factors}}(n, s, d)$. Earlier, only factorization of polynomials whose irreducible factors are all multiquadratic and s-sparse was known [Vol17, Theorem 47].

Theorem 1.14. *Let* $\mathbb{F}$ *be a field of characteristic zero or larger than* $2d$*. There is a deterministic* $\mathrm{poly}(n, d!, s^d, \ell)$*-time algorithm that, given blackbox access to a product* f *of* ℓ *elements of* $\mathcal{P}_{Factors}(n, s, d)$ *defined over* $\mathbb{F}$*, returns all the* s*-sparse multiquadratic irreducible factors of* f*. In particular, the algorithm returns all the multilinear factors of* f *(as these are* s*-sparse by Lemma 2.12). Moreover, there exists such an algorithm that works over arbitrary fields, whose running time is* $\mathrm{poly}(n, (d^2)!, s^{d^2}, \ell)$*.*

We give a similar improvement to the perfect power testing of [BV25].

Theorem 1.15. *Let* $\mathbb{F}$ *be a field of characteristic zero or larger than* $2d$*. There is a deterministic* $\mathrm{poly}(n, d!, s^d, \ell)$*-time algorithm that, given blackbox access to a polynomial* f *that is a product of* ℓ *polynomials in* $\mathcal{P}_{Factors}(n, s, d)$ *(defined over* $\mathbb{F}$*) and a positive integer* e*, returns if* f *is a complete* e*-th power.*

Finally, we extend the polynomial identity testing result of [BV25].

Theorem 1.16. *Let $\mathbb{F}$ be a field of characteristic zero or larger than $2d$. There exists a deterministic $\text{poly}(n, d!, s^d, \ell)$-time algorithm that, given blackbox access to two polynomials (over $\mathbb{F}$) f and g that are products of ℓ polynomials in $\mathcal{P}_{Factors}(n, s, d)$, decides if $f|g$. Moreover, if f and g are actually products of ℓ polynomials in $\mathcal{P}(n, s, d)$, there is an algorithm that works over arbitrary fields, whose running time is* $\text{poly}(n, (d^2)!, s^{d^3}, \ell)$

As mentioned in Section 1.1, we will have to solve some rational interpolation problems. Somewhat surprisingly, it is not known how to deterministically reconstruct two coprime s-sparse polynomials f and g from blackbox access to their quotient g/f in polynomial time. We shall see that with some additional information on f, this is in fact possible. Without this extra information, only exponential deterministic algorithms are known for solving this problem [GKS94]. We note that efficient randomized algorithms for solving this problem are known, see e.g., [CL11, vdHL21].

1.3 Related Work

The questions considered in this work have been studied extensively in recent years. We organize the prior results around three themes that are central to this paper: sparsity bounds, factorization and reconstruction algorithms, and divisibility and identity testing.

Sparsity of Factors of Sparse Polynomials. Volkovich [Vol15] showed that multilinear factors of s-sparse polynomials are again s-sparse. In [Vol17] he proved that factors of an s-sparse multiquadratic polynomial are s-sparse as well. For general polynomials of bounded individual degree d, Bhargava, Saraf and Volkovich [BSV20], obtained the first non-trivial bound on the sparsity of factors of s-sparse polynomials with bounded individual degree d, namely $s^{O(d^2 \log n)}$. Bisht and Saxena [BS25] improved this bound to $s^{\text{poly}(d)}$ for the class of *symmetric* s-sparse polynomials of bounded individual degree d. Bisht and Volkovich [BV25] obtained an $s^{d \log s}$ bound on the sparsity of a quotient of an s-sparse polynomial of bounded individual degree d by a multilinear factor. Finally, the work of Forbes [For15] and results in [BV25] imply that the sparsity of the quotient of $f \in \mathcal{P}(n, s, d)$ by a degree r polynomial is bounded by $(ns)^{O(dr)}$.

Deterministic factorization and reconstruction. Factorization results have largely paralleled the development of sparsity bounds. Volkovich showed in [Vol15] how to efficiently factorize a sparse polynomial whose factors are all multilinear. In [Vol17] he demonstrated how to efficiently compute the irreducible factors of a polynomial from a blackbox access, given that all of them are multiquadratic and sparse. In Theorem 1.14 we further improve this result by relaxing the condition on the input polynomial - we allow it to be a product of factors of some sparse polynomial of bounded individual degree, and not simply a product of sparse multiquadratic polynomials.

Using their sparsity bound, Bhargava, Saraf, and Volkovich [BSV20] derived a deterministic quasi-polynomial algorithm for factoring sparse polynomials of bounded individual degree. However, their methods do not yield a polynomial-time algorithm for finding the sparse divisors of the input polynomial, so they do not imply Theorem 1.9. Furthermore, their algorithm does not extend to factorization of products of sparse, bounded degree polynomials, let alone products of factors of such polynomials.

Kumar, Ramanathan and Saptharishi [KRS24] and Dutta, Sinhababu, and Thierauf [DST24], gave an algorithm that outputs all the irreducible factors of constant degree of a sparse polynomial in quasi-polynomial time. As their result applies to general sparse polynomials without restrictions on the individual degree, it does not follow from the methods of this paper.

In [BV25], an algorithm for deciding if a sparse polynomial of bounded individual degree is a complete power is given; in Section 4.2 we use our stronger methods to obtain this result when the input polynomial is a product of polynomials in $\mathcal{P}_{\text{Factors}}(n, s, d)$, and we have blackbox access to it. However, this generalization does not work when the underlying field is of small characteristic.

Divisibility and identity testing. In [Vol17] it is also shown how to efficiently check if two s-sparse polynomials of bounded individual degree d divide each other; this is improved in Theorem 1.16, where we show a divisibility testing result for two products of factors of sparse polynomials of bounded individual degree, given blackbox access to them. In the same work, Volkovich also gave a deterministic algorithm for testing a factorization of sparse polynomials, with constant individual degrees, into sparse irreducible factors. That is, testing if $f = \prod g_i$ when f has constant individual degrees and g_i-s are irreducible and sparse. We note that by Corollary 8.6 we can efficiently find the g_i's and verify that $f = \prod g_i$.

Forbes [For15] gave a quasi-polynomial-time algorithm for checking if a constant degree polynomial divides a sparse polynomial. Note that while our result allows the dividing polynomial to be much more complicated, Forbes' result requires no restriction on the individual degree of the given sparse polynomial. Therefore, our methods fall short in proving his result, or any other result on sparse polynomials with no restriction on their individual degree.

In [BV25], the authors design an efficient blackbox PIT algorithm for the class $\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d\,]}$; while doing so, they obtain some technical results about the class $\mathcal{P}_{\text{Factors}}(n, s, d)$. In Section 4.1 we show that their methods imply a stronger result than this PIT. Showing PIT for this class is essentially equivalent to giving an algorithm for determining if two products f, g of (n, s, d)-sparse polynomials are equal, given blackbox access to them. We show that the methods of [BV25] can be used for determining if f divides g. In Section 4.2 we further generalize and strengthen the technical results of [BV25], but only for fields of zero or sufficiently large characteristic.

1.4 Proof Overview

We start by describing some technical results about the class of sparse polynomials of bounded individual degree needed for our proofs. We then turn to give an overview of our results regarding factorization of sparse polynomials. We first discuss our result on the recovery of all irreducible sparse multiquadratic factors of a product of (n, s, d)-sparse polynomials, as the proof contains some of the important ideas used in the proofs of our main results. We then move on to describe the recovery of all sparse divisors of sparse polynomials of bounded individual degree and our partial solution to Question 1.4. Finally, we briefly describe how to use our methods for factoring general (n, s, d)-sparse polynomials and products of them, making use of the bound of [BSV20].

1.4.1 Sparse Polynomials of Bounded Individual Degree and Their Factors

The main tool we use for obtaining our results is a generator $G : \mathbb{F}^6 \to \mathbb{F}^n$ that preserves important properties of polynomials in $\mathcal{P}_{\text{Factors}}(n, s, d)$. Roughly speaking, we want this map G to satisfy the following two properties:

(1) Its image $\text{Im}(G)$ is an interpolation set for polynomials in $\mathcal{P}(n, s, d)$.

(2) If ϕ and ψ are non-associate[1] irreducible elements of $\mathcal{P}_{\text{Factors}}(n, s, d)$, then $\phi(G_i)$ and $\psi(G_i)$ are coprime as polynomials in x_i, where G_i should be thought of as "G when reviving the i-th variable" (i.e., keeping x_i alive and substituting the remaining variables according to G).

[1]Two polynomials are associate if one is a scalar product of the other.

The first property is easily achieved by the basic properties of the Klivans-Speilman generator (KS-generator), introduced in [KS01]. We show that the second property follows from the first, together with the fact that resultants of sparse polynomials of bounded individual degree are again sparse. However, we are able to do so only when the underlying field $\mathbb{F}$ is of zero characteristic, or if its characteristic is larger than twice the bound on the individual degree. If this is not the case then we are only able to prove something a little weaker - we prove that the second property holds if one replaces the notion of irreducible polynomials with the new notion of *primitive divisors*, which we first define in this paper (we describe this notion below).

Before sketching the proof that the first property of G implies the second, we note that proving that will automatically imply some algorithms for sparse polynomials. For example, it is straightforward to see that by factoring the composition of the input polynomials with G_i, the following two results can be obtained from the (general) second property of G.

(1) Given blackbox access to two products of elements of $\mathcal{P}_{\text{Factors}}(n, s, d)$, we can decide if one divides the other in deterministic polynomial time (Theorem 1.16).

(2) Given blackbox access to a product of elements of $\mathcal{P}_{\text{Factors}}(n, s, d)$, we can decide if it is a complete power in deterministic polynomial time (Theorem 1.15).

This already generalizes results from [Vol17] and [BV25]. As we are able to obtain the general second property of G only for some fields, these results do not apply to arbitrary fields. It turns out, however, that the aforementioned weaker version of the second property of G is strong enough to obtain Theorem 1.16 over arbitrary fields, in the case we replace $\mathcal{P}_{\text{Factors}}(n, s, d)$ by $\mathcal{P}(n, s, d)$ (see Theorem 4.14).

In the rest of this subsection we describe our proofs of the second property, both for arbitrary fields and for fields with zero or large enough characteristic.

Arbitrary Fields. In [BV25, Lemmas 4.5,4.6] it is shown that the second property of G holds for each pair of irreducible polynomials ϕ and ψ that satisfy a certain non-degeneracy condition: there exist (n, s, d)-sparse polynomials f, g for which $\phi|f, \psi|g$ and the matrix $\begin{pmatrix} \nu_\phi(f) & \nu_\phi(g) \\ \nu_\psi(f) & \nu_\psi(g) \end{pmatrix}$ is invertible, where $\nu_\phi(f)$ is the multiplicity of f as a factor of ϕ, and so are the other entries, respectively. That is, a pair of polynomials $\phi, \psi \in \mathcal{P}_{\text{Factors}}(n, s, d)$ does not satisfy this condition if the multiplicity of ϕ as a factor of an element f of $\mathcal{P}(n, s, d)$ determines the multiplicity of ψ as a factor of f. In this sense, ϕ and ψ are ”inseparable” from the point of view of polynomials in $\mathcal{P}(n, s, d)$.

For each irreducible $\phi \in \mathcal{P}_{\text{Factors}}(n, s, d)$, consider all ψs that are inseparable from it, in the sense that ϕ, ψ do not satisfy the aforementioned non-degeneracy condition. For each $f \in \mathcal{P}(n, s, d)$, consider the vector of multiplicities of these ψs as factors of f - $(\nu_\psi(f))_\psi$. By the degeneracy assumption, these vectors are all proportional. Consider the $\mathbb{Z}$-span of these vectors, and let $(a_\psi)_\psi$ be a generator for this one dimensional lattice. The polynomial $\prod_\psi \psi^{a_\psi}$ is called a primitive divisor for the class $\mathcal{P}(n, s, d)$. Note that by construction, as the property of being ”inseparable” is an equivalence relation on the class of irreducible polynomials in $\mathcal{P}_{\text{Factors}}(n, s, d)$, it must hold that every two primitive divisors are coprime.

The notion of a primitive divisor is a general notion, that applies to a general class of polynomials $\mathcal{P}$, and therefore may be of independent interest; a formal definition of it is given in Definition 4.1. Primitive divisors for a class of polynomials $\mathcal{P}$ are for the class $\mathcal{P}$ what irreducible polynomials are for the class of all polynomials, in the sense that every element of the class $\mathcal{P}$ can be written as a product of primitive divisors for $\mathcal{P}$ in a unique way; this is not too difficult to see by the construction given in the previous paragraph, and is formally done in Lemma 4.6. Therefore, it makes sense to talk about the multiplicity $\nu_\phi(f)$ of a primitive divisor ϕ as a factor of $f \in \mathcal{P}$.

With this definition in hand, one can show that every two non-associate primitive divisors ϕ and ψ for a class $\mathcal{P}$ satisfy the non-degeneracy condition of [BV25]. As a result, we are able to apply the proof of the aforementioned result of [BV25] to obtain the weaker version of the second property of G, namely, for every two non-associate primitive divisors ϕ, ψ for $\mathcal{P}(n, s, d)$ it holds that $\gcd_{x_i}(\phi(G_i), \psi(G_i)) = 1$.

We further note that for every two products f, g of elements of $\mathcal{P}(n, s, d)$ we have that $f|g$ if and only if $\nu_\phi(f) \leqslant \nu_\phi(g)$ for every primitive divisor ϕ for $\mathcal{P}(n, s, d)$; this is a simple corollary of our unique-factorization theorem. As a result, the weaker second property of G is enough for giving a divisibility testing for f and g: just check if $f(G_i)|g(G_i)$ as polynomials in x_i for every i. This is formally done in Theorem 4.14.

Zero or Large Characteristic. In the proof of [BV25, Lemma 4.5], Bisht and Volkovich consider the Sylvester matrix $M_{x_i}(f^\alpha, g^\beta)$ of the some suitably chosen small powers of the polynomials f, g, viewed as polynomials in x_i. They then use the first property of G to prove that the rank of this matrix does not change when composing with the generator G_i, that is, $\operatorname{rank}(M_{x_i}(f^\alpha, g^\beta)) = \operatorname{rank}(M_{x_i}(f(G_i)^\alpha, g(G_i)^\beta))$. From that, they deduce that it must be the case that $\phi(G_i)$ and $\psi(G_i)$ do not share factors, as the existence of such a factor would result in rank decrease.

To obtain our result, instead of considering the Sylvester matrix $M_{x_i}(f^\alpha, g^\beta)$, we consider the matrix defining the discriminant $\Delta(fg)$ of the polynomial fg. That is, we consider the Sylvester matrix $M_{x_i}(fg, (fg)')$ of fg and its formal derivative with respect to x_i, $(fg)'$. We next explain the idea behind it.

Suppose $\phi|f, \psi|g$ are two irreducible elements of $\mathcal{P}_{\text{Factors}}(n, s, d)$. View all polynomials as polynomials in some variable x_i, and consider the Sylvester matrix $M_{x_i}(fg, (fg)')$. It is not hard to prove that its rank essentially counts the number of unique irreducible factors of fg (when appropriately counted; for the formal statement, see Claim 4.16). As all the determinants of minors of $M_{x_i}(fg, (fg)')$ are sparse polynomials of low degree, we are able to show, using the first property of G, that the rank of this matrix does not change when composing with G_i. In particular, this means that $\phi(G_i), \psi(G_i)$ cannot share a common factor, as otherwise the number of unique factors of $fg(G_i)$ would be smaller than that of fg, in contradiction to the fact that the rank remains unchanged. With this result in hand, we obtain Theorem 1.16 and Theorem 1.15 by considering the factorization of the inputs composed with the generators G_i.

1.4.2 Finding Multiquadratic Factors of Products of Sparse Polynomials

We next show how to use the results of Section 1.4.1 to obtain Theorem 1.14. For the sake of presentation we assume here that $\operatorname{char}(\mathbb{F}) = 0$.

We first consider the following general setting. Let f be a product of polynomials in $\mathcal{P}_{\text{Factors}}(n, s, d)$; that is, f is a product of factors of s-sparse polynomials of bounded individual degree d. Consider some irreducible $\phi \in \mathcal{P}_{\text{Factors}}(n, s, d)$. By the second property of G it follows that for every i for which $x_i \in \operatorname{var}(\phi)$, the highest power of $\phi(G_i)$ that divides $f(G_i)$ is precisely the multiplicity of ϕ as a factor of f. In particular, this is true whenever ϕ is an irreducible s-sparse polynomial of bounded individual degree d. With this observation in mind, we are now ready to describe the algorithm for finding multilinear factors. Suppose $f = \prod_{i=1}^{\ell} \phi_i^{e_i} \in \mathbb{F}[x_0, \mathbf{x}]$ where ϕ_i are factors of $(n+1, s, d)$-sparse polynomials. Suppose further that we have blackbox access to

f, and we want to recover all of its irreducible multilinear factors. [2] Our algorithm follows the following steps:

Step 1 - Normalization and Recursion. In this step, we would like to replace our input polynomial f with another polynomial $\tilde{f}$ that still captures all the information about the factors of f, but is normalized (in the sense that its free term is 1 as a polynomial in x_0). To do that, we use a standard method, employed for example in [BV25, BSV20]. Let $f = \sum_{i=0}^{\ell d} f_i x_0^i$ where $f_i \in \mathbb{F}[\mathbf{x}]$ and let $\tilde{f}(y, x_1, \ldots, x_n) = f(f_0 y, x_1, \ldots, x_n)/f_0$.[3] Note that as we have blackbox access to f, we also have blackbox access to f_0 and therefore to $\tilde{f}$. It is straightforward to see that each multilinear factor $f_i(\mathbf{x}) = a_i(\mathbf{x})x_0 + b_i(\mathbf{x})$ of f now corresponds to a multilinear factor $\frac{a_i f_0}{b_i} y + 1$ of $\tilde{f}$. Thus, one sees that:

(1) If $a_i \neq 0$, then all the information about a_i and b_i is captured in $\tilde{f}$.

(2) If $a_i = 0$, then the factor b_i cannot be studied from $\tilde{f}$. However, note that in this case $b_i | f_0$, and therefore it can be studied from a factorization of f_0.

Note that f_0 has the same structure of f - it is itself a product of factors of s-sparse polynomials of individual degree d. Therefore, by applying one recursive call, we can find all irreducible multilinear factors of f_0. Thus, to understand the factors with $a_i = 0$, we just need to understand, for any multilinear irreducible factor ϕ of f_0, how many times it divides f. We will show a way to do so in the next step. For now, the main benefit from this step is the normalization $\tilde{f}$ and the calculation of the multilinear factors ϕ_i of f_0.

Step 2 - Obtaining the factorization. As we saw in the first step, there are two types of factors - those with $a_i = 0$ and those with $a_i \neq 0$. Thus, we split the algorithm into two parts:

(1) Recovering the factors for which $a_i = 0$. In this case, it holds that $b_i | f_0$. Therefore, as we already saw, it suffices to determine which irreducible factors of f_0 divide f (and to what power). To that end, note that for a given sparse irreducible factor ϕ of f_0, it follows from the second property of G and from the fact that f is a product of elements of $\mathcal{P}_{\text{Factors}}(n+1, s, d)$, that to calculate the multiplicity of ϕ as a factor of f it suffices to check how many times $\phi(G_j)$ divides $f(G_j)$, where $x_j \in \text{var}(\phi)$. This can be easily done, for example, by factoring both polynomials, which is possible as they have a constant number of variables and polynomial degree.

(2) Recovering the factors for which $a_i \neq 0$. In this case, using the first step, we can get access to $\frac{a_i(G) f_0(G)}{b_i(G)}$, by factoring $\tilde{f}(y, G)$. Recovering a_i from such an expression is the rational interpolation problem that we earlier alluded to. Here we have the additional information that b_i is a multilinear factor of f_0. Thus, to find b_i, it suffices to understand, for each multilinear irreducible factor ϕ of f_0, what is the maximal power of ϕ that divides b_i. For any such factor ϕ, using the fact that a_i and b_i are coprime (as otherwise $a_i x_0 + b_i$ will not be irreducible), we have that $\phi^k | b_i$ if and only if $\phi^{t-k+1} \nmid \frac{a_i f_0}{b_i}$, where t is the multiplicity of ϕ as a factor of f_0. It follows that we just have to understand the multiplicity of ϕ as a factor of $\frac{a_i f_0}{b_i}$. Since a_i is sparse and f_0 is a product of polynomials in $\mathcal{P}_{\text{Factors}}(n, s, d)$, we have that $\frac{a_i f_0}{b_i}$ is a product of polynomials in $\mathcal{P}_{\text{Factors}}(n, s, d)$ as well, so we can finish by computing the multiplicity of

[2]The general algorithm finds the multiquadratic, sparse, irreducible factors of the input. To keep the presentation here as simple as possible, we consider only the multilinear factors.

[3]It is easy to reduce to the case $f_0 \neq 0$, so to keep the presentation as simple as possible we assume it to be the case.

$\phi(G_j)$ as a factor of $\frac{a_i(G_j)f_0(G_j)}{b_i(G_j)}$ (where $x_j \in \mathrm{var}(\phi)$). This recovers b_i and subsequently a_i as well, completing the algorithm.

In the above, we actually solved a rational interpolation problem. That is, we had access to $\frac{af}{b}$, where $b|f$, and needed to recover a and b from this information. We claim that in fact, the algorithm described above works in greater generality. The exact formulation of this generalization and its proof are given in Theorem 5.1 and Theorem 5.2.

We conclude with the following remarks. First, we note that it is possible to modify this algorithm to recover all the multiquadratic irreducible factors of f, rather than just the multilinear ones. We further note that the algorithm described above still works when the assumption on the field $\mathbb{F}$ is removed, albeit with a slightly worse running time. These claims are proven in full detail in Section 7.

1.4.3 Finding Sparse Divisors of Sparse Polynomials

We now describe the ideas behind our main result (Theorem 1.9). Recall that our goal is to output all the sparse divisors of a given (n, s, d)-sparse polynomial f. That is, we wish to efficiently find those divisors of f that can be "written down" efficiently.

To obtain such an algorithm, a natural thing to do is to follow the lines of the algorithm presented in Section 1.4.2. The first step of this algorithm is performing a normalization and then applying a recursive call for factoring the free term f_0. But now, a problem arises - this recursive call only returns the irreducible sparse factors of f_0, while the free term of each factor of f (b_i in the language of Section 1.4.2) might be a *reducible* sparse divisor of f_0 whose irreducible factors are not sparse; in this case, the recursive call did not provide us with any valuable information about b_i. In particular, we cannot reconstruct b_i by studying how many times each sparse irreducible factor of f_0 divides it as we did in Section 1.4.2. In fact, the true reason the algorithm presented in Section 1.4.2 works is that in that context, b_i is sparse multilinear (or sparse multiquadratic in the more general case), and therefore so are its irreducible factors; hence, we indeed get information about b_i from the recursive call. In other words, the free terms there belong to a class of sparse polynomials that is closed under factorization.

This motivates considering the problem of finding all the *sparse divisors* of a sparse polynomial. This way, we will still have information about the free terms of the factors of f - the recursive call will actually compute them. Indeed, as sparse divisors of sparse divisors of (n, s, d)-sparse polynomials are themselves sparse divisors of (n, s, d)-sparse polynomials, so this class is closed under factorization. This enables us to recover all these sparse divisors in deterministic $\mathrm{poly}(n, s^{\mathrm{poly}(d)})$-time. Indeed, the recursive call returns all the sparse divisors of the free term f_0 rather than just the set of irreducible factors of it, so we can directly brute-force over the list of these factors to find b_i. This yields Theorem 1.9.

At first glance, this approach may seem surprising, as the existence of such an algorithm suggests that the possible number of outputs is bounded above by $\mathrm{poly}(n, s^{\mathrm{poly}(d)})$. Indeed, Theorem 1.8 ensures that s-sparse polynomials of bounded individual degree d have no more than $d \log s$ irreducible factors (that are not monomials), and therefore, no more than s^d divisors (not divisible by a monomial) overall. To prove Theorem 1.8, we show that if $f \in \mathcal{P}(n, s, d)$ is not divisible by any monomial, then one can identify a set of $\log s$ variables that satisfy the property that every irreducible factor of f depends on at least one of them. From this claim it easily follows that f has at most $d \log s$ irreducible factors, and therefore at most $2^{d \log s} = s^d$ divisors overall. To prove that such a set of variables exist, we analyze the Newton polytope of f, and show that if the minimal set of variables satisfying the property is of size k, then the Newton polytope of f has at least 2^k vertices. This is somehow intuitive, as each additional variable in the set effectively adds a dimension to the polytope and therefore doubles the number of its vertices.

We note that Theorem 1.9 is not implied by the methods of [BSV20]; indeed, there the bound on the sparsity of *all irreducible factors* of f is needed to hit all resultants of different factors; therefore, applying their methods to this problem must result in a running time of at least $s^{O(d^2 \log n)}$, which is their bound on the sparsity of the factors of f.

One main drawback of our algorithm is that it does not tell us what are the *irreducible* sparse divisors of the input polynomial, as we are not familiar with a way of efficiently identifying the irreducible polynomials from the outputs we get. On the other hand, if we are guaranteed that f is a product of irreducible sparse factors then we can easily find the irreducible factors among all the sparse divisors. This establishes Corollary 8.6, thereby solving Question 1.3 for the special case of (n, s, d)-sparse polynomials.

These ideas are also used in the proof of Theorem 1.10. Recall that Question 1.4 asks for a deterministic, polynomial-time algorithm that given blackbox access to a product of (n, s, d)-sparse polynomials returns the factors. While we do not know a polynomial-time algorithm when the number of terms is unbounded, we show an algorithm that runs in quasi-polynomial time in their number, making the first step towards a resolution of the problem. The algorithm is the natural, straightforward generalization of the algorithm described above. Unfortunately, it requires quasi-polynomial time as we actually find all the s-sparse divisors of bounded individual degree d of f, and there are (possibly) quasi-polynomially many of them (as shown by Corollary 8.3).

1.4.4 Factorization for Sparse Polynomials

Lastly, we discuss our algorithms for factoring (n, s, d)-sparse polynomials and their products. These algorithms are essentially the same as the algorithms we already presented; the only difference is that now, when dealing with elements of $\mathcal{P}_{\text{Factors}}(n, s, d)$, we view them as elements of $\mathcal{P}(n, s^{O(d^2 \log n)}, d)$.

Consider for example our algorithm for finding all the sparse divisors of an (n, s, d)-sparse polynomial. With a slight modification, we can make it output all the $(n, s^{O(d^2 \log n)}, d)$-sparse divisors of the input polynomial in $\text{poly}(n, s^{d^2 \log n})$-time. By [BSV20], these are all the possible divisors. Identifying the irreducible factors among these can be easily done. For example, we can check for any two divisors ϕ and ψ if $\psi | \phi$ by trying to sparse-interpolate their quotient - which should be an $s^{O(d^2 \log n)}$-sparse polynomial.

As for our algorithm for factoring products of $\mathcal{P}_{\text{Factors}}(n, s, d)$ elements, it is identical to the algorithm described in Section 1.4.2, now taking into account that all irreducible factors of interest are $s^{O(d^2 \log n)}$-sparse, rather than s-sparse and multiquadratic.

1.5 Organization

In Section 2 we give all the background needed for this paper. In Section 3 we construct the generator G, which is the main technical tool of this paper. In Section 4 we prove the basic properties of G we need, both for arbitrary fields and when the characteristic is zero or large enough, and apply them to obtain our divisibility testing and exact power testing results (Theorem 1.16, Theorem 1.15). In Section 5 we prove our rational interpolation result.

We give a *meta-algorithm*, according to which all of our algorithms are modeled, in Section 6. We then demonstrate how to apply it to recover the multiquadratic sparse irreducible factors of a product of polynomials in $\mathcal{P}_{\text{Factors}}(n, s, d)$ (Theorem 1.14) in Section 7. In Section 8 we bound the number of divisors of (n, s, d)-sparse polynomials (Theorem 1.8), and use it to find the all the sparse divisors of such polynomials (Theorem 1.9), as well as to give one partial solution to Question 1.4 (Theorem 1.10). In Section 9 we give our algorithms for factoring (n, s, d)-sparse polynomials and products of them, based on the bound of [BSV20] (Theorem 1.12,Theorem 1.13). The second result gives another partial solution to Question 1.4. We conclude with a short discussion in Section 10. In Section A we give different, yet weaker, proofs for some technical claims from Section 4. Since the proof technique is different, we decided to give the proof in spite of the results being slightly weaker.

2 Preliminaries

We use $\mathbf{x}$ to denote the n-variate vector of variables, $\mathbf{x} = (x_1, \ldots, x_n)$. In general, bold-face letters such as $\mathbf{x}, \mathbf{e}$ will be used to denote n-tuples, where the domain of the n-tuple will always be clear from the context.

We say that two nonzero polynomials f, g are associate if one is a scalar product of the other. We also state the following special case of Gauss' lemma.

Lemma 2.1 (Gauss's Lemma). *A polynomial $f(x_0, \mathbf{x}) = \sum_{i=0}^{d} c_i(\mathbf{x})x_0^i \in \mathbb{F}[\mathbf{x}][x_0] \setminus \mathbb{F}[\mathbf{x}]$ is irreducible if and only if it is both irreducible in $\mathbb{F}(\mathbf{x})[x_0]$ and $\gcd(c_0, \ldots, c_d) = 1$.*

2.1 Reverse Monic Polynomials and Normalization

Throughout the paper we will use the following definition.

Definition 2.2. A polynomial $f \in \mathbb{F}[x_0, \mathbf{x}]$ is said to be reversed-monic with respect to x_0 if its free term as a polynomial in $\mathbb{F}(\mathbf{x})[x_0]$ is 1. Equivalently, f is reversed-monic with respect to x_0 if it is of the form $f = 1 + \sum_{i \geqslant 1} f_i x_0^i$ where $f_i \in \mathbb{F}[\mathbf{x}]$.

We prove the following simple claim about reversed-monic polynomials.

Claim 2.3. *Let $f \in \mathbb{F}[x_0, \mathbf{x}]$ be a reversed-monic polynomial with respect to x_0. Then, the number of irreducible factors of f is at most $\deg_{x_0}(f)$.*

Proof. We first note that all irreducible factors of f must depend on x_0. Indeed, write $f = \sum_{i=0}^{\deg_{x_0}(f)} f_i x_0^i$ where $f_i \in \mathbb{F}[\mathbf{x}]$. By definition of reversed-monic polynomials, $f_0 = 1$. Clearly, any divisor of f that does not depend on x_0 must divide f_0, and hence no such non-trivial divisors exist. The claim follows, since there are at most $\deg_{x_0}(f)$ irreducible factors of f depending on x_0. □

We next define a *normalization* operation that turns a polynomial into a reverse monic one.

Definition 2.4 (Normalization). Let $f(x_0, \mathbf{x}) = \sum_{i=0}^{d} f_i(\mathbf{x})x_0^i \in \mathbb{F}[x_0, \mathbf{x}]$, such that $x_0 \nmid f$. The normalization of f with respect to x_0 is the reverse monic polynomial

$$\tilde{f}(y, \mathbf{x}) = f(yf_0, x_1, \ldots, x_n)/f_0 = 1 + \sum_{i=1}^{d} f_i(\mathbf{x}) f_0^{i-1}(\mathbf{x}) y^i. \tag{1}$$

Normalization is a standard and well-known technique, and different versions of our next claim can be also found in [BSV20, BV25]. Our version of the claim describes the structure of the factorization of $\tilde{f}$, which will play an important role in our proofs and reconstruction algorithms.

Claim 2.5. *Let* f *as in Definition 2.4. Suppose* $f = \prod_{i=0}^{N} h_i \in \mathbb{F}[x_0, \mathbf{x}]$ *is the factorization of* f *into irreducible factors* h_i*, and denote each factor by* $h_i = \sum_{j=0}^{d_i} c_{i,j} {x_0}^j$ *(where* $c_{i,j} \in \mathbb{F}[\mathbf{x}]$*). Then, the factorization of* $\tilde{f}$ *is given by*

$$\tilde{f} = \prod_{i=0}^{N} \tilde{h}_i \text{ , where } \quad \tilde{h}_i = \sum_{j=0}^{d_i} \frac{c_{i,j}}{c_{i,0}} {f_0}^j y^j \ .$$

Moreover, every factor (not necessarily irreducible) of f *of the form* $\sum_{j=0}^{d} \zeta_j {x_0}^j$ *(where* $\zeta_j \in \mathbb{F}[\mathbf{x}]$*) corresponds to a factor of* $\tilde{f}$ *of the form* $\sum_{j=0}^{d} \frac{\zeta_j}{\zeta_0} {f_0}^j y^j$*. In particular, every multiquadratic factor of* f *of the form* $a_2 x_0^2 + a_1 x_0 + b$ *corresponds to a factor of* $\tilde{f}$ *of the form* $1 + \frac{a_1 f_0}{b} y + \frac{a_2 f_0^2}{b} y^2$*. In addition, we have blackbox access to* $\tilde{f}$ *from blackbox access to* f*.*

Proof. First, note that $f_0 = \prod_{i=0}^{N} c_{i,0}$. Next, let

$$\tilde{f}(y, \mathbf{x}) = f(y f_0, \mathbf{x})/f_0 = \prod_{i=0}^{N} \frac{h_i(y f_0, \mathbf{x})}{c_{i,0}} \ .$$

Define

$$\tilde{h}_i(y, \mathbf{x}) := \frac{h_i(y f_0, \mathbf{x})}{c_{i,0}} = \sum_{j=0}^{d_i} \frac{c_{i,j} (y f_0)^j}{c_{i,0}} = \sum_{j=0}^{d_i} \frac{c_{i,j}}{c_{i,0}} {f_0}^j y^j \in \mathbb{F}[y, \mathbf{x}] \ .$$

We claim that $\tilde{h}_i$ is irreducible. If $d_i = 0$, then $\tilde{h}_i = 1$, so this is clear. Assume $d_i > 0$. By definition, h_i is an irreducible polynomial as an element of $\mathbb{F}[x_0, \mathbf{x}]$, and therefore it is also irreducible as an element of $\mathbb{F}(\mathbf{x})[x_0]$. In this ring it holds that $\tilde{h}_i(x_0) = h_i(f_0 x_0)/c_{i,0}$. In particular, $\tilde{h}_i$ differs from h_i by a linear change of variables and a multiplication by scalar (in the field $\mathbb{F}(\mathbf{x})$), so it is irreducible as well. Since the free term of $\tilde{h}_i$ is 1, it follows from Gauss' lemma (Lemma 2.1) that $\tilde{h}_i$ is irreducible as an element of $\mathbb{F}[x_0, \mathbf{x}]$, as we wanted.

The "moreover" part follows from the fact that the argument on the form of $\tilde{h}_i$ holds regardless of the irreducibility of h_i.

The claim regarding blackbox access to $\tilde{f}$ follows from simple interpolation: given an evaluation point $(y_0, \boldsymbol{\alpha})$ we interpolate the polynomial $f(x_0, \boldsymbol{\alpha})$ as a polynomial in x_0 to get access to each $f_i(\boldsymbol{\alpha})$. One can then obtain $\tilde{f}(y_0, \boldsymbol{\alpha})$ simply by using the right hand side of (1). In particular, every query to $\tilde{f}$ requires at most $d + 1$ queries to f. □

2.2 Factorization and Interpolation of Polynomials

We start by stating a result on factorization of univariate polynomials. The first is the classical algorithm of Lenstra, Lenstra and Lovasz [LLL82].

Theorem 2.6 (Univariate factorization over $\mathbb{Q}$)**.** *Let* $f \in \mathbb{Q}[x]$ *have degree* d *and bit complexity* B*. Then, there is a deterministic algorithm that outputs all irreducible factors of* f *whose running time is* $\text{poly}(d, B)$.

For the following theorem see e.g., [vzGS92, Section 9].

Theorem 2.7 (Univariate factorization over Finite Fields)**.** *Let* $\mathbb{F}_q$ *be a field of size* $q = p^k$ *and characteristic* $p > 0$*. Let* $f(x) \in \mathbb{F}_q[x]$ *be a polynomial of degree* d*. Then, there is a deterministic algorithm that outputs all irreducible factors of* f *in time* $\text{poly}(p, \log q, d)$.

We next state a theorem concerning deterministic multivariate factorization in the multivariate case. See e.g., [Len85, Kal85b]. These results determine the dependence on the field, its characteristic and the bit complexity of the input polynomial, in all our algorithms that invoke polynomial factorization.

Theorem 2.8. *There exists a deterministic algorithm that given an n-variate polynomial $f \in \mathbb{F}[\mathbf{x}]$ of total degree d, and bit complexity B, outputs its factorization into irreducible polynomials. Its running time is* $\text{poly}(d^n) \cdot T(d, B, \mathbb{F})$, *where* $T(d, B, \mathbb{F})$ *is the running time of univariate factorization of degree d polynomials, with bit complexity B, over $\mathbb{F}$.*

Next, we state the following folklore lemma on interpolation of polynomials on a small numbers of variables.

Lemma 2.9. *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be an n-variate polynomial of total degree d. Then, there is a deterministic polynomial-time algorithm that given blackbox access to f returns its monomial representation. Its running time is* $\text{poly}((d+1)^n)$.

Finally, we conclude this section by recalling the definition of Lagrange interpolation.

Definition 2.10. Let $\alpha_1, \ldots, \alpha_r \in \mathbb{F}$ be different field elements. Then the polynomials

$$A_i(x) = \prod_{1 \leqslant j \leqslant r, j \neq i} \frac{x - \alpha_j}{\alpha_i - \alpha_j} \tag{2}$$

satisfy $\deg(A_i) = r - 1$ and $A_i(\alpha_j) = \delta_{i,j}$, where $\delta_{i,j}$ is Kronecker's delta. The A_i-s are called the Lagrange interpolating polynomials of $\alpha_1, \ldots, \alpha_r$.

2.3 Sparse Polynomials

We give some definitions and state several known results concerning sparse polynomials.

Definition 2.11. An s-sparse polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ is a polynomial which is a sum of at most s monomials. It is said to be of bounded individual degree d if f is of degree at most d as an element of $\mathbb{F}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)[x_i]$ for every $1 \leqslant i \leqslant n$. We denote the class of n-variate, s-sparse polynomials of bounded individual degree d by $\mathcal{P}(n, s, d)$, and refer to its elements as (n, s, d)-sparse polynomials. Furthermore, we denote the class of factors of elements of $\mathcal{P}(n, s, d)$ by $\mathcal{P}_{\text{Factors}}(n, s, d)$. Note that by factors we mean all possible factors of (n, s, d)-sparse polynomials, not just the irreducible ones.

The next two lemmas give two special cases in which upper bounds on sparsity of factors of sparse polynomials is known.

Lemma 2.12 ([GKL12, Lemma 10])**.** *A multilinear factor of an s-sparse polynomial is again s-sparse.*

Lemma 2.13 ([Vol17, Lemma 32])**.** *Let g be an s-sparse multiquadratic polynomial (that is, g is of individual degree at most two) and suppose that $f | g$. Then f is also s-sparse.*

In [BSV20], Bhargava, Saraf and Volkovich upper bounded the sparsity of factors of (n, s, d)-sparse polynomials, and obtained the best known factorization algorithm for (n, s, d)-sparse polynomials. We state these results next.

Theorem 2.14 ([BSV20, Theorem 1])**.** $\mathcal{P}_{\mathit{Factors}}(n, s, d) \subset \mathcal{P}(n, s^{O(d^2 \log n)}, d)$. *In other words, every factor of an s-sparse polynomial of bounded individual degree d in n-variables is $s^{O(d^2 \log n)}$-sparse.*

In the next result, one should add a factor that depends on the field according to Theorem 2.8.

Theorem 2.15 ([BSV20, Theorem 2]). *There is a deterministic* $\mathrm{poly}(n, d^d, s^{d^7 \log n})$*-time algorithm, that given an* (n, s, d)*-sparse polynomial, returns its factorization into irreducible factors.*

To obtain Theorem 2.14, the authors study sparse polynomials via the study of their Newton polytopes. As we will use the language of Newton polytopes as well, we give the definition of Newton polytopes here.

Definition 2.16. The Newton polytope of a polynomial $f = \sum_e \gamma_e x^e$, denoted by P_f, is the convex hull of the points $e \in \mathbb{R}^n$. The set of vertices of the polytope P_f is denoted by $V(P_f)$.

The following well known fact can be found in [BSV20, Corollary 3.5].

Lemma 2.17. *Let* $f|g$ *be two polynomials, and suppose that* g *is* s*-sparse. Then,* $|V(f)| \leqslant |V(g)| \leqslant s$*.*

2.4 Resultants and Discriminants

We next discuss resultants and discriminants. See [GKZ08] for a detailed treatment.

Definition 2.18. Let $f, g \in \mathbb{F}[x]$ be two univariate polynomials of degrees d_1, d_2 respectively. Define $\phi : \mathbb{F}[x]_{\leqslant(d_1-1)} \times \mathbb{F}[x]_{\leqslant(d_2-1)} \to \mathbb{F}[x]_{\leqslant(d_1+d_2-1)}$ by

$$\phi(h_1, h_2) = h_1 g + h_2 f. \tag{3}$$

Here, $\mathbb{F}[x]_{\leqslant(d)}$ is the $(d+1)$-dimensional $\mathbb{F}$-vector space of univariate polynomials over $\mathbb{F}$ of degree at most d. This space has a standard monomial basis given by the monomials $1, x, \ldots, x^d$.

The Sylvester matrix $M(f, g)$ of f and g is the $(d_1 + d_2) \times (d_1 + d_2)$ matrix representing ϕ with respect to the standard monomial basis. The determinant of this matrix is called the resultant of f and g, and is denoted by $\mathrm{res}(f, g) \in \mathbb{F}$. If $f, g \in \mathbb{F}[x_0, x_1, \ldots, x_n]$ are two multivariate polynomials, we denote by $M_{x_0}(f, g)$ and $\mathrm{res}_{x_0}(f, g) \in \mathbb{F}[x_1, \ldots, x_n]$ the Sylvester matrix and resultant of f and g when viewed as elements of $\mathbb{F}(x_1, \ldots, x_n)[x_0]$.

Definition 2.19. Let $f \in \mathbb{F}[x]$ be a univariate polynomial. The discriminant of f is given by

$$\Delta(f) = \mathrm{res}(f, f'), \tag{4}$$

where f' is the standard formal derivative of f. For a multivariate polynomial $f \in \mathbb{F}[\mathbf{x}]$, denote by $\Delta_{x_i}(f) \in \mathbb{F}[x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n]$ the discriminant of f as an element of $\mathbb{F}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)[x_i]$.

Lemma 2.20. *Let* $\mathbb{F}$ *be a field.*

(1) Let $f, g \in \mathbb{F}[x]$ *be two monic polynomials. Then,* $\mathrm{res}(f, g) = \prod_{\alpha : f(\alpha)=0} g(\alpha)$*, where the product is over all the roots* α *of* f *over* $\overline{\mathbb{F}}$*.*

(2) $f, g \in \mathbb{F}[x]$ *have a common factor in* $\mathbb{F}[x]$ *if and only if their resultant vanishes.*

(3) If char$(\mathbb{F}) = 0$*, a polynomial* f *is square free if and only if* $\Delta(f) \neq 0$*. Moreover, the last statement is also true if deg*$(f) <$ *char*$(\mathbb{F})$*.*

(4) Let $f, g \in \mathbb{F}[x_0, \mathbf{x}]$ *be two* s*-sparse polynomials of bounded individual degree* d*. Then, their Sylvester matrix* $M_{x_0}(f, g)$ *is a* $2d \times 2d$ *matrix whose entries are* s*-sparse elements of* $\mathbb{F}[\mathbf{x}]$*, and moreover,* $\dim(\ker(M_{x_0}(f, g))) = \deg_{x_0}(\gcd(f, g))$*. In particular,*

(1) *If* $f, g \in \mathbb{F}[x_0, \mathbf{x}]$ *are two* s*-sparse polynomials of bounded individual degree* d*, then* $\mathrm{res}_{x_0}(f, g) \in \mathbb{F}[\mathbf{x}]$ *is a* $(2d)!s^{2d}$*-sparse polynomial of bounded individual degree* $2d^2$*.*

(2) *If* $f \in \mathbb{F}[x_0, \mathbf{x}]$ *is an* s*-sparse polynomial of bounded individual degree* d*, then* $\Delta_{x_0} f \in \mathbb{F}[\mathbf{x}]$ *is a* $(2d)!s^{2d}$*-sparse polynomial of bounded individual degree* $2d^2$*.*

Proof. The first three statements are standard. The last statement is also well known, and its explanation is given for example in [Vol17], in the language of subresultants. We give the full proof of the "moreover" part here as well, as it plays an important role in the proof of Lemma 4.15.

Suppose $f, g \in \mathbb{F}[x_0, \mathbf{x}]$ are two s-sparse polynomials of bounded individual degree d. Think of both as elements of $\mathbb{F}(\mathbf{x})[x_0]$, and write $f = \prod_i h_i^{a_i}, g = \prod_i h_i^{b_i}$, where h_i are all the irreducible factors of fg, and $a_i, b_i \geqslant 0$ (for example, if f, g are coprime, then $a_i b_i = 0$ for every i). Note that $\gcd(f, g) = \prod_i h_i^{\min(a_i, b_i)}$ and $\mathrm{lcm}(f, g) = \prod_i h_i^{\max(a_i, b_i)}$.

Consider an element $(w_1, w_2) \in \ker(M_{x_0}(f, g))$. Then, by definition,

$$w_1 g = -w_2 f. \tag{5}$$

Therefore, by considering the factorization into irreducible factors of both sides, it must be the case that $\prod_i h_i^{\max(a_i - b_i, 0)} | w_1$ and $\prod_i h_i^{\max(b_i - a_i, 0)} | w_2$. Thus, we can write

$$w_1 = \prod_i h_i^{\max(a_i - b_i, 0)} \tilde{w}_1, \quad w_2 = \prod_i h_i^{\max(b_i - a_i, 0)} \tilde{w}_2 \,. \tag{6}$$

It follows that we have

$$\begin{aligned} w_1 g &= \prod_i h_i^{b_i + \min(a_i - b_i, 0)} \tilde{w}_1 = \prod_i h_i^{\max(a_i, b_i)} \tilde{w}_1 = \mathrm{lcm}(f, g) \tilde{w}_1 \\ w_2 f &= \prod_i h_i^{a_i + \min(b_i - a_i, 0)} \tilde{w}_2 = \prod_i h_i^{\max(a_i, b_i)} \tilde{w}_2 = \mathrm{lcm}(f, g) \tilde{w}_2 \,. \end{aligned} \tag{7}$$

Combining (5) and (7), we get that $\tilde{w}_1 = -\tilde{w}_2$. Since $\deg w_1, \deg w_2 \leqslant d - 1$, we get from (6) that an element in $\ker(M_{x_0}(f, g))$ corresponds to a polynomial $\tilde{w}$ of degree at most

$$\begin{aligned} d - 1 - \sum_i \max(a_i - b_i, 0) \deg h_i &= \sum_i (a_i - \max(a_i - b_i, 0)) \deg(h_i) - 1 \\ &= \sum_i \min(a_i, b_i) \deg(h_i) - 1 = \deg_{x_0}(\gcd(f, g)) - 1 \,. \end{aligned} \tag{8}$$

Therefore, $\ker(M_{x_0}(f, g))$ is isomorphic to $\mathbb{F}[x]_{\leqslant (\deg_{x_0} \gcd(f,g) - 1)}$, and hence is of dimension $\deg_{x_0} \gcd(f, g)$, as claimed. $\square$

3 Generator Construction

In this section, we construct our generator, which is a map $G : \mathbb{F}^6 \to \mathbb{F}^n$ that satisfies the properties described in Section 1.4.1.

In what follows we let m be some parameter (in our setting we will usually set it to $m = (nsd)^2$ when dealing when (n, s, d)-sparse polynomials). To define our generator $G_{(m)}$ we will use the following notational conventions. Let $q = \Theta(m)$ be a prime (for concreteness q can be the first prime larger than m). We shall also assume that $|\mathbb{F}| > ndq$. If this is not the case then we work over an extension field $\mathbb{F} \subset \mathbb{E}$ of size $dnq < |\mathbb{E}| \leqslant (dnq)^2$.

Fix the following sets:

(1) A set of integers $\mathcal{I} = \{1, \ldots, m\}$ of size m, that contains m different residues mod q.

(2) A set of field elements $\mathcal{A} = \{\alpha_1, \ldots, \alpha_m\} \subset \mathbb{F}$ of size m. For every $1 \leqslant i \leqslant m$ let A_i be the Lagrange interpolating polynomials for $\mathcal{A}$.

(3) A set of field elements $\mathcal{B} = \{\beta_1, \ldots, \beta_n\} \subset \mathbb{F}$ of size n. For every $1 \leqslant i \leqslant n$ let B_i be the Lagrange interpolating polynomials for $\mathcal{B}$.

(4) A set of field elements $\mathcal{C} = \{\gamma_1, \ldots, \gamma_n\} \subset \mathbb{F}$ of size n. For every $1 \leqslant i \leqslant n$ let C_i be the Lagrange interpolating polynomials for $\mathcal{C}$.

(5) A set of field elements $\mathcal{T} = \{\delta_1, \ldots, \delta_{ndq}\} \subset \mathbb{F}$ of size ndq.

To define our generator we first define the generator of Klivans and Spielman and state some of its properties.

Definition 3.1 (The KS-generator)**.** In the notations above, the KS-generator is defined as

$$G^{\mathrm{KS}}_{(m)}(x, y, z, w) = \sum_{i=1}^{m} A_i(y) \left((1 + B_1(z)(w-1))x^{i \bmod q}, \ldots, (1 + B_n(z)(w-1))x^{i^n \bmod q} \right) . \tag{9}$$

Theorem 3.2 ([KS01, Theorem 11])**.** *Let $f \in \mathbb{F}[\mathbf{x}]$ be an (n, s, d)-sparse polynomial. Let $m = (nds)^2$. Then, f can be reconstructed in time $\mathrm{poly}(n, s, d)$ from the following set of evaluations:*

$$\left\{ f\left(G^{KS}_{(m)}(x, y, z, w) \right) \mid y \in \mathcal{A},\ z \in \mathcal{B},\ x, w \in \mathcal{T} \right\} .$$

Proof. Observe that when setting $y = a_i$ and $z = b_j$ we obtain

$$G^{\mathrm{KS}}_{(m)}(x, a_i, b_j, w) = \left(x^{i \bmod q}, \ldots, x^{i^{j-1} \bmod q}, wx^{i^j \bmod q}, x^{i^{j+1} \bmod q}, \ldots, x^{i^n \bmod q} \right) .$$

The claim follows from the analysis in [KS01, Theorem 11]. □

Another tool that we will use is the generator defined in [SV09], which is given by the formula

$$G^{\mathrm{SV}}(u, v) = (C_1(v)u, \ldots, C_n(v)u) \ . \tag{10}$$

The basic property of this generator is that it makes it possible to "revive" a variable of our choice. E.g., by setting $v = \gamma_j$ we get the vector that has zeros in all coordinates except of the j-th coordinate where we have the variable u.

We are now ready to define our generator.

Definition 3.3 (The generator $G_{(m)}$)**.** The generator $G_{(m)} : \mathbb{F}^6 \to \mathbb{F}^n$ is defined as

$$G_{(m)}(x, y, z, w, u, v) = G^{\mathrm{KS}}_{(m)}(x, y, z, w) + G^{\mathrm{SV}}(u, v) \ . \tag{11}$$

We further define the generators obtained from reviving a variable. Denote with $G^{\mathrm{KS}}_{(m)}(x, y, z, w)_i$ the i-th coordinate of $G^{\mathrm{KS}}_{(m)}(x, y, z, w)$.

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

$$G_{(m,i)}(x, y, z, w, u) = \left(G^{\mathrm{KS}}_{(m)}(x, y, z, w)_1, \ldots, u, \ldots, G^{\mathrm{KS}}_{(m)}(x, y, z, w)_n \right) \ . \tag{12}$$

Let's explain this definition. The first part $G^{KS}_{(m)}$ enables us to interpolate (n, s, d)-sparse polynomials (Theorem 3.2). The second part allows us to "revive" a variable of our choice, a feature that will play an important role in the analysis.

The following lemma shows that our generator indeed satisfies the first property described in Section 1.4.1.

Lemma 3.4. *Let* f *be an* (n, s, d)*-sparse polynomial and* $m = (nds)^2$*. Then one can recover* f *from* $f(G_{(m)})$ *in time* $\mathrm{poly}(m)$*. In particular, it holds that* $f = 0$ *if and only if* $f(G_{(m)}) = 0$*. In fact, for every* $1 \leqslant i \leqslant n$*, the statement is also true when replacing* $G_{(m)}$ *with* $G_{(m,i)}$ *and with* $G^{KS}_{(m)}$*.*

Proof. For every $1 \leqslant i \leqslant n$, the image of each one of $G_{(m)}$, $G_{(m,i)}$ and $G^{KS}_{(m)}$ includes image of $G^{KS}_{(m)}$, so the claim follows from Theorem 3.2. □

We conclude this section with the following three corollaries of the previous lemma.

Corollary 3.5. *Suppose* $f \in \mathcal{P}_{Factors}(n, s, d)$*. Then* $f(G_{(m)}) \neq 0$ *for* $m = (nds)^2$*. In fact, for every* $1 \leqslant i \leqslant n$*, the statement is also true when replacing* $G_{(m)}$ *with* $G_{(m,i)}$ *and with* $G^{KS}_{(m)}$*.*

Proof. Let $g \in \mathcal{P}(n, s, d)$ so that $f|g$, and let G be any one of $G_{(m)}$, $G_{(m,i)}$ and $G^{KS}_{(m)}$. Lemma 3.4 implies that $g(G) \neq 0$. The claim follows from the fact that $f(G)|g(G)$. □

Corollary 3.6. *Let* $f \in \mathcal{P}_{Factors}(n, s, d)$*. Then,* $\deg_u(f(G_{(m,i)})) = \deg_{x_i}(f)$ *for every* $1 \leqslant i \leqslant n$ *and* $m = (nsd)^2$*.*

Proof. Consider the leading coefficient[4] $LC_{x_i}(f)$ of f as a polynomial in x_i, and let $g \in \mathcal{P}(n, s, d)$ be any polynomial such that $f \mid g$. Then the leading coefficient $LC_{x_i}(g)$ of g (viewed as a polynomial in x_i) is divisible by $LC_{x_i}(f)$. Since g is s-sparse, its leading coefficient is also s-sparse. Therefore, $LC_{x_i}(f)$ is a factor of an $(n-1, s, d)$-sparse polynomial, and hence, by Corollary 3.5, we have $LC_{x_i}(f)(G^{KS}_{(m)}) \neq 0$ when substituting all variables except x_i.

By (12), $LC_{x_i}(f)(G^{KS}_{(m)})$ is precisely the leading coefficient of $f(G_{(m,i)})$ as a polynomial in u. Therefore, we must have $\deg_u f(G_{(m,i)}) = \deg_{x_i}(f)$. □

Corollary 3.7. *There is a deterministic* $\mathrm{poly}(n, d, s, \ell)$*-time algorithm that, given blackbox access to a product* f *of* ℓ *polynomials in* $\mathcal{P}_{Factors}(n+1, s, d) \subset \mathbb{F}[x_0, \mathbf{x}]$*, returns the multiplicity* k *of* x_0 *as a factor of* f*. Moreover, if* $f = {x_0}^k g$*, one can obtain blackbox access to* g *from blackbox access to* f*.*

Proof. Let $f = {x_0}^k \prod_i f_i$ be the factorization of f into irreducible factors, where $f_i \neq x_0$. For each i, let g_i be an $(n+1, d, s)$-sparse polynomial such that $f_i|g_i$. Let $m = ((n+1)ds)^2$. Consider the polynomial

$$f(x_0, G_{(m)}) = {x_0}^k \prod_i f_i(G_{(m)}) .$$

Fix some i, and denote $f_i = \sum_{j=0}^{d} f_{i,j}{x_0}^j$ and $g_i = \sum_{j=0}^{d} g_{i,j}{x_0}^j$, where $f_{i,j}, g_{i,j} \in \mathbb{F}[\mathbf{x}]$. As $x_0 \nmid f_i$, it follows that $f_{i,0} \neq 0$ for every i. By dividing out powers of x_0 we may therefore assume that $x_0 \nmid g_i$ (note that when doing so, we do not change the sparsity of g_i).

Since g_i is an $(n+1, d, s)$-sparse polynomial it follows that $g_{i,0}$ is (n, s, d)-sparse polynomial. Furthermore, for every i we have $f_{i,0}|g_{i,0}$. Therefore, it follows from Corollary 3.5 that $f_{i,0}(G_{(m)}) \neq 0$, and hence $x_0 \nmid f_i(G_{(m)})$. That is, the multiplicity of x_0 as a factor of f is the same

[4]The leading coefficient of a polynomial in a variable x is the coefficient of the highest power of x appearing nontrivially in the polynomial.

as its multiplicity as a factor of $f(x_0, G_{(m)})$. As the latter is a polynomial on a constant number of variables of bounded individual degree poly(n, d, s, ℓ), we can interpolate it using Lemma 2.9 and directly find k (which is the minimal power of x_0 dividing all monomials of $f(x_0, G_{(m)})$).

For the "moreover" part, let $f = {x_0}^k g$, and suppose $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \ldots, \alpha_n) \in \mathbb{F}^{n+1}$ is a point we want to evaluate g on. Consider $h(x) = f(x, \alpha_1, \ldots, \alpha_n) = x^k g(x, \alpha_1, \ldots, \alpha_n)$ as a polynomial in $\mathbb{F}[x]$. One can use a standard one-variable interpolation (as stated in Lemma 2.9) to recover $h(x)$ from blackbox access to f, and therefore to recover the polynomial $g(x, \alpha_1, \ldots, \alpha_n) = h(x)/x^k$. By plugging $x = \alpha_0$ one can then recover $g(\boldsymbol{\alpha})$, as we wanted. $\square$

Corollary 3.8 (Finding monomial divisors)**.** *There is a deterministic* poly(n, d, s, ℓ)*-time algorithm that, given blackbox access to a product f of ℓ polynomials in $\mathcal{P}_{Factors}(n, s, d) \subset \mathbb{F}[\mathbf{x}]$, returns the multiplicity k_i of x_i as a factor of f. Moreover, if M is the largest monomial divisor of f, such that $f = Mg$, then we can obtain blackbox access to g from blackbox access to f.*

Proof. This is an immediate corollary of Corollary 3.7. We find the maximal power of each variable dividing f. This gives us the monomial M. To get blackbox access to g we do the following: Given a point $\boldsymbol{\alpha} \in \mathbb{F}^n$, if all coordinates of $\boldsymbol{\alpha}$ are nonzero then we can compute $g(\boldsymbol{\alpha}) = f(\boldsymbol{\alpha})/M(\boldsymbol{\alpha})$. If some coordinates of $\boldsymbol{\alpha}$ are zero, so we cannot divide by $M(\boldsymbol{\alpha})$, then we do the following. Consider the substitution $\mathbf{x} \leftarrow \boldsymbol{\alpha} + (t, \ldots, t)$, where t is a new variable. By evaluating $\tilde{g}_{\boldsymbol{\alpha}}(t) := (f/M)(\boldsymbol{\alpha} + (t, \ldots, t))$ at more than dn values of t such that $t \notin \{-\alpha_1, \ldots, -\alpha_n\}$, we can interpolate $\tilde{g}(t)$. Then we can evaluate $g(\boldsymbol{\alpha}) = \tilde{g}_{\boldsymbol{\alpha}}(0)$. Thus, any query to g requires $O(dn)$ queries to f. $\square$

Remark 3.9. The first part of the previous corollary is an easy special case of the technical lemmas in Section 4. In particular, it is always possible to determine the multiplicity of an irreducible sparse polynomial as a factor of such an f using the factorization of $f(x_0, G_{(m)})$. However, as we show in the next section, a larger value of m is required in general.

4 Divisibility Testing and Exact Power testing

In this section we prove our divisibility testing and exact power testing results. For that we first have to prove some technical results about our generator G. We split the discussion on these technical results into two parts. In Section 4.1 we obtain some results that are valid over arbitrary fields; in Section 4.2 we prove a stronger version of these results, that is only valid over zero or large enough characteristic.

4.1 Arbitrary Characteristic Results

In this subsection we build on the methods of [BV25] to obtain our divisibility testing results over fields of arbitrary characteristic. To do that, we define and study the notion of primitive divisors for a class of polynomials and develop the tools needed for our algorithms; this is done in Section 4.1.1. In Section 4.1.2 we use these tools to obtain our algorithms.

4.1.1 Arbitrary Characteristic Tools

We start with the definition of a *primitive divisor*.

Definition 4.1. Let $\mathcal{P} \subset \mathbb{F}[\mathbf{x}]$ be a class of polynomials. We say that a polynomial $g \in \mathbb{F}[\mathbf{x}]$ is a primitive divisor for $\mathcal{P}$ if it satisfies the following two conditions.

(1) For every $f \in \mathcal{P}$ and every $t \in \mathbb{N}$, the polynomial $\gcd(f, g^t)$ is a power of g. Equivalently, for every $f \in \mathcal{P}$, there exists $k \geqslant 0$ and a polynomial $\tilde{g}$ coprime to g such that $f = g^k \cdot \tilde{g}$.

(2) No non-scalar multiple of g satisfies the first condition.

Example 4.2. *An irreducible polynomial $f \in \mathcal{P}$ is clearly a primitive divisor for $\mathcal{P}$.*

Remark 4.3. Every primitive divisor for $\mathcal{P}$ divides some polynomial $f \in \mathcal{P}$. Indeed, suppose that g is a primitive divisor for $\mathcal{P}$ that does not divide any polynomial in $\mathcal{P}$. Then, for every $f \in \mathcal{P}$, we have $\gcd(f, g) = 1$, and hence g is coprime to every element of $\mathcal{P}$. It follows that $\gcd(f, g^2) = 1$ for every $f \in \mathcal{P}$, so g^2 also satisfies the first condition. This contradicts the second condition for g.

Remark 4.4. It is not true that primitive divisors for $\mathcal{P}$ are necessarily elements of $\mathcal{P}$. Consider for example the class of polynomials $\mathcal{P} = \{g^2, g^3\}$ for some non-constant polynomial $g \in \mathbb{F}[\mathbf{x}]$. Then, the only primitive divisor (up to multiplication by scalars) for $\mathcal{P}$ is g, which is not an element of $\mathcal{P}$.

In some sense, primitive divisors are to a class of polynomials $\mathcal{P}$ what irreducible polynomials are to the class of all polynomials. To make this statement precise, we prove that every $f \in \mathcal{P}$ can be written uniquely as a product of primitive divisors. That is, primitive divisors for $\mathcal{P}$ are indeed the "minimal building blocks" for the polynomials in $\mathcal{P}$.

To prove that, we start by showing that any two different primitive divisors for a class $\mathcal{P}$ are coprime.

Lemma 4.5. *Let $\mathcal{P} \subset \mathbb{F}[\mathbf{x}]$ be a class of polynomials, and let g and h be two non-associate primitive divisors for it. Then $\gcd(g, h) = 1$.*

Proof. Suppose for the sake of contradiction that $\gcd(g, h) \neq 1$, and let $f_1, \dots, f_N$ be the irreducible factors of it, for $N \geqslant 1$. Write:

$$g = \prod_{1 \leqslant i \leqslant N} {f_i}^{u_i} \cdot \prod_j g_j^{e_j}, \quad h = \prod_{1 \leqslant i \leqslant N} {f_i}^{v_i} \cdot \prod_j h_j^{r_j} \tag{13}$$

where f_i, g_j, h_j are all different irreducible polynomials, and all the exponents u_i, v_i, e_j, r_j are positive integers.

We claim that the vectors $(u_1, \dots, u_N)$ and $(v_1, \dots, v_N)$ are proportional. Indeed, consider some $\phi \in \mathcal{P}$ for which $g | \phi$. Then, by the first condition of Definition 4.1, we must have k' and t' for which

$$\phi = g^{k'} \cdot \tilde{g}, \text{ where } \gcd(g, \tilde{g}) = 1 \text{ and}$$
$$\phi = h^{t'} \cdot \tilde{h}, \text{ where } \gcd(h, \tilde{h}) = 1\,.$$

Therefore, by considering the multiplicities of the f_i's as factors of ϕ, we must have that

$$k'(u_1, \dots, u_n) = t'(v_1, \dots, v_n). \tag{14}$$

Let $(k, t) := (k'/\gcd(k', t'), t'/\gcd(k', t'))$. That is, (k, t) is the smallest pair of positive integers satisfying (14). Consider the polynomial given by:

$$\tilde{\phi} = \prod_{1 \leqslant i \leqslant n} f_i^{k u_i} \cdot \prod_j g_j^{k e_j} \cdot \prod_j h_j^{t r_j}. \tag{15}$$

Note that as $k, t \geqslant 1$ ($k \geqslant 1$ because $g|\phi$, and $t \geqslant 1$ by (14)), ϕ is a non-trivial multiple of g and h.

We claim that $\tilde{\phi}$ satisfies the first condition of Definition 4.1. Indeed, let $\psi \in \mathcal{P}$. Since g and h are primitive divisors, we can write:

$$\begin{aligned} \psi &= g^{k_0} \cdot \hat{g} \text{ where } \gcd(g, \hat{g}) = 1. \\ \psi &= h^{t_0} \cdot \hat{h} \text{ where } \gcd(h, \hat{h}) = 1. \end{aligned} \tag{16}$$

By examining the multiplicities of f_i as factors of ψ, we get that $k_0(u_1, \ldots, u_N) = t_0(\nu_1, \ldots, \nu_N)$. As (k, t) is the minimal pair of positive integers satisfying (14), we must have that $k_0 = ck$ and $t_0 = ct$ for some non-negative integer c. Now, by plugging this information back to (16), it can be easily checked that $\psi = \tilde{\phi}^c \cdot \hat{\psi}$ for some polynomial $\hat{\psi}$ coprime to $\tilde{\phi}$. This proves that $\tilde{\phi}$ satisfies the first condition of Definition 4.1. However, this is a contradiction to the fact that g and h satisfy the second condition of it. This proves that g and h cannot share an irreducible factor, and completes the proof. □

We next state a lemma capturing the intuition that primitive divisors play a similar role to irreducible polynomials.

Lemma 4.6. *Let $\mathcal{P}$ be some class of polynomials, and let $f \in \mathcal{P}$. Then, f can be uniquely written as a product of primitive divisors for $\mathcal{P}$.*

Proof. Let $f \in \mathcal{P}$. Consider some irreducible factor f_0 of f. Clearly, f_0 satisfies the first condition of Definition 4.1. Let g_0 be a maximal divisor of f that satisfies the property that it is divisible by f_0 and satisfies the first condition of Definition 4.1. We claim that g_0 must be a primitive divisor. For this we must show that it satisfies the second condition of Definition 4.1. Assume for a contradiction that there is some h, which is a non-trivial multiple of g_0, that satisfies the first condition of Definition 4.1. By maximality of g_0, $h \nmid f$. From the first condition we obtain that $\gcd(f, h) = 1$. However, since $f_0 \mid g_0$ and $g_0 \mid h$, we get that $f_0 | \gcd(f, h)$, in contradiction.

Thus, we identified some primitive divisor $g_0 | f$. By definition, we can write $f = g_0^k \tilde{g}_0$ with $\gcd(g_0, \tilde{g}_0) = 1$. If $\tilde{g}_0 = 1$ then we are done. If not, fix some irreducible factor $f_1 | \tilde{g}_0$. By the same reasoning as before, there is a multiple of f_1, denoted g_1, which is a primitive divisor for $\mathcal{P}$, and that divides f. By Lemma 4.5, and by the fact that f_1 divides g_1 but not g_0, it follows that we must have that g_0 and g_1 are coprime. Therefore, we can write $f = g_0^{k_0} g_1^{k_1} \tilde{g}_1$, where $\tilde{g}_1$ is coprime to both g_0 and g_1. Continuing this way, we can obtain the desired factorization of f to a product of primitive divisors.

To end the proof, we note that the uniqueness of the factorization thus obtained follows from Lemma 4.5. □

We can therefore define the multiplicity with which a primitive divisor divides a polynomial in the class.

Definition 4.7. Let $f \in \mathcal{P}$, and let g be a primitive divisor for $\mathcal{P}$. The multiplicity of g as a factor of f, denoted by $\nu_g(f)$, is the power of g appearing in the factorization of f to primitive divisors.

Next, we discuss the connection of the definition of primitive divisors and the non-degeneracy condition of [BV25]. The proof of the next lemma follows from [BV25, Lemma 4.5,4.6].

Lemma 4.8 ([BV25, Lemma 4.5,4.6])**.** *Let ϕ and ψ be two non-associate* irreducible *polynomials. Fix some i for which $x_i \in var(\phi)$. Suppose that there are s-sparse polynomials f and g of bounded individual degree d so that $\phi | f$ and $\psi | g$. Denote with $\nu_\phi(f)$ the multiplicity of ϕ as a factor of f, and similarly for ψ and g. Suppose that the matrix $\begin{pmatrix} \nu_\phi(f) & \nu_\phi(g) \\ \nu_\psi(f) & \nu_\psi(g) \end{pmatrix}$ is invertible. Then, the polynomials $\phi(G_{(m,i)})$ and $\psi(G_{(m,i)})$ are coprime as polynomials in u for some explicit $m = \mathrm{poly}(n, (d^2)!, s^{d^3})$.*

The condition of the lemma can be better understood in terms of primitive divisors, as demonstrated by the following claim.

Claim 4.9. *Let $\mathcal{P}$ be a class of polynomials, and let ϕ and ψ be two irreducible polynomials that divide some (not necessarily the same) polynomial in $\mathcal{P}$. Then, the following are equivalent.*

(1) ϕ and ψ divide the same primitive divisor of $\mathcal{P}$.

(2) For every $f, g \in \mathcal{P}$ it holds that the matrix $\begin{pmatrix} \nu_\phi(f) & \nu_\phi(g) \\ \nu_\psi(f) & \nu_\psi(g) \end{pmatrix}$ is not invertible.

Proof. First, note that the term "the primitive divisor that ϕ divides" is well defined, as by Lemma 4.5 there is at most one (up to multiplication by scalars) such primitive divisor, and by the fact that ϕ divides some polynomial in $\mathcal{P}$ there is at least one such divisor; clearly, the same is also true for ψ.

For the implication ((1)) $\implies$ ((2)) note that if h is the primitive divisor that ϕ and ψ divide, by Lemma 4.5 and Lemma 4.6 it follows that for every $f \in \mathcal{P}$ we have that $(\nu_\phi(f), \nu_\psi(f)) = \nu_h(f)(\nu_\phi(h), \nu_\psi(h))$. Therefore, it is clear that for every pair $f, g \in \mathcal{P}$ the vectors $(\nu_\phi(f), \nu_\psi(f))$ and $(\nu_\phi(g), \nu_\psi(g))$ are proportional, and hence the matrix whose columns are these vectors is not invertible.

The next claim establishes the implication ((2)) $\implies$ ((1)).

Claim 4.10. *Let h_1 and h_2 be two non-associate primitive divisors for a class $\mathcal{P}$. Then, there are polynomials $f_1, f_2 \in \mathcal{P}$ for which the matrix $\begin{pmatrix} \nu_{h_1}(f_1) & \nu_{h_1}(f_2) \\ \nu_{h_2}(f_1) & \nu_{h_2}(f_2) \end{pmatrix}$ is invertible.*

Proof. Consider the set of vectors $(\nu_{h_1}(f), \nu_{h_2}(f))_{f \in \mathcal{P}}$. To prove the claim, it is enough to prove that there are two independent vectors in this set. Suppose for the sake of contradiction that this is not the case. Then, it follows that the $\mathbb{Z}$-span of this set is a one dimensional lattice, generated by some vector (k, t).

We claim that $h := h_1^k h_2^t$ must satisfy the first condition of Definition 4.1. Indeed, consider some element $\tilde{f} \in \mathcal{P}$. By Lemma 4.6 we can write $\tilde{f} = h_1^{\nu_{h_1}(f)} h_2^{\nu_{h_2}(f)} g$, for some polynomial g coprime to both h_1 and h_2. By definition of (k, t), there is some non-negative integer c for which $(\nu_{h_1}(f), \nu_{h_2}(f)) = c(k, t)$. Thus, $\tilde{f} = h_1^{ck} h_2^{ct} g = h^c g$, where g is coprime to h. That is, h indeed satisfies the first condition of Definition 4.1. As h_1 is a primitive divisor, $\nu_{h_1}(\tilde{f}) \geqslant 1$ for some $\tilde{f} \in \mathcal{P}$, and therefore we must have that $k \geqslant 1$. Similarly, we obtain $t \geqslant 1$. That is, h is a non-trivial multiple of h_1 and h_2 satisfying the first condition of Definition 4.1, in contradiction. This concludes the proof. $\square$

Observe that Claim 4.10 implies $\neg$ ((1)) $\implies$ $\neg$((2)). Indeed, let h_1 and h_2 be the primitive divisors that ϕ and ψ divide, respectively. Suppose that h_1 and h_2 are not associated. By Claim 4.10, there are $f_1, f_2 \in \mathcal{P}$ such that the matrix $\begin{pmatrix} \nu_{h_1}(f_1) & \nu_{h_1}(f_2) \\ \nu_{h_2}(f_1) & \nu_{h_2}(f_2) \end{pmatrix}$ is invertible. From divisibility we conclude

$$\begin{pmatrix} \nu_{h_1}(f_1) & \nu_{h_1}(f_2) \\ \nu_{h_2}(f_1) & \nu_{h_2}(f_2) \end{pmatrix} = \begin{pmatrix} \frac{\nu_\phi(f_1)}{\nu_\phi(h_1)} & \frac{\nu_\phi(f_2)}{\nu_\phi(h_1)} \\ \frac{\nu_\psi(f_1)}{\nu_\psi(h_2)} & \frac{\nu_\psi(f_2)}{\nu_\psi(h_2)} \end{pmatrix}$$

and therefore

$$\det \begin{pmatrix} \nu_\phi(f_1) & \nu_\phi(f_2) \\ \nu_\psi(f_1) & \nu_\psi(f_2) \end{pmatrix} = \nu_\phi(h_1)\nu_\psi(h_2) \det \begin{pmatrix} \nu_{h_1}(f_1) & \nu_{h_1}(f_2) \\ \nu_{h_2}(f_1) & \nu_{h_2}(f_2) \end{pmatrix} \neq 0.$$

Hence the matrix $\begin{pmatrix} \nu_\phi(f_1) & \nu_\phi(f_2) \\ \nu_\psi(f_1) & \nu_\psi(f_2) \end{pmatrix}$ is invertible. This concludes the proof. □

Claim 4.9 explains in a more natural way the condition from Lemma 4.8: if the matrix of multiplicities is degenerate, then ϕ and ψ divide the same primitive divisor for $\mathcal{P}(n, s, d)$, and are therefore "inseparable" from the point of view of this class. This motivates us to formulate an analog of Lemma 4.8 for every two primitive divisors of $\mathcal{P}(n, s, d)$.

Lemma 4.11. *Let ϕ and ψ be two non-associate primitive divisors for $\mathcal{P}(n, s, d)$. Fix some $1 \leqslant i \leqslant n$ such that $x_i \in var(\phi)$. Then for some explicit value $m = \mathrm{poly}(n, (d^2)!, s^{d^3})$ we have that $\phi(G_{(m,i)})$ is coprime to $\psi(G_{(m,i)})$, when both are viewed as polynomials in u.*

The proof will be nearly identical to the proof of Lemma 4.8, which captures [BV25, Lemma 4.6], with the main difference being that Lemma 4.8 concerns irreducible polynomials while Lemma 4.11 considers primitive divisors. We give a sketch of it for the sake of completeness.

Proof. In this proof, whenever we refer to a primitive divisor, we mean a primitive divisor for $\mathcal{P}(n, s, d)$.

By Claim 4.9, there are $f_1, f_2 \in \mathcal{P}(n, s, d)$ for which the matrix

$$\begin{pmatrix} k_1 & t_1 \\ k_2 & t_2 \end{pmatrix} := \begin{pmatrix} \nu_\phi(f_1) & \nu_\phi(f_2) \\ \nu_\psi(f_1) & \nu_\psi(f_2) \end{pmatrix}$$

is invertible. Let $h_1 = f_1^{t_1+t_2}$ and $h_2 = f_2^{k_1+k_2}$. By a simple calculation it can be shown that, up to switching h_1 and h_2, we have $h_1 = \gcd(h_1, h_2)\phi^a H_1$ and $h_2 = \gcd(h_1, h_2)\psi^b H_2$, where $a, b \geqslant 1$ and H_1, H_2 are coprime to ϕ, ψ (by unique factorization and properties of primitive divisors).

Consider the Sylvester matrix of $M_{x_i}(h_1, h_2)$. As both f_1 and f_2 have individual degrees bounded by d we have that $k_1 + k_2, m_1 + m_2 \leqslant 2d$. Therefore, h_1 and h_2 are s^{2d}-sparse polynomials of bounded individual degree $2d^2$. Thus, by Lemma 2.20, determinants of minors of $M_{x_i}(h_1, h_2)$ are $(4d^2)!s^{8d^3}$-sparse polynomials of individual degree at most $8d^4$. Lemma 3.4 implies that for

$$m = \left(n(8d^4)((4d^2)!s^{8d^3})\right)^2 = \mathrm{poly}(n, (d^2)!, s^{d^3})$$

no nonzero determinant of a minor vanishes when composing with $G_{(m,i)}$. Therefore, it follows that

$$\mathrm{rank}(M_{x_i}(h_1, h_2)) = \mathrm{rank}(M_u(h_1(G_{(m,i)}), h_2(G_{(m,i)}))) .$$

Lemma 2.20((4)) then ensures that $\deg_{x_i}(\gcd(h_1, h_2)) = \deg_u(\gcd(h_1(G_{(m,i)}), h_2(G_{(m,i)})))$. Hence, composition with $G_{(m,i)}$ did not change the degree in u. Consequently, $\phi(G_{(m,i)})$ and $\psi(G_{(m,i)})$ are coprime as polynomials in u, since otherwise their common factor would increase the degree of the gcd.

This concludes the sketch of the proof. We note that in Lemma 4.15, we prove a similar claim, where we explain in more detail the technical points in the argument above. These are also explained, of course, in [BV25, Lemma 4.5,4.6]. □

We next state an important corollary.

Corollary 4.12. *Let $\phi \in \mathcal{P}(n, s, d)$ be an irreducible polynomial, and let $\psi \in \mathcal{P}_{Factors}(n, s, d)$ be an irreducible factor of some $g \in \mathcal{P}(n, s, d)$, which is not associate to ϕ. Fix some $1 \leqslant i \leqslant n$ such that $x_i \in var(\phi)$. Then, for an explicit $m = \mathrm{poly}(n, (d^2)!, s^{d^3})$, we have that $\phi(G_{(m,i)})$ is coprime to $\psi(G_{(m,i)})$, when viewed as polynomials in u. Moreover, if ϕ is multiquadratic, we can take $m = \mathrm{poly}(n, (d^2)!, s^{d^2})$.*

Proof. Intuitively, this should follow from Lemma 4.11 (recalling that an irreducible polynomial in $\mathcal{P}(n,s,d)$ is also a primitive divisor for it Example 4.2). Technically speaking we have in hand an irreducible polynomial ψ, which is not necessarily a primitive divisor (as it might not belong to $\mathcal{P}(n,s,d)$). We get around this issue in a similar manner to the proof of Claim 4.9. We instead consider the (unique, up to multiplication by scalars) primitive divisor $\tilde{\psi}$ that ψ divides. Clearly ϕ and $\tilde{\psi}$ are non-associate, so we can just apply Lemma 4.11 with ϕ and $\tilde{\psi}$ and conclude the claim.

For the "moreover" part, we note that the proof of Lemma 4.11 actually gives a better result when ϕ is multiquadratic. We describe what changes should be made in the proof of Lemma 4.11 to see it; from now on, we use the notations from that argument.

In this case, as $\phi \in \mathcal{P}(n,s,d)$ is irreducible, one can take $f_1 = \phi$, and f_2 to be some polynomial that $\tilde{\psi}$ divides. Therefore, we have that $k_1 = 1, k_2 = 0$. It follows that:

- $h_1 = f_1^{t_1+t_2} = \phi^{t_1+t_2}$ is a polynomial of degree at most $2(t_1+t_2) \leqslant 4d$, instead of $2d^2$, as a polynomial in x_i. Its sparsity is bounded above by s^{2d}.
- $h_2 = f_2^{k_1+k_2} = f_2$ is a polynomial of degree at most d, instead of $2d^2$, as a polynomial in x_i. Its sparsity is bounded above by s.

As a result, it follows that any determinant of a minor of $M_{x_i}(h_1,h_2)$ is an $s^{O(d^2)}$-sparse polynomial, of bounded individual degree at most $O(d^2)$. This implies that it is sufficient to take $m = \mathrm{poly}(n,(d^2)!,s^{d^2})$ in this case. □

4.1.2 Arbitrary Characteristic Algorithms

Our next goal is to obtain a divisibility testing result for products of sparse polynomials of bounded individual degree. To do that, we need the following lemma, which is the heart of the divisibility testing algorithm. This lemma will play a crucial part in Section 5, when we prove our rational interpolation result.

Lemma 4.13. *There is a deterministic* $\mathrm{poly}(n,(d^2)!,s^{d^3},\ell)$*-time algorithm that, given blackbox access to a product* f *of* ℓ (n,s,d)*-sparse polynomials, and blackbox access to a primitive divisor* ϕ *for* $\mathcal{P}(n,s,d)$*, returns the multiplicity of* ϕ *as a factor of* f *(in the sense of Definition 4.7). In fact, the algorithm only needs to know the values of* f *on the image of the generator* $G_{(m)}$*, for some explicit value* $m = \mathrm{poly}(n,(d^2)!,s^{d^3})$*. Moreover, if* ϕ *is an irreducible* s*-sparse polynomial, this algorithm works if* f *is a product of polynomials in* $\mathcal{P}_{Factors}(n,s,d)$ *(rather than* $\mathcal{P}(n,s,d)$*); if furthermore* ϕ *is multiquadratic, taking* $m = \mathrm{poly}(n,(d^2)!,s^{d^2})$ *suffices, and the running time reduces to* $\mathrm{poly}(n,(d^2)!,s^{d^2},\ell)$*.*

Proof. Fix some i for which $x_i \in \mathrm{var}(\phi)$, and let $m = \mathrm{poly}(n,(d^2)!,s^{d^3})$ so that the conclusion of Lemma 4.11 holds. Corollary 3.6 implies that $\deg_u(\phi(G_{(m,i)})) = \deg_{x_i}(\phi) \geqslant 1$.

By Lemma 4.6 we can write $f = \phi^k \cdot \prod \psi_j$, where ψ_j are primitive divisors for $\mathcal{P}(n,s,d)$ that are not associate to ϕ. From Lemma 4.11 it follows that for every ψ_j we have that $\phi(G_{(m,i)})$ is coprime to $\psi_j(G_{(m,i)})$, when both are viewed as polynomials in u. In particular, we have that $\gcd_u(\phi(G_{(m,i)}), \prod \psi_j(G_{(m,i)})) = 1$. As $\deg_u(\phi(G_{(m,i)})) \geqslant 1$, we must have that the highest power of $\phi(G_{(m,i)})$ that divides f as a polynomial in u is exactly k.

Thus, to recover the desired multiplicity, we will calculate, for every $1 \leqslant i \leqslant n$, the highest power of $\phi(G_{(m,i)})$ that divides f as polynomials in u, and return the minimal value. This power can be calculated by interpolating both $\phi(G_{(m,i)})$ and $f(G_{(m,i)})$ using Lemma 2.9 and computing the highest power of $\phi(G_{(m,i)})$ dividing $f(G_{(m,i)})$ as polynomial in u (just by regular polynomial division, there is no need for polynomial factorization here). Note that the degree of both polynomials is at most $\mathrm{poly}(n, d, s, m, \ell)$ so the claim on the running time holds. This ends the description of the algorithm.

As for the "moreover" part, it follows from the same proof, by considering the factorization into irreducible polynomials (rather than to primitive divisors), and by replacing the use of Lemma 4.11 with Corollary 4.12. □

We are finally ready to obtain our divisibility testing result. This is a generalization of the divisibility testing result of [Vol17, Theorem 7].

Theorem 4.14. *There exists a deterministic* $\mathrm{poly}(n, (d^2)!, s^{d^3}, \ell)$*-time algorithm that, given blackbox access to two products* f, g *of* ℓ (n, s, d)*-sparse polynomials, decides if* $f|g$*. Moreover, if all irreducible factors of* f *are in fact sparse (that is,* f *is a product of irreducible* (n, s, d)*-sparse polynomials), this algorithm works even if* g *is a product of polynomials in* $\mathcal{P}_{Factors}(n, s, d)$ *(rather than in* $\mathcal{P}(n, s, d)$*); if furthermore all factors of* f *are multiquadratic, then the running time of this algorithm can be reduced to* $\mathrm{poly}(n, (d^2)!, s^{d^2}, \ell)$.

Proof. Fix some $m = \mathrm{poly}(n, (d^2)!, s^{d^3})$ such that Lemma 4.13 applies. For every $1 \leqslant i \leqslant n$ check if $f(G_{(m,i)})$ divides $g(G_{(m,i)})$, and return that $f|g$ if all these checks passed. Clearly, if $f|g$ we return the correct answer. Suppose now that $f \nmid g$. By Lemma 4.6, we know that there is a primitive divisor ϕ whose multiplicity as a factor of f exceeds its multiplicity in g. Then, by the proof of Lemma 4.13 it follows that for every i for which $x_i \in \mathrm{var}(\phi)$, the highest power k for which $\phi(G_{(m,i)})^k$ divides $f(G_{(m,i)})$ actually equals the multiplicity of ϕ as a factor of f, and analogously for $g(G_{(m,i)})$. In particular, for such values of i it cannot be the case that $f(G_{(m,i)})|g(G_{(m,i)})$, so we will indeed return that $f \nmid g$.

As for the "moreover" part, it follows from the same proof, when considering the factorization of f and g into irreducible polynomials (rather than to primitive divisors), and by using the "moreover" part of Lemma 4.13. □

As we saw in this subsection, the method of [BV25] is pretty strong for dealing with problems concerning the factorization of products of sparse polynomials of bounded individual degree. Indeed, we will see in Section 5 that Lemma 4.13 is what we need for our rational interpolation result, and as a consequence, for our algorithm that recovers multiquadratic sparse irreducible factors of a product of sparse polynomials of bounded individual degree.

However, this method has some limitations. Consider for example the problem of deciding if a given product of sparse polynomials of bounded individual degree is a complete power. A natural thing to do is to apply an algorithm that is similar to the one presented in Theorem 4.14 - check if $f(G_{(m,i)})$ are complete powers for every i. This algorithm, however, does not work. To see why, suppose our input polynomial is a primitive divisor of the form $\phi = \phi_1\phi_2$ where ϕ_1, ϕ_2 are two non-associate irreducible polynomials. The methods of this subsection do not say anything about the interaction of $\phi_1(G_{(m,i)})$ and $\phi_2(G_{(m,i)})$. In particular, it might be the case that for every i, $\phi_1(G_{(m,i)}) = \phi_2(G_{(m,i)})$, so that $\phi(G_{(m,i)})$ is a perfect square.

The goal of the next subsection is to generalize the results of this subsection from dealing only with primitive divisors to dealing with irreducible polynomials. This allows us, for example, to make the suggested complete power testing work. Moreover, it enables us to generalize all results from the class of products of sparse polynomials to the class of products of factors of sparse polynomials.

Our generalization, however, only works for fields of characteristic zero, or when the characteristic is larger than twice the bound on the individual degree. In particular, the results of the present subsection are the best we can do for small characteristic fields.

4.2 Zero or Large Characteristic Results

In this subsection we prove stronger versions of the results of Section 4.1, both in terms of generality and running times. However, the results of this subsection are valid only when $\mathrm{char}(\mathbb{F}) = 0$ or when $\mathrm{char}(\mathbb{F}) > 2d$ (where d is the bound on the individual degree).

Similar to Section 4.1, in Section 4.2.1 we build the tools that we will use in our algorithms, and in Section 4.2.2 we give our algorithms.

4.2.1 Zero or Large Characteristic Tools

The main technical tool we will need is a generalization of Lemma 4.11 and Lemma 4.13 for the case of irreducible factors rather than primitive divisors. We prove the following analogue of Lemma 4.11.

Lemma 4.15. *Let $\mathbb{F}$ be a field of characteristic zero, or of characteristic larger than $2d$. Let ϕ, ψ be non-associate irreducible polynomials in $\mathcal{P}_{\mathit{Factors}}(n, s, d)$ that divide the (n, s, d)-sparse polynomials f, and g (possibly identical to f), respectively. Fix some i for which $x_i \in \mathit{var}(\phi)$. Then, for some explicit value $m = \mathrm{poly}(n, d!, s^d)$, we have that $\phi(G_{(m,i)})$ is coprime to $\psi(G_{(m,i)})$, when viewed as polynomials in u. Moreover, $\phi(G_{(m,i)})$ and $\psi(G_{(m,i)})$ are both square free as polynomials in u.*

Proof. Fix some i for which $x_i \in \mathrm{var}(\phi)$. From now on, view all polynomials as polynomials in x_i. Consider $M_{x_i}(fg, (fg)')$, the Sylvester matrix of fg and its formal derivative with respect to x_i. Let $fg = \prod_j h_j^{e_j}$ be the factorization of fg into irreducible factors (as a polynomial in x_i over $\mathbb{F}(\mathbf{x} \setminus x_i)$).

Claim 4.16. *Let $\mathbb{F}$ as in Lemma 4.15. Let $f \in \mathbb{F}[x]$ be a non-constant univariate polynomial of degree at most $2d$. Suppose $f = \prod_i h_i^{e_i}$ is the factorization of f into irreducible polynomials. Then $\dim(\ker(M(f, f')) = \sum_i \deg(h_i)(e_i - 1)$.*

Proof. Clearly, $f' = \sum_i e_i h_i^{e_i - 1} h_i' \hat{h}_i$, where we denote by $\hat{h}_i$ the product of the factors of f different than h_i. As h_i is a non-constant polynomial of degree at most $2d$, the conditions on $\mathrm{char}(\mathbb{F})$ imply $h_i' \neq 0$. Since h_i is irreducible, it must be the case that it is coprime to h_i'. Moreover, as h_i and $\hat{h}_i$ are clearly coprime, and as $e_i \neq 0$ (because of the condition on $\mathrm{char}(\mathbb{F})$), we have that the multiplicity of h_i as a factor of f' is precisely $e_i - 1$. As h_i are the only irreducible factors of f, it follows that $\gcd(f, f') = \prod_i h_i^{e_i - 1}$. Lemma 2.20 then implies that $\dim(\ker(M(f, f')) = \deg(\gcd(f, f')) = \deg(\prod_i h_i^{e_i - 1}) = \sum_i \deg(h_i)(e_i - 1)$, as claimed. □

Continuing with the proof, we clearly have that $\deg_{x_i}(fg) \leqslant 2d$. Therefore, by Claim 4.16 $r := \dim(\ker(M_{x_i}(fg, (fg)'))) = \sum_j \deg_{x_i}(h_j)(e_j - 1)$. As $M_{x_i}(fg, (fg)')$ is an $(4d-1) \times (4d-1)$ matrix, it contains an invertible $(4d - 1 - r) \times (4d - 1 - r)$ minor M. As fg and $(fg)'$ are s^2-sparse, it follows from Lemma 2.20 that the entries of M are s^2-sparse polynomials of bounded individual degree $2d$, and hence $\det M$ is an $(4d)!s^{8d}$-sparse polynomial of individual degree $8d^2$. Therefore, Corollary 3.5 implies that for $m = (n(8d^2)((4d)!s^{8d}))^2 = \mathrm{poly}(n, d!, s^d)$, it holds that $\det M(G_{(m,i)}) \neq 0$ (where we substitute $G_{(m,i)}$ to all coordinates except the i-th). Note that $\det M(G_{(m,i)})$ is precisely the determinant of a $(4d - 1 - r) \times (4d - 1 - r)$ minor of the matrix $M_u(fg(G_{(m,i)}), (fg')(G_{(m,i)}))$, and therefore it follows that

$$\dim(\ker(M_u(fg(G_{(m,i)}), (fg)'(G_{(m,i)})))) \leqslant r. \tag{17}$$

Now, we know that $fg(G_{(m,i)}) = \prod_j h_j(G_{(m,i)})^{e_j}$; let $fg(G_{(m,i)}) = \prod_k \tilde{h}_k^{r_k}$ be the factorization into irreducible factors of $fg(G_{(m,i)})$. We now claim the following.

Claim 4.17.

$$\sum_k \deg \tilde{h}_k (r_k - 1) \geqslant \sum_j \deg h_j(G_{(m,i)})(e_j - 1)$$

and equality holds if and only if the set $\{h_j(G_{(m,i)})\}$ is a set of square free, pairwise coprime polynomials.

Proof. By considering the factorization into irreducible polynomials, we can write $h_j(G_{(m,i)}) = \prod_k \tilde{h}_k^{e_{j,k}}$. Now, for every k we have that $\sum_j e_j e_{j,k} = r_k$. Therefore:

$$\begin{aligned}\sum_k \deg \tilde{h}_k (r_k - 1) &= \sum_k \deg \tilde{h}_k (\sum_j e_j e_{j,k} - 1) \\ &\geqslant \sum_k \deg \tilde{h}_k (\sum_j e_{j,k}(e_j - 1)) = \sum_j (e_j - 1) \sum_k e_{j,k} \deg \tilde{h}_k \\ &= \sum_j \deg h_j(G_{(m,i)})(e_j - 1).\end{aligned}$$

This proves the inequality. Note that equality holds if and only if, for every k, $\sum_j e_{j,k} = 1$. But this clearly holds if and only if the set $\{h_j(G_{(m,i)})\}$ is a set of square free, pairwise coprime polynomials, as claimed. $\square$

Thus, from Claim 4.17, Claim 4.16 and (17) we deduce that:

$$\begin{aligned} r &\geqslant \dim(\ker\left(M_u(fg(G_{(m,i)}), (fg(G_{(m,i)}))')\right) = \sum_k \deg \tilde{h}_k (r_k - 1) \\ &\geqslant \sum_j \deg h_j(G_{(m,i)})(e_j - 1) = \sum_j \deg h_j (e_j - 1) = r \end{aligned}$$

Here we used that $\deg h_j(G_{(m,i)}) = \deg h_j$, where the degree is with respect to u in the LHS and the with respect to x_i in the RHS. Indeed, note that when viewed as an element of $\mathbb{F}[\mathbf{x}]$, we have that $h_j \in \mathcal{P}_{\text{Factors}}(n, s^2, 2d)$, therefore the equality follows from Corollary 3.6.

Thus, it follows that we are in the equality case of Claim 4.17, and therefore the set $\{h_j(G_{(m,i)})\}$ is a set of square free, pairwise coprime polynomials. This ends the proof, as $\{h_j\}$ contains, by definition, ϕ. If ψ is also there then we are done. Otherwise, in the case where $x_i \notin \text{var}(\psi)$, the claim is trivial. $\square$

Remark 4.18. In fact, we can use the results of Section 4.1 to say something even if $d < \text{char}(\mathbb{F}) \leqslant 2d$: Consider the primitive divisors h_1, h_2 that ϕ, ψ divide. There are two cases:

- If h_1 and h_2 are not associates, then the claim follows from Lemma 4.11, but now we must take $m = \text{poly}(n, (d^2)!, s^{d^3})$.
- If $h_1 = h_2$, consider some $\tilde{h} \in \mathcal{P}(n, s, d)$ so that $h_1 = h_2 | \tilde{h}$. Then, we can replace the polynomial fg in the proof above by the polynomial $\tilde{h}$ and get the result for $m = \text{poly}(n, d!, s^d)$. Note that now $\deg_{x_i}(\tilde{h}) \leqslant d$, so indeed we only need that $\text{char}(\mathbb{F}) \geqslant d + 1$ (or $\text{char}(\mathbb{F}) = 0$).

4.2.2 Zero or Large Characteristic Algorithms

From Lemma 4.15, we are able to obtain the following generalization of Lemma 4.13.

Lemma 4.19. *Let $\mathbb{F}$ be a field of characteristic zero, or of characteristic larger than $2d$. There is a deterministic $\text{poly}(n, d!, s^d, \ell)$-time algorithm that, given blackbox access to a product, f, of ℓ polynomials in $\mathcal{P}_{Factors}(n, d, s)$ and blackbox access to an irreducible polynomial ϕ in $\mathcal{P}_{Factors}(n, s, d)$, both defined over $\mathbb{F}$, returns the multiplicity of ϕ as a factor of f. In fact, the algorithm only needs to know the values of f and ϕ on the image of the generator $G_{(m)}$ for some explicit $m = \text{poly}(n, d!, s^d)$.*

Proof. Let $m = \text{poly}(n, d!, s^d)$ as required by Lemma 4.15. For every $1 \leqslant i \leqslant n$, using Lemma 2.9, we can recover $\phi(G_{(m,i)})$ and $f(G_{(m,i)})$. Therefore, using regular division of univariate polynomials, we can find the largest k for which for every i it holds that $\phi(G_{(m,i)})^k | f(G_{(m,i)})$ as polynomials in u.

Denote with t the multiplicity of ϕ as a factor of f. We claim that $k = t$. Indeed, for every i it holds that $\phi(G_{(m,i)})^k | f(G_{(m,i)})$, so clearly $t \leqslant k$. For the inequality in the other direction, fix some i for which $x_i \in \text{var}(\phi)$, and write $f = \phi^k \cdot \prod_j \psi_j$ where ψ_j are all irreducible and non-associate to ϕ. Then, by the Lemma 4.15, $\phi(G_{(m,i)})$ is coprime to $\psi_j(G_{(m,i)})$ for any j (as polynomials in u), and therefore $\phi(G_{(m,i)}) \nmid \prod_j \psi_j(G_{(m,i)})$. Note that here we used that $x_i \in \text{var}(\phi)$, as in this case we know that $\deg_u(\phi(G_{(m,i)})) = \deg_{x_i}(\phi) \geqslant 1$ by Corollary 3.6. Thus, $\phi(G_{(m,i)})$ divides $f(G_{(m,i)})$ precisely k times as polynomials in u, and the lemma follows. □

As we did in the previous subsection, we can use the proof of Lemma 4.19 to obtain a divisibility testing result. Since the results of the present subsection apply to general irreducible factors of sparse polynomials, the resulting divisibility testing algorithms are valid not just for products of sparse polynomials, but for products of factors of sparse polynomials. In contrast, the results in Section 4.1.2 apply only to products of polynomials from $\mathcal{P}(n, s, d)$.

Theorem 4.20. *Let $\mathbb{F}$ be a field of characteristic zero or larger than $2d$. There exists a deterministic $\text{poly}(n, d!, s^d, \ell)$-time algorithm that given blackbox access to polynomials f and g, such that each is a product of ℓ polynomials in $\mathcal{P}_{Factors}(n, s, d)$, decides if $f|g$.*

Proof. Fix some $m = \text{poly}(n, d!, s^d)$, satisfying the conclusion of Lemma 4.19. For every $1 \leqslant i \leqslant n$ check if $f(G_{(m,i)})$ divides $g(G_{(m,i)})$ (as polynomials in u; throughout this argument, we view them in this way), and return that $f|g$ if all these checks passed. Note that no polynomial factorization is needed to do that; regular univariate polynomial division actually suffices. Clearly, if $f|g$ we return the correct answer. Suppose now that $f \nmid g$; let ϕ be an irreducible factor whose multiplicity as a factor of f is larger than its multiplicity as a factor of g. Then, by the proof of Lemma 4.19 it follows that for every i for which $x_i \in \text{var}(\phi)$, the highest powers k_1, k_2 for which $\phi(G_{(m,i)})^{k_1}$ divides $f(G_{(m,i)})$ and $\phi(G_{(m,i)})^{k_2}$ divides $g(G_{(m,i)})$, actually equal the multiplicity of ϕ as a factor of f and g, respectively. In particular, for these values of i it cannot be the case that $f(G_{(m,i)}) | g(G_{(m,i)})$, so we will indeed return that $f \nmid g$. □

We can now prove Theorem 1.16. We repeat its statement for convenience.

Theorem 1.16. *Let $\mathbb{F}$ be a field of characteristic zero or larger than $2d$. There exists a deterministic $\text{poly}(n, d!, s^d, \ell)$-time algorithm that, given blackbox access to two polynomials (over $\mathbb{F}$) f and g that are products of ℓ polynomials in $\mathcal{P}_{Factors}(n, s, d)$, decides if $f|g$. Moreover, if f and g are actually products of ℓ polynomials in $\mathcal{P}(n, s, d)$, there is an algorithm that works over arbitrary fields, whose running time is* $\text{poly}(n, (d^2)!, s^{d^3}, \ell)$

Proof. The proof follows from combining Theorem 4.14 and Theorem 4.20. □

Taking a closer look at the proof of Theorem 4.20, we are able to obtain Theorem 1.15 that generalizes a result of [BV25]. As mentioned at the end of the previous subsection, we currently do not know a version that works over arbitrary fields, as we need the full power of Lemma 4.15.

Theorem 1.15. *Let $\mathbb{F}$ be a field of characteristic zero or larger than $2d$. There is a deterministic $\mathrm{poly}(n, d!, s^d, \ell)$-time algorithm that, given blackbox access to a polynomial f that is a product of ℓ polynomials in $\mathcal{P}_{Factors}(n, s, d)$ (defined over $\mathbb{F}$) and a positive integer e, returns if f is a complete e-th power.*

Proof. As before, fix some large enough $m = \mathrm{poly}(n, d!, s^d)$ so that Lemma 4.15 holds. For every $1 \leqslant i \leqslant n$ compute the factorization of $f(G_{(m,i)})$, using Lemma 2.9 and Theorem 2.8. From these factorizations, it is easy to check, for every $1 \leqslant i \leqslant n$, if $f(G_{(m,i)})$ is a complete e-th power. Indeed, a polynomial is an e-th power if and only if every irreducible factor of it divides it with multiplicity divisible by e, and its leading coefficient is an e-th power as an element of $\mathbb{F}$. The former can be easily checked for $f(G_{(m,i)})$ from its factorization into irreducible factors. As for the latter, it can be directly checked for every polynomial $h \neq 0$. For instance:

- If $\mathbb{F} = \mathbb{F}_q$ for some prime power q, check if the leading coefficient α of h satisfies that $\alpha^{\frac{q-1}{e}} = 1$.
- If $\mathbb{F} = \mathbb{Q}$, use Theorem 2.6 to check if $x^e - \alpha$ has a rational root, where α is the leading coefficient of h.

We return that f is a complete e-th power if $f(G_{(m,i)})$ is a complete e-th power for every $1 \leqslant i \leqslant n$.

Clearly, if f is a complete e-th power, the algorithm returns the right answer. So, suppose that f is not a complete e-th power. Then, by the discussion above, one of the following must hold.

- There's an irreducible factor $\phi | f$ so that its multiplicity as a factor of f is some integer k, not divisible by e. Consider some value i for which $x_i \in \mathrm{var}(\phi)$, and consider some irreducible factor $\tilde{\phi}$ of $\phi(G_{(m,i)})$. We note that such an irreducible factor of positive degree in u exists, by Corollary 3.6. We claim that the multiplicity of $\tilde{\phi}$ as a factor of $f(G_{(m,i)})$ is k. Indeed, by Lemma 4.15 it is coprime to $\psi(G_{(m,i)})$ for every irreducible $\psi | f$ that is not associate to ϕ. In addition, it divides $\phi(G_{(m,i)})$ precisely once (by the "moreover" part of the lemma). Therefore $\tilde{\phi}$ is an irreducible factor of $f(G_{(m,i)})$ that divides it exactly k times, hence $f(G_{(m,i)})$ is not a complete e-th power. Thus, we will indeed return that f is not an e-th power.
- $f = \alpha g^e$ for some $g \in \mathbb{F}[\mathbf{x}]$ and $\alpha \in \mathbb{F}$, so that α is not an e-th power in $\mathbb{F}$. In this case, for every $1 \leqslant i \leqslant n$ have that $f(G_{(m,i)}) = \alpha g(G_{(m,i)})^e$ is not an e-th power as well, hence we indeed return that f is not an e-th power.

This completes the proof. □

5 A Rational Interpolation Theorem

In this section we prove our rational interpolation result.

We state our rational interpolation result. The result requires a lot of information about the functions involved, but in our applications we show how to obtain that information.

Theorem 5.1. *Suppose we have blackbox access to some rational function of the form $\frac{af}{b}$, defined over a field $\mathbb{F}$, where:*

(1) a is an (n, s, d)-sparse polynomial.

(2) We have blackbox access f, which is a product of ℓ polynomials in $\mathcal{P}_{Factors}(n, s, d)$.

(3) We have a set $\mathcal{F}$ of s-sparse irreducible factors of f, call them ϕ_i.

(4) b *is an* s*-sparse factor of* f *whose irreducible factors are all elements of* $\mathcal{F}$.

(5) a *and* b *are coprime.*

Then, there is an algorithm that recovers a *and* b *from evaluating* $\frac{af}{b}$ *on the image of the generator* $G_{(m)}$ *for some explicit* m *(depending on* $\mathrm{char}(\mathbb{F})$*), such that the running time is* $\mathrm{poly}(m, \ell)$*, where:*

- *If* $\mathrm{char}(\mathbb{F}) = 0$ *or* $\mathrm{char}(\mathbb{F}) > 2d$*, one can take* $m = \mathrm{poly}(n, d!, s^d)$.
- *Over arbitrary fields,* $m = \mathrm{poly}(n, (d^2)!, s^{d^3})$*. If we further assume that all elements of* $\mathcal{F}$ *are multi-quadratic, then we can take* $m = \mathrm{poly}(n, (d^2)!, s^{d^2})$.

Proof. Assume first that the characteristic of the field is zero or larger than $2d$. Fix some $m = \mathrm{poly}(n, d!, s^d)$ for which the conclusion of Lemma 4.19 holds. We start by recovering b. As all irreducible factors of b belong to $\mathcal{F}$, it suffices to understand the multiplicity of each $\phi \in \mathcal{F}$ as a factor of b. Fix such an irreducible factor $\phi \in \mathcal{F}$. Using Lemma 4.19 we can compute the multiplicity of ϕ as a factor of f, which we denote by t.

As a and b are coprime, it follows that, for any k, $\phi^k | b$ if and only if $\phi^{t-k+1} \nmid \frac{af}{b}$. Therefore, it is sufficient to understand the multiplicity of ϕ as a factor of $\frac{af}{b}$. As f is a product of factors of s-sparse polynomials, $b|f$, and a is s-sparse, $\frac{af}{b}$ is also a product of factors of s-sparse polynomials. We can therefore apply Lemma 4.19 to learn the multiplicity of ϕ as a factor of $\frac{af}{b}$. We conclude that we can learn the factorization of b into irreducible polynomials. In particular, we can explicitly recover b using Lemma 3.4. To learn a, we can simply compute $a(G_{(m)}) = f(G_{(m)})/b(G_{(m)})$ and apply Lemma 3.4.

For fields of small characteristic, and the multiquadratic case, we repeat the same argument, replacing the use of Lemma 4.19 by the "moreover" part of Lemma 4.13. □

We next state a slight generalization of the theorem above that we will later use.

Theorem 5.2. *Suppose we have blackbox access to a set of rational functions, defined over a field* $\mathbb{F}$*, of the form* $\frac{a_j f^j}{b}$*, for* $1 \leqslant j \leqslant D$*, where:*

(1) a_j *is an* (n, s, d)*-sparse polynomial for every* $1 \leqslant j \leqslant D$.

(2) f *is a product of* ℓ *polynomials in* $\mathcal{P}_{\mathit{Factors}}(n, s, d)$ *to which we have blackbox access.*

(3) We have a set $\mathcal{F}$ *of* s*-sparse irreducible factors of* f*, call them* ϕ_i.

(4) b *is an* s*-sparse factor of* f *whose irreducible factors are all elements of* $\mathcal{F}$.

(5) $\gcd(a_1, \ldots, a_D, b) = 1$.

Then, there is an algorithm that recovers the a_j*-s and* b *from evaluations of* $\frac{a_j f^j}{b}$ *on the image of the generator* $G_{(m)}$*, for some explicit* m *(depending on* $\mathrm{char}(\mathbb{F})$*), such that the running time is* $\mathrm{poly}(m, \ell, D)$*, where:*

- *If* $\mathrm{char}(\mathbb{F}) = 0$ *or* $\mathrm{char}(\mathbb{F}) > 2d$*, one can take* $m = \mathrm{poly}(n, d!, s^d, D)$.
- *For arbitrary fields we have* $m = \mathrm{poly}(n, (d^2)!, s^{d^3}, D)$*. If we further assume that all elements of* $\mathcal{F}$ *are multiquadratic, then we can take* $m = \mathrm{poly}(n, (d^2)!, s^{d^2}, D)$.

Proof. The proof goes along the same lines as the proof of Theorem 5.1. The only difference is that now, when recovering b, it is does not necessarily hold that $\gcd(a_j, b) = 1$. To overcome this obstacle, observe that for every $\phi \in \mathcal{F}$ of multiplicity t as a factor of f, it holds that, for every $k \geqslant 1$, $\phi^k | b$ if and only if there is $1 \leqslant j \leqslant D$ for which $\phi^{tj-k+1} \nmid \frac{a_j f^j}{b}$. This is true by the same reasoning as before. Indeed, there is some $1 \leqslant j \leqslant D$ for which $\phi \nmid a_j$, as otherwise $\phi | \gcd(a_1, \ldots, a_D, b)$. The algorithm then proceeds in the exact same way as before. □

6 Meta-Algorithm

In this section we present a meta-algorithm that will be used, with some variations, in the proofs of Theorems 1.9, 1.10, 1.11, 1.12, 1.13 and 1.14, which state our algorithmic results. Since Theorem 1.10 follows as a corollary of Theorem 1.11, we will not discuss it here.

Each of these algorithms follows the outline of the meta-algorithm, and in their proofs we will refer to steps of the meta-algorithm and describe how they are implemented in the relevant setting.

(1) **Setting:**

We assume that we want to learn properties of a polynomial f such that f is either

(1) (n, s, d)-sparse (Theorem 1.9, Theorem 1.12)

(2) a product of polynomials in $\mathcal{P}_{\text{Factors}}(n, s, d)$ (Theorem 1.11, Theorem 1.13, Theorem 1.14)

and we want to find a list $\mathcal{F}(f)$ of divisors of f having certain properties. For every variable x_i, the polynomial $f|_{x_i=0}$ is also of the same type as f. This is clearly true when f is (n, s, d)-sparse. When f is a product of polynomials in $\mathcal{P}_{\text{Factors}}(n, s, d)$, this follows from Lemma 6.2, whose formulation and proof are given at the end of this section.

In what follows, for the sake of convenience, we assume that f is an $n+1$-variate polynomial and is an element of $\mathbb{F}[x_0, \mathbf{x}]$.

(2) **Preprocessing:**

(1) Remove from f its largest monomial divisor (using Corollary 3.8).

(2) Normalize f to get $\tilde{f}(y, \mathbf{x}) = f(yf_0, x_1, \ldots, x_n)/f_0$. Set $\hat{f} = f(y, G)$ for an appropriate generator G. Observe that $\tilde{f}(y, \mathbf{x})$ is reverse monic with respect to y, and hence so is $\hat{f}$. By Claim 2.3 it then follows that $\hat{f}$ has at most $\deg_y(\hat{f}) \leqslant d$ irreducible factors if f is (n, s, d)-sparse, and at most $\deg_y(\hat{f}) \leqslant \ell d$ if f is a product of ℓ polynomials in $\mathcal{P}_{\text{Factors}}(n, s, d)$.

(3) Use Lemma 2.9 to interpolate $\hat{f}$ as a polynomial in $O(1)$ many variables. Then apply Theorem 2.8 to obtain its factorization into irreducible factors.

(3) **Recursive call to solve for f_0:** Recursively solve the problem for $f_0 = f|_{x_0=0}$, obtaining a list $\mathcal{F}(f_0)$ of divisors not divisible by a monomial or irreducible factors, according to the problem.

(4) **Generating a list of candidate solutions:**

By Claim 2.5, every divisor $h = \sum_{i=0}^{d} c_i x_0^i$ of f gives rise to a divisor $\sum_{i=0}^{d} \frac{c_i}{c_0} f_0^i y^i$ of $\tilde{f}$ and hence to a divisor $\hat{h} = \sum_{i=0}^{d} \frac{c_i(G)}{c_0(G)} f_0^i y^i$ of $\hat{f}$. We would like to get access to $c_i(G)$ in order to interpolate them, and through that obtain access to h. This leads us to the rational interpolation problem, where we have some information about c_0 (or even c_0 itself) coming from $\mathcal{F}(f_0)$.

(1) In Theorem 1.9, Theorem 1.11 we will have that $c_0 \in \mathcal{F}(f_0)$. In Theorem 1.12 c_0 is a product of elements of $\mathcal{F}(f_0)$, but we have enough time to enumerate all relevant products so we can pretend that $c_0 \in \mathcal{F}(f_0)$.

In fact, the above statement is not entirely accurate. The recursive call returns the set $\mathcal{F}(f_0)$ of all divisors of f_0 that are not divisible by a monomial, together with the largest monomial dividing f_0. It might be the case that c_0 is a divisor of f_0 that *is* divisible by a monomial. In this case, c_0 will not belong to $\mathcal{F}(f_0)$. However, c_0 does lie in $\mathcal{F}(f_0)$ *up to a multiplication by a monomial*. As we shall see, this suffices for our purposes.

(2) In Theorem 1.13 and Theorem 1.14 the situation is a bit different. Here, c_0 may be a product of elements from $\mathcal{F}(f_0)$, and we cannot go over all possible ways to obtain c_0. However, in this case we will have that $\mathcal{F}(f_0)$ contains all the irreducible factors of c_0, which is what Theorem 5.2 requires.

Accordingly, for each factor $\hat{h} = \sum_{i=0}^{d} e_i y^i$ we compose a list $\mathcal{F}_{\hat{h}}$ as follows:

(1) When we have a list containing (up to multiplication by a monomial) the correct c_0 (Theorem 1.9, Theorem 1.11, Theorem 1.12), we iterate over all possible candidates c_0 in that list. For each such candidate, we interpolate the polynomials of the form $\left((x_1 \cdots x_n)^d c_0/f_0^i\right)(G)\, e_i$ in order to recover the coefficients c_i. We then add to $\mathcal{F}_{\hat{h}}$ those polynomials $\sum_{i=0}^{d} c_i x_0^i$ that are valid candidates for the problem, after dividing by their largest monomial divisor:

   (1) In Theorem 1.9, Theorem 1.11 we keep all (n, s, d)-sparse polynomials.

   (2) In Theorem 1.12 we keep all (n, S, d)-sparse polynomials, where $S = s^{O(d^2 \log n)}$ is the upper bound on sparsity of factors of (n, s, d)-sparse polynomials obtained in Theorem 2.14.

   The term $(x_1 \cdots x_n)^d$ is needed to deal with the fact that c_0 is in $\mathcal{F}(f_0)$ only up to division by a monomial. We shall explain this point in full detail in the proof of Theorem 1.9.

(2) If we only have a list of irreducible factors of c_0 (Theorem 1.13 and Theorem 1.14), apply Theorem 5.2 on the coefficients of $\hat{h}$ to recover potential c_0 and $c_1, \ldots, c_d$. We put the resulting polynomial $\sum_{i=0}^{d} c_i x_0^i$ in the list $\mathcal{F}_{\hat{h}}$ if:

   (1) For Theorem 1.13 it is (n, S, d)-sparse.

   (2) For Theorem 1.14, it is multiquadratic and (n, s, d)-sparse.

   We note that, in both cases, we set $\mathcal{F}_{\hat{h}} = \mathcal{F}(f_0)$ for $\hat{h} = 1$.

(5) **Pruning the list:**

Depending on the application we either have to keep divisors or irreducible divisors. We do that in two steps - we first find the divisors in the list of candidates, and then, if needed, identify the irreducible polynomials among them. The implementation of the two steps vary in the different algorithms; we give here a solution that works for all cases, though not always optimal, and provide a better implementation in the proofs of the algorithms.

In all cases we apply Theorem 1.16 to remove from the list those polynomials that do not divide f. This already suffices for Theorem 1.9 and Theorem 1.11.

To obtain irreducible divisors as required in Theorem 1.12, Theorem 1.13, Theorem 1.14, we apply Theorem 1.16 to each two polynomials g, h in the list of divisors to check whether g divides h, and keep only those divisors that are not divisible by any other divisor.

(6) **Computing multiplicities:**

To compute multiplicities, where needed, we apply Lemma 4.19 or Lemma 4.13, depending on the characteristic. Here as well there are better solutions in some cases; these are described in the proofs of the algorithms.

The analysis of the running time of the algorithm depends on the exact setting. However, in all settings we perform exactly one recursive call so the overall complexity is determined by the cost of the different steps of the algorithm, and it does not blow up because of the recursive call.

Remark 6.1. The described meta-algorithm provides the outline that all our algorithms follow. It is, however, *not* an algorithm that *generalizes* all of them. It is possible to formulate two concrete algorithms which all our results follow - one generalizing Theorems 1.9, 1.11 and 1.12, and one generalizing Theorems 1.13 and 1.14. However, the precise formulations of these general algorithms are rather complicated and not particularly intuitive; hence, we have chosen not to include them in the present paper.

We close this section by proving the following lemma.

Lemma 6.2. *Let* f *be a product of* ℓ *polynomials in* $\mathcal{P}_{Factors}(n+1, s, d) \subset \mathbb{F}[x_0, \mathbf{x}]$. *Suppose further that* $x_0 \nmid f$, *and let* f_0 *be the free term of* f *as a polynomial in* x_0. *Then,* f_0 *is a product of* ℓ *elements of* $\mathcal{P}_{Factors}(n, s, d) \subset \mathbb{F}[\mathbf{x}]$.

Proof. Write $f = \prod_{i=1}^{\ell} g_i$, where $g_i \in \mathcal{P}_{\text{Factors}}(n+1, s, d)$. Denote further $g_i = \sum_{j=0}^{d} g_{i,j} x_0^j$, where $g_{i,j} \in \mathbb{F}[\mathbf{x}]$, and let $\hat{g}_i$ be an $(n+1, s, d)$-sparse polynomial that g_i divides. First, note that $f_0 = \prod_{i=1}^{\ell} g_{i,0}$. Next, observe that $g_{i,0}$ divides the trailing monomial of $\hat{g}_i$ as a polynomial in x_0. As this trailing monomial is clearly (n, s, d)-sparse, it follows that $g_{i,0} \in \mathcal{P}_{\text{Factors}}(n, s, d)$. Consequently, f_0 is indeed a product of ℓ polynomials in $\mathcal{P}_{\text{Factors}}(n, s, d)$. □

7 Finding Multiquadratic Factors

In this section we prove Theorem 1.14.

Theorem 1.14. *Let* $\mathbb{F}$ *be a field of characteristic zero or larger than* $2d$. *There is a deterministic* $\text{poly}(n, d!, s^d, \ell)$*-time algorithm that, given blackbox access to a product* f *of* ℓ *elements of* $\mathcal{P}_{Factors}(n, s, d)$ *defined over* $\mathbb{F}$, *returns all the* s*-sparse multiquadratic irreducible factors of* f. *In particular, the algorithm returns all the multilinear factors of* f *(as these are* s*-sparse by Lemma 2.12). Moreover, there exists such an algorithm that works over arbitrary fields, whose running time is* $\text{poly}(n, (d^2)!, s^{d^2}, \ell)$.

Proof. We follow the outline of the meta-algorithm described in Section 6. As this is the first time we implement the meta-algorithm, we give the full details of the implementation. We describe the algorithm for the case $\text{char}(\mathbb{F}) = 0$ or $\text{char}(\mathbb{F}) > 2d$ and then explain which components change when we work over arbitrary fields.

To simplify the presentation we shall assume that the input f is an element of $\mathbb{F}[x_0, \mathbf{x}]$, and is a product of ℓ elements of $\mathcal{P}_{\text{Factors}}(n+1, s, d)$. As described in Section 6(2), we begin with a preprocessing step.

Preprocessing: By Corollary 3.8 we may assume that f is not divisible by any monomial. In particular, we may assume that $x_0 \nmid f$. Write $f = \sum_{i=0}^{\ell d} f_i x_0^i$, where $f_i \in \mathbb{F}[\mathbf{x}]$. Let $\tilde{f}$ be the normalization of f as in Definition 2.4. By Claim 2.5 $\tilde{f}$ is reverse monic and we have blackbox access to it. Denote

$$\hat{f} := \tilde{f}(y, G),$$

for $G = G_{(m)}$, with m as given in Theorem 5.2. Using Lemma 2.9 and Theorem 2.8, obtain the factorization of $\hat{f}$. Note that, as $\tilde{f}$ is reversed-monic with respect to y, so is $\hat{f}$. Thus, by Claim 2.3, $\hat{f}$ has at most ℓd irreducible factors. Consequently, it has at most $O((\ell d)^2)$ (not necessarily irreducible) factors of degree at most two in y overall. We note that $\hat{f}$ is a polynomial in $O(1)$ variables of degree $\text{poly}(m, \ell)$, therefore the running time of this step is at most $\text{poly}(m, \ell)$.

Recursive call: We would like to first solve the problem for f_0. By Lemma 6.2, f_0 is also a product of ℓ elements of $\mathcal{P}_{\text{Factors}}(n, s, d)$. Hence, we can, by applying one recursive call, find the set of all multiquadratic, s-irreducible factors of it. Denote this set by $\mathcal{F}$.

Generating a list of candidate solutions: Recall that Claim 2.5 shows that for every irreducible factor $\phi = b + a_1x_0 + a_2x_0^2$ of f, there is a corresponding irreducible factor $\tilde{\phi} = 1 + \frac{a_1 f_0}{b}y + \frac{a_2 f_0^2}{b}y^2$ of $\tilde{f}$, which corresponds to some (not necessarily irreducible) quadratic factor of $\hat{f}$, which we denote by $\hat{\phi} = \tilde{\phi}(y, G)$. Our plan is to learn the (a_1, a_2, b)'s, corresponding to the different factors, from the factorization of from $\hat{f}$ and $\mathcal{F}$.

For that, we run over all factors $\hat{\phi}$ of $\hat{f}$, that have degree at most two as polynomials in y. For each such $\hat{\phi}$ we run the algorithm of Theorem 5.2 to try and recover corresponding a_1, a_2 and b (or (a_1, b) if $\deg_y(\hat{\phi}) = 1$). If $b + a_1x_0 + a_2x_0^2$ is an s-sparse multiquadratic polynomial then we add it to the list of candidate factors.

Claim 7.1. *In the notation above, assume that $\phi = b + a_1x_0 + a_2x_0^2$ corresponds to a multiquadratic (or multilinear) irreducible factor of f. Then the procedure described above will add ϕ to the list of candidate multiquadratic factors when executed on $\hat{\phi}$.*

Proof. We need to show that we have all the required information to run Theorem 5.2 successfully. Indeed:

(1) As ϕ is s-sparse, a_1, a_2 are s-sparse as well.

(2) f_0 is a product of ℓ elements of $\mathcal{P}_{\text{Factors}}(n, s, d)$ to which we have a blackbox access.

(3) By the recursive call, we have a set $\mathcal{F}$ containing all multiquadratic, s-sparse irreducible factors of f_0.

(4) As ϕ is multiquadratic then so is b. Therefore, by Lemma 2.13, all irreducible factors of b are s-sparse and multiquadratic as well. Since $b | f_0$, these factors must be elements of $\mathcal{F}$.

(5) From the irreducibility of ϕ we obtain $\gcd(a_1, a_2, b) = 1$.

Thus, Theorem 5.2 will return a_1, a_2 and b. □

Therefore, by running over all factors of $\hat{f}$ of degree at most 2 in y, we are guaranteed to have included all irreducible multiquadratic factors of f in the list of candidates. Since f may have multiquadratic factors that do not depend on x_0, we also add the elements of $\mathcal{F}$ to the list of candidates for factors of f.

Our next step is finding all the "true" factors. Indeed, our list of candidates may contain "fake" candidates as well. For example, these may come from a successful interpolation process to some factor ϕ that has degree higher than two in some variable x_i, a property that was not maintained in the composition, factorization and interpolation process.

Pruning the list: We now have a set of $O((\ell d)^2)$ candidates for divisors of f, all of which are multiquadratic and s-sparse, and we are left with the problem of identifying the real irreducible divisors among them and computing their multiplicities.

By running the algorithm of Theorem 1.16 we decide, for every candidate $\phi \in \mathcal{F}$, if it really divides f. We now have a set of candidates, all of which are multiquadratic and s-sparse, that divide f, and we have to identify the irreducible ones.

Lemma 2.13 implies that since each factor ϕ is multiquadratic and s-sparse, then so are all its factors. In particular, it is divisible by some multiquadratic irreducible factor of f, which must be in the list. Therefore, for every pair ϕ, ψ, we check if $\psi | \phi$ (using Theorem 1.16). The candidates ϕ for which there are no other candidates that divide them are the desired irreducible factors.

Computing the multiplicities: We have found all the multiquadratic, s-sparse irreducible factors of f. To compute their multiplicities as factors of f we invoke Lemma 4.19.

Arbitrary fields: For arbitrary fields we let m be as in Theorem 5.2 for the case of multiquadratic polynomials. Instead of using Theorem 1.16 we apply the ”moreover” part of Theorem 4.14. Similarly, instead of Lemma 4.19 we rely on the ”moreover” part of Lemma 4.13.

Running time: Note that for fields of characteristic zero or larger than $2d$, each step runs in time $\mathrm{poly}(n, d!, s^d, \ell)$. This is the running time guaranteed by Theorem 5.2, Theorem 1.16, and Lemma 4.19. Since we have one recursive call the total running time is also $\mathrm{poly}(n, d!, s^d, \ell)$. For arbitrary fields, the running time is $\mathrm{poly}(n, (d^2)!, s^{d^2}, \ell)$, as this is what we obtain from Theorem 5.2, the ”moreover” part of Theorem 4.14 and the ”moreover” part of Lemma 4.13.

This completes the description, and analysis, of the algorithm. $\square$

8 Computing all Sparse Divisors of a Sparse Polynomial

In this section, we analyze the set of sparse divisors of a sparse polynomial of bounded individual degree, and give a polynomial time algorithm for constructing it. We then apply our methods to give a partial solution to the problem raised in [DST24] regarding the recovery of sparse irreducible polynomials of bounded individual degree from blackbox access to their product (Question 1.4).

8.1 Number of Divisors of Sparse Polynomials of Bounded Individual Degree

We first prove an upper bound on the number of divisors (that are not divisible by any monomial) of sparse polynomials with bounded individual degree.

Theorem 1.8 (Divisor bound for (n, s, d)-sparse polynomials). *Let $f \in \mathcal{P}(n, s, d)$. The number of non-monomial irreducible factors of f, counted with multiplicities, is at most $d \log s$. In particular, the number of divisors of an s-sparse polynomial f with bounded individual degree d, that are not divisible by any monomial, is at most $2^{d \log s} = s^d$.*

Proof. By dividing by a monomial, we may assume that f is not divisible by any x_i. We choose a sequence of variables in the following iterative way. First, choose some $x_{i_1} \in \mathrm{var}(f)$. Suppose we have already chosen $x_{i_1}, \ldots, x_{i_k}$, and consider the factorization of f into irreducible polynomials. If there is an irreducible factor ϕ that contains none of the variables chosen thus far, we choose some $x_{i_{k+1}} \in \mathrm{var}(\phi)$. If no such factor exists, then the process terminates.

Assume that in the process we chose k variables (in sequence), which we denote without loss of generality by $x_1, x_2, \ldots, x_k$ (i.e., we assume that $i_1 = 1$ etc.). The sequence of chosen variables naturally corresponds to a sequence of *multisets* of irreducible factors of f. The first multiset I_1 is the multiset of irreducible factors that depend on x_1 (counted with multiplicities); the second multiset I_2 consists of irreducible factors that do not depend on x_1 but depend on x_2; the j-th multiset I_j is the multiset of irreducible factors that do not depend on $x_1, \ldots, x_{j-1}$ but depend on x_j. It is easy to see that $\{I_j\}_{1 \leqslant j \leqslant k}$ form a partition of the multiset of irreducible factors of f.

As f is of bounded individual degree d and every factor of I_j depends on x_j, it must be the case that $|I_j| \leqslant d$ as a multiset. Thus, the total number of irreducible factors of f, counted with multiplicities, is at most kd. Therefore, to prove our theorem, it is sufficient to show that $k \leqslant \log s$.

To this end, let h_j be the product of all the polynomials in I_j, so that $f = \prod_{j=1}^{k} h_j$.

Claim 8.1. *Denote $f_j = \prod_{t=k-j+1}^{k} h_t$. Then, $2^j \leqslant |V(P_{f_j})|$, where P_{f_j} is the Newton polytope corresponding to f_j and $V(P_{f_j})$ is the set of its vertices (as defined in Definition 2.16).*

We first conclude the proof of the theorem and then prove the claim. By the claim, it follows that $2^k \leqslant |V(P_{f_k})| = |V(P_f)| \leqslant s$, and thus $k \leqslant \log s$, as we wanted to show. Note that we used here that f is s-sparse, to get, using Lemma 2.17, that $|V(P_f)| \leqslant s$. Note that as $d \log s$ is an upper bound on the number of factors, including multiplicities, the number of divisors of f is at most $2^{d \log s} = s^d$. □

Proof of Claim 8.1. The proof is by induction of j. For $j = 1$, the claim follows from the fact that $f_1 = h_k$ is not a monomial (as we removed all monomial divisors of f).

For the inductive step, suppose that $2^j \leqslant |V(P_{f_j})|$. We will show that $2^{j+1} \leqslant |V(P_{f_{j+1}})|$. To this end, express h_{k-j} as a polynomial in x_{k-j}. That is, $h_{k-j} = \sum_{i=0}^{d_{k-j}} g_i x_{k-j}^i$ where $g_i \in \mathbb{F}[x_{k-j+1}, \dots, x_n]$. By construction, $x_{k-j} \in \operatorname{var}(h_{k-j})$, so $d_{k-j} = \deg_{x_{k-j}}(h_{k-j}) > 0$. Moreover, since $x_{k-j} \nmid h_{k-j}$, it must be the case that $g_0 \neq 0$. Observe that $f_{j+1} = f_j h_{k-j} = \sum_{i=0}^{d_{k-j}} f_j g_i x_{k-j}^i$, where $f_j g_i \in \mathbb{F}[x_{k-j+1}, \dots, x_n]$. We claim the following.

(1) Any vertex $u \in V(P_{f_j g_0}) \subset \mathbb{R}^{n+j-k}$ corresponds to a vertex $(0, u) \in \mathbb{R}^{n+j-k+1}$ of $V(P_{f_{j+1}})$.

(2) Any vertex $v \in V(P_{f_j g_{d_{k-j}}}) \subset \mathbb{R}^{n+j-k}$ corresponds to a vertex $(d_{k-j}, v) \in \mathbb{R}^{n+j-k+1}$ of $V(P_{f_{j+1}})$.

(3) We have $|V(P_{f_j g_0})| \geqslant 2^j$ and $|V(P_{f_j g_{d_{k-j}}})| \geqslant 2^j$.

The first and second claims are clear from the fact that the described sets of vertices are precisely the sets of vertices with the lowest/largest value on the first coordinate (that is, the coordinate that corresponds to the exponent of x_{k-j}), respectively. For the third claim, note that by Lemma 2.17 and the induction hypothesis we get

$$|V(P_{f_j g_0})| \geqslant |V(P_{f_j})| \geqslant 2^j, \quad \text{and} \quad |V(P_{f_j g_{d_{k-j}}})| \geqslant |V(P_{f_j})| \geqslant 2^j .$$

Therefore, we showed that overall there are at least 2^{j+1} elements in $V(P_{f_{j+1}})$, completing the proof of inductive step. □

Remark 8.2. The above theorem is easily seen to be tight, for example by considering $f(x_1, \dots, x_n) = \prod_{i=1}^{\log s} ({x_i}^d - 1)$ over $\mathbb{C}$.

As corollary we obtain the following upper bound on the number of sparse divisors of a product of factors of (n, s, d)-polynomials.

Corollary 8.3. *Let f be a product of ℓ elements of $\mathcal{P}_{Factors}(n, s, d)$. The number of (n, s, d)-sparse divisors of f that are not divisible by a monomial is at most*

$$\sum_{i=0}^{d \log s} \binom{\ell d \log s}{i} \leqslant s^{d(2+\log \ell)}.$$

Proof. We assume without loss of generality that f is not divisible by any monomial.

Suppose that $f = \prod_{i=1}^{\ell} \phi_i$ where $\phi_i \in \mathcal{P}_{\text{Factors}}(n,s,d)$, and denote by g_i an (n,s,d)-sparse polynomial that ϕ_i divides.

By Theorem 1.8, each g_i has at most $d \log s$ irreducible factors (we repeat a factor several times according to its multiplicity), so this must also be the case for ϕ_i. Therefore, f has at most $\ell d \log s$ irreducible factors, which we denote by $\{\psi_j\}$. We note that this is a multiset where factors may appear several times according to their multiplicity in the factorization of f.

We claim that any (n,s,d)-sparse divisor of f is a product of at most $d \log s$ such factors ψ_j. Indeed, any divisor of f is a product of a subset of the ψ_js, and by Theorem 1.8, any such s-sparse divisor cannot be a product of more than $d \log s$ of the ψ_js. Thus, the number of s-sparse factors of bounded individual degree d of f is bounded from above by

$$\sum_{i=0}^{d \log s} \binom{\ell d \log s}{i}. \tag{18}$$

Now, if $\ell = 1$, (18) equals $2^{d \log s} = s^d$. If $\ell \geqslant 2$, we have that $\binom{\ell d \log s}{d \log s}$ is the largest summand in (18), and it can be easily seen that our expression (18) is bounded above by $d \log s \binom{\ell d \log s}{d \log s}$. Either way, we get that the number of relevant factors is upper bounded by

$$\begin{aligned} \max\left\{ s^d, d \log s \binom{\ell d \log s}{d \log s} \right\} &\leqslant \max\left\{ s^d, d \log s \left(\frac{e \ell d \log s}{d \log s} \right)^{d \log s} \right\} \\ &= \max\left\{ s^d, d \log s (e\ell)^{d \log s} \right\} \\ &= (d \log s) s^{d(\log \ell + \log e)} \\ &\leqslant s^{d(2+\log \ell)}, \end{aligned}$$

where the last inequality follows from the fact that for every $c > 0$ and $x \geqslant 1$ it holds that $(ce) \ln x \leqslant x^c$, which can be verified by noting that $\frac{\ln x}{x^c}$ is maximized at $x = e^{1/c}$. Applying this inequality for $c = d(2 - \log e)$ and using the fact that $e \ln 2(2 - \log e) > 1$ we obtain the claimed result. $\square$

8.2 Finding Sparse Divisors of Sparse Polynomials with Bounded Individual Degree

In this section we prove Theorem 1.9. We recall its statement.

Theorem 1.9 (Finding all sparse divisors of (n,s,d)-sparse polynomials). *There is a deterministic, $\text{poly}(n, d!, s^d)$-time algorithm which, given $f \in \mathcal{P}(n,s,d)$, outputs all the s-sparse divisors of f that are not divisible by any monomial, as well as the monomial of maximal degree dividing f.*

Proof. As in the proof of Theorem 1.14, we follow the steps of the meta-algorithm of Section 6, first studying fields such that $\text{char}(\mathbb{F}) = 0$ or $\text{char}(\mathbb{F}) > 2d$, and then arbitrary fields. For the sake of presentation, we shall assume that $f \in \mathcal{P}(n+1,s,d) \subset \mathbb{F}[x_0, \mathbf{x}]$.

Preprocessing: This step is performed as in the proof of Theorem 1.14. We first learn the monomial divisors of f. We thus assume $x_0 \nmid f$. Write $f = \sum_{i=0}^{d} f_i x_0^i$, where $f_i \in \mathbb{F}[\mathbf{x}]$, and note that $f_0 \neq 0$ by our assumption. Furthermore, $f_0 \in \mathcal{P}(n,s,d)$. Let $\tilde{f}$ denote the normalization of f and $\hat{f} := f(y, G)$, for $G = G_{(m)}$ with $m = (2nsd)^2$, as given by Lemma 3.4 for $(n,s,2d)$-sparse polynomials. Using Lemma 2.9 and Theorem 2.8, obtain the factorization of $\hat{f} = f(y, G)$ into irreducible polynomials. As $\tilde{f}$ is reversed-monic with respect to y, so is $\hat{f}$. By Claim 2.3 it follows that $\hat{f}$ has at most d irreducible factors, and therefore at most 2^d divisors overall. Since $\hat{f}$ has $O(1)$ variables and degree $\mathrm{poly}(m)$, this step runs in time $\mathrm{poly}(m)$.

Recursive call: Since f_0 is clearly an (n, s, d)-sparse polynomial, we can, by applying one recursive call, find the set of all s-sparse divisors of it. Denote this set by $\mathcal{F}(f_0)$. By Theorem 1.8, we have that $|\mathcal{F}(f_0)| \leqslant s^d$.

Generating, and pruning, a list of candidate solutions: By Claim 2.5, every divisor $h = \sum_{i=0}^{d} c_i x_0^i$ of f corresponds to a divisor $\tilde{h} = \sum_{i=0}^{d} \frac{c_i}{c_0} {f_0}^i y^i$ of $\tilde{f}$, and therefore to a divisor

$$\hat{h} := \tilde{h}(y, G) = \sum_{i=0}^{d} \frac{c_i(G)}{c_0(G)} f_0(G)^i y^i$$

of $\hat{f}$. As before, we run over all divisors $\hat{h} = \sum_{i=0}^{d} e_i y^i$ of $\hat{f}$. For each such divisor we produce a list $\mathcal{F}_{\hat{h}}$ of at most $|\mathcal{F}(f_0)|$ potential candidates for s-sparse divisors of f as follows: For each $c \in \mathcal{F}(f_0)$, let $\hat{e}_i = \left((x_1 \cdot \ldots \cdot x_n)^d c\right)(G) \frac{e_i}{f_0^i(G)}$. Using Lemma 3.4 we sparse interpolate $\hat{e}_0, \hat{e}_1, \ldots, \hat{e}_d$ to $(n, s, 2d)$-polynomials $\tilde{e}_0, \tilde{e}_1, \ldots, \tilde{e}_d$. If the interpolation failed, for any reason, then we move to the next $c \in \mathcal{F}(f_0)$. If the interpolation succeeded then we compute the largest monomial M that divides each of the polynomials $\tilde{e}_0, \tilde{e}_1, \ldots \tilde{e}_d$. If $h = \frac{\sum_{i=0}^{d} \tilde{e}_i x_0^i}{M}$ is an (n, s, d)-sparse polynomial then we run Theorem 1.16 to check whether it divides f, and in that case we add it to $\mathcal{F}_{\hat{h}}$.

We let $\mathcal{F}'(f_0)$ be those elements of $\mathcal{F}(f_0)$ that divide f (again, using Theorem 1.16 to check that).

We return the set $\mathcal{F}(f) = \mathcal{F}'(f_0) \cup \left(\cup_{\hat{h}|\hat{f}} \mathcal{F}_{\hat{h}}\right)$, which is the union of all the sets produced by the algorithm.

Claim 8.4. *$\mathcal{F}(f)$ is the set of all the divisors $h = \sum_{i=0}^{d} c_i x_0^i$ of f.*

Proof. Since $\mathcal{F}(f)$ only contains divisors of f, we need to show that it contains all the divisors of f.

Recall that every divisor $h = \sum_{i=0}^{d} c_i x_0^i$ of f gives rise to a divisor $\hat{h} = \sum_{i=0}^{d} \frac{c_i(G)}{c_0(G)} f_0^i(G) y^i$ of $\hat{f}$. Moreover, c_0 is an (n, s, d)-sparse polynomial satisfying $c_0 \mid f_0$. Therefore, for if M is the largest monomial dividing c_0 then $c_0 = Mc$ for some $c \in \mathcal{F}$.

Denote $e_i = \frac{c_i(G)}{c_0(G)} f_0^i(G)$, so that $\hat{h} = \sum_{i=0}^{d} e_i y^i$. Observe that for $\tilde{M} = (x_1 \cdot \ldots \cdot x_n)^d / M$ we have

$$\hat{e}_i = \left((x_1 \cdot \ldots \cdot x_n)^d c\right)(G) \frac{e_i}{f_0^i(G)} = \tilde{M}(G) \cdot \frac{c_0(G) \frac{c_i(G)}{c_0(G)} f_0^i(G)}{f_0^i(G)} = \tilde{M}(G) c_i(G).$$

Since c_i is an (n, s, d)-sparse polynomial and $\tilde{M}$ is a monomial of individual degree bounded by d, $\tilde{M}c_i$ is an $(n, s, 2d)$-sparse polynomial. Therefore, Lemma 3.4 will return the polynomial $\tilde{M}c_i$ when run on $\hat{e}_i$. Consequently, we will add $h = \sum_{i=0}^{d} c_i x_0^i$ to $\mathcal{F}_{\hat{h}}$ and hence to $\mathcal{F}(f)$. □

Arbitrary fields: The assumption on the field is used only to invoke Theorem 1.16 for pairs of $(n+1, s, d)$-sparse polynomials. [Vol17, Lemma 42] gives a deterministic algorithm for divisibility testing of pairs of $(n+1, s, d)$-sparse polynomials whose running time is $\mathrm{poly}(n, d!, s^d)$. Therefore, by replacing Theorem 1.16 with this result, we can remove the assumption on the field.

Running time: As before, the preprocessing step requires $\mathrm{poly}(nsd)$ time. Lemma 3.4 requires $\mathrm{poly}(nsd)$ time. Theorem 1.16 and [Vol17, Lemma 42] take $\mathrm{poly}(n, d!, s^d)$ time. Since we make one recursive call the total running time is $\mathrm{poly}(n, d!, s^d)$. □

Remark 8.5. The algorithm above solves another rational interpolation problem. Suppose we have blackbox access to $\frac{af_0}{b}$, where f_0 is a known s-sparse polynomial of bounded individual degree d, and a, b are unknown, relatively prime, (n, s, d)-sparse polynomials. Suppose further that $b|f_0$. Then it is possible to recover a, b in $\mathrm{poly}(n, d!, s^d)$-time. This algorithm works over arbitrary fields as well. Indeed, we first compute all the divisors of f_0 using Theorem 1.9 and then proceed as above.

As a corollary we obtain the following result that settles Question 1.3 for the case of bounded individual degree polynomials. We note that this result also follows from the techniques of earlier work such as [BSV20, Vol15], but we explicitly state it as we could not find it in the literature.

Corollary 8.6 (Question 1.3 for (n, s, d)-sparse polynomials). *Let $\mathbb{F}$ be a field. There exists a deterministic,* $\mathrm{poly}(n, d!, s^d)$*-time algorithm that given an (n, s, d)-sparse polynomial f (over $\mathbb{F}$) that factors as a product of irreducible s-sparse polynomials $f = \prod g_i^{e_i}$, outputs all the g_is. Moreover, there is a deterministic algorithm for computing the multiplicities e_i, where: If $\mathrm{char}(\mathbb{F}) = 0$ or $\mathrm{char}(\mathbb{F}) > 2d$, the algorithm runs in time* $\mathrm{poly}(n, d!, s^d)$*. If $\mathbb{F}$ is arbitrary, the algorithm runs in time* $\mathrm{poly}(n, (d^2)!, s^{d^3})$.

Proof. Computing the multiplicities k_i of x_i as factors of f is trivial as f is given to us (it is (n, s, d)-sparse). Therefore, we may assume that f is not divisible by any monomial. By Theorem 1.9 we can find the set $\mathcal{F}$ of all the s-sparse divisors of f. By Theorem 1.8, we have that $|\mathcal{F}| \leqslant s^d$. As all irreducible factors g_i of f are s-sparse, $g_i \in \mathcal{F}$. Thus, we are left with the problem of identifying the g_is from $\mathcal{F}$. To that end, for every pair of two (non-constant) elements of $\mathcal{F}$, use the divisibility testing of [Vol17] to check if one divides the other. All non-constant elements of $\mathcal{F}$ that are not divisible by any other member of $\mathcal{F}$ are clearly the desired irreducible factors g_i.

As for the "moreover" part, we compute multiplicities using Lemma 4.19 if $\mathrm{char}(\mathbb{F}) = 0$ or $\mathrm{char}(\mathbb{F}) > 2d$; and Lemma 4.13 for arbitrary fields. □

8.3 Factoring a Product of Sparse Irreducible Polynomials

In this subsection we show our partial solution to the open problem from [DST24] (Question 1.4). As shown in the previous sections, it is natural to consider products of factors of sparse polynomials, as well as to seek a larger class of sparse divisors beyond the irreducible ones. Accordingly, we deduce our algorithm for solving Question 1.4 from the following more general theorem.

Theorem 1.11 (Factoring products of factors of sparse polynomials). *Let $\mathbb{F}$ be a field with $\mathrm{char}(\mathbb{F}) = 0$ or $\mathrm{char}(\mathbb{F}) > 2d$. There is a deterministic* $\mathrm{poly}(n, d^d, s^{d\log\ell}, \ell^d)$*-time algorithm which, given blackbox access to a product f of ℓ polynomials in $\mathcal{P}_{Factors}(n, s, d)$ over the field $\mathbb{F}$, returns all the s-sparse divisors of bounded individual degree d of f that are not divisible by any monomial, as well as the multiplicity k_i of each x_i as a factor of f. Moreover, if f was in fact a product of polynomials in $\mathcal{P}(n, s, d)$, there is such an algorithm that works over arbitrary fields, whose running time is* $\mathrm{poly}(n, (d^2)!, s^{d\log\ell + d^3}, \ell^d)$.

Proof. We again use the meta-algorithm of Section 6. Let $f \in \mathbb{F}[x_0, x]$ be a product of ℓ polynomials in $\mathcal{P}_{\mathrm{Factors}}(n+1, s, d)$.

Preprocessing: This step is identical to the corresponding step in the proof of Theorem 1.14, where we use $G = G_{(m)}$ for $m = (2nds)^2$, in the definition of $\hat{f}$. Note that $\tilde{f}$ has at most ℓd irreducible factors (as polynomials in y). Hence, it has $O((\ell d)^d)$ divisors of degree at most d in y.

Recursive call: Since f_0 is a product of ℓ polynomials in $\mathcal{P}_{\text{Factors}}(n, s, d)$ (by Lemma 6.2), we can find the set of all (n, s, d)-sparse divisors of f_0 not divisible by a monomial by applying one recursive call. Denote this set by $\mathcal{F}(f_0)$. By Corollary 8.3, we have that $|\mathcal{F}(f_0)| \leqslant \text{poly}(s^{d \log \ell})$.

Generating, and pruning, a list of candidate solutions: This step is identical to the corresponding step in the proof of Theorem 1.9. Indeed, generating the set of candidates from $\mathcal{F}(f_0)$ can be done in a similar manner, when we consider divisors $\hat{h}$ of $\hat{f}$ of degree at most d in y, as we are interested only in divisors of bounded individual degree d. Pruning can be done again by invoking Theorem 1.16.

Arbitrary fields: The only problem with extending the proof to arbitrary fields is the use of Theorem 1.16. Therefore, for fields with small characteristic, we assume that f is a product of s-sparse polynomials. In this case we can apply the "moreover" part of Theorem 1.16 and the argument goes through.

Running time: As in the previous proofs, the running time is determined by the size of the lists and the running time of Theorem 1.16. The claim bound follows. □

We finish with the following straightforward corollary of the last theorem, which gives the promised partial solution to Question 1.4.

Theorem 1.10 (Factoring products of (n, s, d)-sparse polynomials). *Let $\mathbb{F}$ be a field of characteristic zero or larger than $2d$. There is a deterministic $\text{poly}(n, d^d, s^{d \log \ell}, \ell^d)$-time algorithm which, given blackbox access to a product $f = \prod_{i=1}^{\ell} \phi_i$, where $\phi_1, \ldots, \phi_\ell \in \mathcal{P}(n, s, d)$ are all (not necessarily different) irreducible, returns the ϕ_i-s. Moreover, when the assumption on the field is removed, there is an algorithm that solves this problem in $\text{poly}(n, (d^2)!, s^{d \log \ell + d^3}, \ell^d)$-time.*

Proof. By Theorem 1.11 we can find in deterministic $\text{poly}(n, d^d, s^{d \log \ell}, \ell^d)$-time all the s-sparse divisors of f of bounded individual degree d. Applying Theorem 1.16, we can determine, for any two candidates ϕ and ψ, whether $\psi \mid \phi$. Since every irreducible factor of f is s-sparse with individual degree at most d, it follows that the irreducible factors are precisely those candidates ϕ that are not divisible by any other candidate ψ. To compute the multiplicity of each obtained irreducible factor as a factor of f, just invoke Lemma 4.19.

As for the "moreover" part, use the "moreover" part of Theorem 1.11 to get all the (n, s, d)-sparse divisors of f in $\text{poly}(n, (d^2)!, s^{d^3}, \ell^d)$-time. Replace Theorem 1.16 by its "moreover" part (or by the divisibility test of [Vol17]) to check for every two candidates if one divides the other, and replace the use of Lemma 4.19 by the "moreover" part of Lemma 4.13 to compute multiplicities. □

9 Factoring (n, s, d)-Sparse Polynomials

In this section we apply our methods to obtain factorization results for (n, s, d)-sparse polynomials, without any extra assumptions on the sparsity of their factors. For that, we rely on the sparsity upper bound of Theorem 2.14.

As discussed in Section 1.4.4, we give two algorithms: one for factoring a single (n, s, d)-sparse polynomial, and one for factoring a product of such polynomials. The first algorithm runs in time $\text{poly}(n, S)$ (where S is the bound of Theorem 2.14), while the second runs in $\text{poly}(n, d!, S^d)$-time. As we shall shortly see, the main reason for this difference is that it is easy to check if a given S-sparse polynomial divides the input polynomial in case it is s-sparse, since in this case the quotient should be S-sparse as well and therefore can be interpolated and reconstructed, but this is no longer the case for divisors of products of s-sparse polynomials.

9.1 Factorization of Sparse Polynomials

In this subsection we prove the following theorem, which is an improvement to the algorithm of Theorem 2.15.

Theorem 1.12. *There is a deterministic* $\mathrm{poly}(n, s^{d^2 \log n})$*-time algorithm which, given an* s*-sparse polynomial of bounded individual degree* d *on* n*-variables, returns its factorization into irreducible factors.*

Observe that, given only the bound in Theorem 2.14, this is the best one can hope for. Furthermore, as can be easily seen from the proof, an improvement to Theorem 2.14 automatically implies an improvement to the running time of our algorithm: if the bound in Theorem 2.14 is improved to a smaller bound M, then the running time of our algorithm will become $\mathrm{poly}(n, M, s^d)$.

Proof. The proof follows the lines of the proof of Theorem 1.9, with some small modifications that we shall explain. The algorithm is the same for any field $\mathbb{F}$, regardless of the characteristic.

Denote by S an explicit sparsity bound that follows from Theorem 2.14 for $(n+1, s, d)$-sparse polynomials, $S = \mathrm{poly}(n, s^{d^2 \log n})$.

Preprocessing step: We perform the same preprocessing step as in the proof of Theorem 1.9, where we define $\hat{f}$ with respect to the generator $G = G_{(m)}$ for $m = (2ndS)^2$.

Recursive call: Since f_0 is an (n, s, d)-sparse polynomial, a recursive call to the algorithm will return a list $\mathcal{F}'(f_0)$ containing its irreducible factors. Note that these factors may have sparsity S. By Theorem 1.8 the list has size at most $d \log s$, including multiplicities, when removing the variables it might contain (that is, we consider $\mathcal{F}'(f_0) \setminus \{x_i\}_{1 \leqslant i \leqslant n}$ instead of $\mathcal{F}'(f_0)$). Therefore, f_0 has at most s^d divisors not divisible by a monomial, that can be enumerated, and interpolated, in time $s^d \,\mathrm{poly}(S)$, by going over all subsets of irreducible divisors (taking multiplicities into account). We denote the list of all divisors not divisible by a monomial of f_0 by $\mathcal{F}(f_0)$.

Generating, and pruning, a list of candidate solutions: Generating the list of candidates is done as in the proof of Theorem 1.9. Indeed, $\mathcal{F}(f_0)$ contains all the divisors (not divisible by a monomial) of f_0, so we can proceed in exactly the same way to find a list of candidates for divisors of f (not divisible by a monomial). Regarding pruning, we note that in order to decide if a candidate polynomial g is a divisor of f, we don't need to use Theorem 1.16. Indeed, if $g \mid f$, then f/g is S-sparse. Therefore, we can interpolate $f(G)/g(G)$ as an S-sparse polynomial h and check whether $g \cdot h = f$.

This shows how to find all the divisors of f not divisible by a monomial. To conclude the proof, we need to show how to use this information to get the factorization of f into irreducible polynomials. We first show how to identify the irreducible factors among the divisors that we have found. Observe that if a divisor g is reducible, then its factors are also divisors of f. We can therefore check whether there exists a pair of divisors h_1, h_2 such that $g = h_1 \cdot h_2$, and return that g is irreducible if no such h_1, h_2 were found.

Computing the multiplicities: So far, we have the list of irreducible factors of f that are not monomials. Since the largest monomial dividing f was computed in the preprocessing step, we are left with the problem of computing the multiplicity of each of the irreducible factors of f. This can be done as in the previous step. Going over all values of $k \geqslant 0$ one by one we check whether there exists a divisor h of f such that $h \cdot g^k = f$. We stop at the first k where such a divisor does not exist and return that the multiplicity of g is $k-1$. Since, for $i < k$, all g^i are divisors of f, their sparsities are all $\mathrm{poly}(S)$, hence this algorithm runs in $\mathrm{poly}(S)$-time.

Running time: It is clear that the running time is polynomial in the sparsity S, the number of variables n, and the size of the lists which is upper bounded by s^d. □

9.2 Factorization of Products of Sparse Polynomials

We now prove the following theorem. Note that this theorem is different than Theorem 1.11; while Theorem 1.11 finds only the s-sparse divisors of a product of polynomials in $\mathcal{P}_{\mathrm{Factors}}(n, s, d)$, Theorem 1.13 computes the factorization of such products into irreducible factors, even if these are not s-sparse.

Theorem 1.13. *Let $\mathbb{F}$ be a field with $\mathrm{char}(\mathbb{F}) = 0$ or $\mathrm{char}(\mathbb{F}) > 2d$. There is a deterministic $\mathrm{poly}(n, s^{d^3 \log n}, (d\ell)^d)$-time algorithm which, given blackbox access to a product f of ℓ polynomials in $\mathcal{P}_{\mathit{Factors}}(n, s, d)$ over $\mathbb{F}$, returns its factorization into irreducible factors. Moreover, there is an algorithm that works over arbitrary fields, whose running time is $\mathrm{poly}(n, (d^2)!, s^{d^5 \log n}, \ell^d)$.*

Proof. Denote by $S = s^{O(d^2 \log(n+1))}$ the sparsity bound given by Theorem 2.14 for $(n+1, s, d)$-sparse polynomials.

We follow the lines of the proof of Theorem 1.14, but combine it the sparsity bound of Theorem 2.14.

Preprocessing: This is the same as in the proof of Theorem 1.14, where $\hat{f}$ is defined as $\hat{f} = \tilde{f}(y, G)$, where $G = G_{(m)}$. Here, m is as given by Theorem 5.2 for (n, S, d)-sparse polynomials (that is, $m = \mathrm{poly}(n, d!, S^d) = \mathrm{poly}(n, d!, s^{d^3 \log n})$). By Claim 2.3, $\hat{f}$ has at most ℓd irreducible factors (as polynomials in y), and hence at most $O((\ell d)^d)$ divisors of degree at most d in y.

Recursive call: By Lemma 6.2, f_0 is a product of ℓ elements of $\mathcal{P}_{\mathrm{Factors}}(n, s, d)$. Thus, by making one recursive call, we obtain its factorization into irreducible polynomials. Denote by $\mathcal{F}(f_0)$ the set of irreducible factors of f_0.

Generating, and pruning, a list of candidate solutions: Since we have all the irreducible factors of f_0, we can proceed as in the proof of Theorem 1.14 and obtain a list of (n, S, d)-sparse divisors of f, using Theorem 5.2 and Theorem 1.16.

Pruning the list: We now have a set of candidates, all of which are S-sparse divisors of f. Note that if g is such a divisor that is not irreducible, then every irreducible factor of g must also be in the list. Therefore, to identify the irreducible factors, we invoke Theorem 1.16 (for S-sparse polynomials) to each pair of candidates ϕ, ψ and check if $\psi | \phi$. The candidates ϕ for which there are no other candidates that divide them are the desired irreducible factors.

Computing the multiplicities: We compute the multiplicities of the irreducible factors using Lemma 4.19.

Arbitrary fields: To obtain the result we replace the use of Theorem 5.2 and Theorem 1.16 by their respective "moreover" parts. We also use Lemma 4.13 instead of Lemma 4.19.

Running time: First consider fields of zero or large characteristic. The running time is determined by the number of divisors of $\hat{F}$, and the application of Theorem 5.2, Theorem 1.16, and Lemma 4.19 for $(n+1, S, d)$-sparse polynomials. This yields the claimed $\mathrm{poly}(n, d!, S^d, (d\ell)^d) = \mathrm{poly}(n, s^{d^3 \log n}, (d\ell)^d)$ running time.

For arbitrary fields, the "moreover" parts of Theorem 5.2 and Theorem 1.16 and Lemma 4.13 imply a running time of $\mathrm{poly}(n, (d^2)!, S^{d^3}, (d\ell)^d) = \mathrm{poly}(n, (d^2)!, s^{d^5 \log n}, \ell^d)$ □

Remark 9.1. In Theorem 1.10 we gave a $\mathrm{poly}(n, d^d, s^{d \log \ell}, \ell^d)$-time algorithm for this problem, with the extra assumption that the irreducible factors of f are in fact in $\mathcal{P}(n, s, d)$. Hence, Theorem 1.13 is more general, and in fact gives a $\mathrm{poly}(n, s^{d^3 \log n}, (d\ell)^d)$ algorithm for Theorem 1.10. That is, we have two algorithms for solving Question 1.4: one is quasi-polynomial in the length of the product and polynomial in the number of variables, and the other is quasi-polynomial in the number of variables and polynomial in the number of terms in the product.

10 Future Work and Open Problems

We conclude with a discussion of possible directions for future research.

First, it is somewhat disappointing that our main technical lemma, Lemma 4.15, is only valid over fields of zero or sufficiently large characteristic. It would be interesting to extend this result to arbitrary fields. As discussed in Section 4, some results are known in this direction; the remaining challenge is to eliminate the need for the additional non-degeneracy condition required in small characteristic.

Next, it would be extremely interesting to identify the irreducible factors in Theorem 1.9. In other words, one would like an efficient algorithm that, given an (n, s, d)-sparse polynomial f, decides whether f is irreducible in time $\mathrm{poly}(n, s^d)$. The techniques developed in this paper appear insufficient for addressing this problem, as they only provide information about the coprimality of $\phi(G_{(m,i)})$ and $\psi(G_{(m,i)})$ for $\phi, \psi \in \mathcal{P}_{\mathrm{Factors}}(n, s, d)$, and do not yield information about the number of irreducible factors.

It would also be interesting to generalize our rational interpolation theorem. For instance, even the seemingly basic problem of deterministically interpolating a quotient of coprime sparse polynomials in polynomial time remains wide open.

Obtaining a polynomial-time algorithm for Question 1.4, thereby improving Theorem 1.10, also appears to require new ideas. As observed throughout the paper, the main difficulty is that c_0 may contain non-sparse irreducible factors, forcing the recursive call on f_0 to enumerate all possibilities and leading to a quasi-polynomial running time. By contrast, the polynomial-time algorithm of Theorem 1.14 relies on the fact that c_0 (or b in that setting) factors only into sparse polynomials, due to Lemma 2.13. This structure, together with the ability to determine multiplicities via Corollary 4.12, suffices to obtain a polynomial-time algorithm.

Finally, any improvement to the bound in Theorem 2.14 would directly strengthen the results of Section 9, and would be of significant interest.

Bibliography

[BKR$^+$25] Somnath Bhattacharjee, Mrinal Kumar, Shanthanu S Rai, Varun Ramanathan, Ramprasad Saptharishi, and Shubhangi Saraf. Closure under factorization from a result of Furstenberg. *arXiv preprint arXiv:2506.23214*, 2025. 1

[BOT88] Michael Ben-Or and Prasoon Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 301–309, 1988. 1

[BS25] Pranav Bisht and Nitin Saxena. Derandomization via symmetric polytopes: Poly-time factorization of certain sparse polynomials. *ACM Transactions on Computation Theory*, 17(2):1–20, 2025. 5

[BSV20] 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. 1, 2, 3, 4, 5, 6, 9, 11, 12, 14, 15, 39

[BV25] Pranav Bisht and Ilya Volkovich. On solving sparse polynomial factorization related problems. *Computational Complexity*, 34(1):7, 2025. 1, 2, 4, 5, 6, 7, 8, 9, 12, 19, 21, 23, 25, 28

[CDGK91] Michael Clausen, Andreas Dress, Johannes Grabmeier, and Marek Karpinski. On zero-testing and interpolation of k-sparse multivariate polynomials over finite fields. *Theoretical Computer Science*, 84(2):151–164, 1991. 1

[CL11] Annie Cuyt and Wen-shin Lee. Sparse interpolation of multivariate rational functions. *Theoretical Computer Science*, 412(16):1445–1456, 2011. 5

[DG24] Pranjal Dutta and Sumanta Ghosh. Sigact news complexity theory column 121. *SIGACT News*, 55(2):53–88, June 2024. 1

[DST24] Pranjal Dutta, Amit Sinhababu, and Thomas Thierauf. Derandomizing multivariate polynomial factoring for low degree factors. In Amit Kumar and Noga Ron-Zewi, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2024, London School of Economics, London, UK, August 28-30, 2024*, volume 317 of *LIPIcs*, pages 75:1–75:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. 1, 2, 5, 35, 39

[For15] 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. 5, 6

[FS15] Michael A Forbes and Amir Shpilka. Complexity theory column 88: Challenges in polynomial factorization1. *ACM SIGACT News*, 46(4):32–49, 2015. 2

[GK87] Dima Yu Grigoriev and Marek Karpinski. The matching problem for bipartite graphs with polynomially bounded permanents is in nc. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 166–172. IEEE, 1987. 1

[GKL12] Ankit Gupta, Neeraj Kayal, and Satyanarayana V. Lokam. Reconstruction of depth-4 multilinear circuits with top fan-in 2. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 625–642. ACM, 2012. 14

[GKS90] Dima Grigoriev, Marek Karpinski, and Michael F. Singer. Fast parallel algorithms for sparse multivariate polynomial interpolation over finite fields. *SIAM J. Comput.*, 19(6):1059–1063, 1990. 1

[GKS94] Dima Grigoriev, Marek Karpinski, and Michael F Singer. Computational complexity of sparse rational interpolation. *SIAM Journal on Computing*, 23(1):1–11, 1994. 2, 5

[GKZ08] Israel M Gelfand, Mikhail M Kapranov, and Andrei V Zelevinsky. *Discriminants, resultants and multidimensional determinants*. Modern Birkhäuser Classics. Birkhäuser Boston, Inc., Boston, MA, 2008. Reprint of the 1994 edition. 15

[Kal85a] Erich Kaltofen. Computing with polynomials given by straight-line programs I: greatest common divisors. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 131–142, 1985. 1

[Kal85b] Erich Kaltofen. Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization. *SIAM J. Comput.*, 14(2):469–489, 1985. 14

[KRS24] 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. 5

[KS01] Adam R. Klivans and Daniel Spielman. Randomness efficient identity testing of multivariate polynomials. STOC '01, page 216–223, New York, NY, USA, 2001. Association for Computing Machinery. 1, 7, 17

[Len85] Arjen Klaas Lenstra. Factoring multivariate polynomials over finite fields. *J. Comput. System Sci.*, 30(2):235–248, 1985. 14

[LLL82] Arjen K. Lenstra, Hendrik W. jr. Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982. 13

[Sax09] Nitin Saxena. Progress on polynomial identity testing. *Bull. EATCS*, 99:49–79, 2009. 1

[Sax14] Nitin Saxena. Progress on polynomial identity testing-ii. In *Perspectives in Computational Complexity: The Somenath Biswas Anniversary Volume*, pages 131–146. Springer, 2014. 1

[SV09] Amir Shpilka and Ilya Volkovich. Improved polynomial identity testing for read-once formulas. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 700–713. Springer, 2009. 17

[SY10] Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends® in Theoretical Computer Science*, 5(3–4):207–388, 2010. 1

[vdHL21] Joris van der Hoeven and Grégoire Lecerf. On sparse interpolation of rational functions and gcds. *ACM Commun. Comput. Algebra*, 55(1):1–12, 2021. 5

[Vol15] 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)*, pages 943–958. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2015. 1, 2, 4, 5, 39

[Vol17] Ilya Volkovich. On some computations on sparse polynomials. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017)*, pages 48–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017. 1, 2, 4, 5, 6, 7, 14, 16, 25, 38, 39, 40

[VZGK85] Joachim Von Zur Gathen and Erich Kaltofen. Factoring sparse multivariate polynomials. *Journal of Computer and System Sciences*, 31(2):265–287, 1985. 1, 2

[vzGS92] Joachim von zur Gathen and Victor Shoup. Computing Frobenius maps and factoring polynomials. *Comput. Complexity*, 2(3):187–224, 1992. 13

A An Alternative Proof of a Result from Section 4

In this section we describe an alternative proof of Corollary 4.12. While this lemma is valid over arbitrary fields, the proof presented here works only for zero characteristic fields or when the characteristic is larger than the individual degree. Thus, both Corollary 4.12 and Lemma 4.15 are stronger than what we present here. However, the proof method is slightly different, and we include it here for the possibility it would be useful for other applications.

While the proofs of Corollary 4.12 and Lemma 4.15 use the rank of the Sylvester matrix to study the factorization of a polynomial, our proof here uses an operator we call Δ_k. The well known discriminant Δ of a polynomial (defined in Definition 2.19) is an operator that can be used to detect the existence of irreducible factors of multiplicity greater than one. Its generalization Δ_k, that we soon define, is an operator that detects irreducible factors of multiplicity at least $k+1$, for small values of k. The second important property of Δ_k is that, like the discriminant, it does not blow-up the sparsity of its input by too much.

We now give the definition of Δ_k.

Definition A.1. Let $f \in \mathbb{F}[x]$ be a non-constant polynomial. Define

$$\Delta_k(f)(\lambda_1, \ldots, \lambda_k) = \mathrm{res}(f, \lambda_1 f^{(1)} + \ldots + \lambda_k f^{(k)}) \in \mathbb{F}[\lambda_1, \ldots, \lambda_k] \tag{19}$$

where $f^{(i)}$ is the i-th derivative of f.

For $f \in \mathbb{F}[x_0, x_1, \ldots, x_n]$, $\Delta_{k,x_0}(f) \in \mathbb{F}[x_1, \ldots, x_n, \lambda_1, \ldots, \lambda_k]$ is just $\Delta_k(f)$ when one views f as an element of $\mathbb{F}(x_1, \ldots, x_n)[x_0]$.

Example A.2. *For $k = 1$, one sees that $\Delta_1 = \lambda_1 \Delta$. Thus, for example, $\Delta_1(f) = 0$ if and only if $\Delta(f) = 0$.*

As discussed, one should think of Δ_k as some generalization of Δ that captures the existence of factors of multiplicity at least $k+1$, the same way Δ captures the existence of factors of multiplicity at least two. The example justifies this statement for $k = 1$; the general statement is given by the following lemma.

Lemma A.3. *Let $f \in \mathbb{F}[x]$ be a non-constant polynomial of at most degree d, and suppose that $\mathrm{char}(\mathbb{F})$ is zero or larger than d. Then f has an irreducible factor of multiplicity at least $k+1$ if and only if $\Delta_k(f) = 0$. In particular, if $f \in \mathbb{F}[x_0, x_1, \ldots, x_n]$ is such that $x_0 \in var(f)$ and $\deg_{x_0}(f) \leqslant d$, under the same assumption on $\mathbb{F}$, then f has an irreducible factor of multiplicity at least $k+1$ that depends on x_0 if and only if $\Delta_{k,x_0}(f) = 0$.*

Proof. Note that as $\mathrm{char}(\mathbb{F}) \in \{0\} \cup [d+1, \infty)$, and $\deg f \leqslant d$, every irreducible factor ϕ of f is separable, that is, it factors to a product of different linear polynomials over $\overline{\mathbb{F}}$. One way to see it is to note that $\phi' \neq 0$ (because of the assumption on $\mathbb{F}$), and therefore ϕ' must be coprime to ϕ, so by Lemma 2.20 we must have that $\Delta(\phi) = \mathrm{res}(\phi, \phi') \neq 0$, and hence (again by Lemma 2.20) ϕ is square-free.

Therefore, f has an irreducible factors of multiplicity at least $k+1$ if and only if it has a root of multiplicity at least $k+1$ over $\overline{\mathbb{F}}$.

Suppose, without loss of generality, that f is monic. In this case, by Lemma 2.20, we know that:

$$\Delta_k(f) = \mathrm{res}(f, \lambda_1 f^{(1)} + \ldots + \lambda_k f^{(k)}) = \prod_{\alpha : f(\alpha)=0} \left(\lambda_1 f^{(1)}(\alpha) + \ldots + \lambda_k f^{(k)}(\alpha)\right) \tag{20}$$

That is, $\Delta_k(f)$ is a product of linear functionals in $\overline{\mathbb{F}}[\lambda_1, \ldots, \lambda_k]$, and therefore vanishes if and only if one of these functionals vanishes.

To this end, note that for a root α of f, a functional $\lambda_1 f^{(1)}(\alpha) + \ldots + \lambda_k f^{(k)}(\alpha)$ vanishes if and only if $f^{(i)}(\alpha) = 0$ for $0 \leqslant i \leqslant k$, that is, if and only if α is a root of multiplicity at least $k+1$ of f. This proves the lemma.

As for the "in particular" part, note that by Gauss' lemma (Lemma 2.1), irreducible factors of $f \in \mathbb{F}[x_0, x_1, \ldots, x_n]$ that depend on x_0 correspond to irreducible factors of f as an element of $\mathbb{F}(x_1, \ldots, x_n)[x_0]$. Therefore, the last statement of the lemma follows easily from the first by working over $\mathbb{F}(x_1, \ldots, x_n)$, noting that its characteristic is the characteristic of $\mathbb{F}$. □

Remark A.4. The condition on $\deg(f)$ and $\mathrm{char}(\mathbb{F})$ is necessary. For example, consider the polynomial $f(x,y) = x^p - y \in \mathbb{F}_p[x,y]$ for a prime p. f is clearly irreducible, but $\Delta_x(f) = 0$, as f is a square (and, in fact, a p-th power) as an element of $\overline{\mathbb{F}_p(y)}[x]$.

The lemma above provides one manner in which Δ_k generalizes Δ. For our purpose, we need to study another manner in which such a generalization holds - action on sparse polynomials.

Lemma A.5. *Let $f \in \mathbb{F}[x_0, x_1, \ldots, x_n]$ be an s-sparse polynomial with bounded individual degree d. Then, $\Delta_k f \in \mathbb{F}[x_1, \ldots, x_n, \lambda_1, \ldots, \lambda_k]$ is a $(2d)!(ks)^{2d}$-sparse polynomial of individual degree at most $2d^2$.*

Proof. The lemma follows immediately by the Lemma 2.20, by noting that $\lambda_1 f^{(1)}(\alpha) + \ldots + \lambda_k f^{(k)}(\alpha)$ is a polynomial of sparsity ks and bounded individual degree d. □

We now show how to use Δ_k to obtain a similar result to Corollary 4.12. To do that, we start with the following lemma.

Lemma A.6. *Suppose $\mathbb{F}$ is a field for which $\mathrm{char}(\mathbb{F}) \in \{0\} \cup [d+1, \infty)$. Let f be an (n,s,d)-sparse polynomial defined over $\mathbb{F}$, and fix some coordinate i. Suppose that f does not have an irreducible factor of multiplicity at least $k+1$ as a polynomial in x_i. Then there exists an explicit $m = \mathrm{poly}(n, d!, (ks)^d)$ for which $f(G_{(m,i)})$ also does not have an irreducible factor of multiplicity at least $k+1$ as a polynomial in u.*

Proof. By Lemma A.3, as f does not have an irreducible factor of multiplicity at least $k+1$ as a polynomial in x_i, we have that $\Delta_{k,x_i}(f) \neq 0$. By Lemma A.5, $\Delta_{k,x_i}(f)$ is a $\mathrm{poly}(d!(ks)^d)$-sparse polynomial of bounded individual degree $2d^2$ on $n+d-1$ variables. Therefore, by Lemma 3.4, for $m = ((n+d)(d!(ks)^d)(2d^2))^2 = \mathrm{poly}(n, d!, (ks)^d)$, we know that $\Delta_{k,x_i}(f)(G_{(m,i)}) \neq 0$, where we substitute every coordinate x_j for $j \neq i$ according to $G_{(m,i)}$ (leaving $\lambda_1, \ldots, \lambda_k$ alive). It is easy to see that $\Delta_{k,u}(f(G_{(m,i)})) = \Delta_{k,x_i}(f)(G_{(m,i)})$, so it follows that $\Delta_{k,u}(f(G_{(m,i)})) \neq 0$. The claim then follows from the other direction of Lemma A.3. □

We now apply the last lemma to prove the following weaker version of Corollary 4.12.

Lemma A.7. *Suppose $\mathbb{F}$ is a field for which* $\mathrm{char}(\mathbb{F}) \in \{0\} \cup [d+1, \infty)$*. Let ϕ be an irreducible s-sparse polynomial of bounded individual degree d, and let ψ be an irreducible factor of some s-sparse polynomial g of bounded individual degree d, different than ϕ, both defined over $\mathbb{F}$. Fix some i for which $x_i \in var(\phi)$. Then, for some explicit value of* $m = \mathrm{poly}(n, d^{d^2}, s^{d^3})$*, we have that $\phi(G_{(m,i)})$ is coprime to $\psi(G_{(m,i)})$, when viewed as polynomials in u. Moreover, if ϕ is multiquadratic, taking* $m = \mathrm{poly}(n, d^d, s^{d^2})$ *actually suffices.*

Proof. Let $g = \prod \psi \cdot \phi^k$, each ψ is irreducible not associate to ϕ. From now on we view these polynomials as polynomials in x_i.

Consider the polynomial $\tilde{g} = \phi^{d+1} g$. Its sparsity is at most s^{d+2}, its degree is at most $O(d^2)$, and ϕ divides it exactly $k+d+1$ times. As $\deg_{x_i}(g) \leqslant d$, every other irreducible factor of $\tilde{g}$ is of multiplicity at most $d-k < d+1$, so ϕ must be the factor of largest multiplicity of $\tilde{g}$. That is, $\tilde{g}$ has no factors of multiplicity more than $k+d+1$. By Lemma A.6, if $m = \mathrm{poly}(n, (d^2)!, ((k+d)s^{d+2})^{d^2}) = \mathrm{poly}(n, d^{d^2}, s^{d^3})$ the same is true for $\tilde{g}(G_{(m,i)})$ (as a polynomial in u). Thus, it must be the case that $\psi(G_{(m,i)})$ is prime to $\phi(G_{(m,i)})$ - otherwise we would have had a factor of $\phi(G_{(m,i)})$ that appears at least $k+d+2$ times in the factorization of $\tilde{g}(G_{(m,i)})$ ($k+d+1$ times that come from the ϕ factors of $\tilde{g}$, and one extra coming from the ψ factor).

As for the "moreover" part, note that if ϕ is multiquadratic then $\tilde{g}$ is of degree $O(d)$, so taking $m = \mathrm{poly}(n, d!, ((k+d)s^{d+2})^d) = \mathrm{poly}(n, d^d, s^{d^3})$ is enough. □

Remark A.8. In fact, the proof implies that $\phi(G_{(m,i)})$ is square free, since the generator G preserves the non-vanishing of the discriminant $\Delta(\phi)$.