On Solving Sparse Polynomial Factorization Related Problems

Pranav Bisht * Ilya Volkovich†

Abstract

In a recent result of Bhargava, Saraf and Volkovich [FOCS'18; JACM'20], the first factor sparsity bound for constant individual degree polynomials was shown. In particular, it was shown that any factor of a polynomial with at most s terms and individual degree bounded by d can itself have at most $s^{O(d^2 \log n)}$ terms. It is conjectured, though, that the "true" sparsity bound should be polynomial (i.e. $s^{\mathrm{poly}(d)}$). In this paper we provide supporting evidence for this conjecture by presenting polynomial-time algorithms for several problems that would be implied by a polynomial-size sparsity bound. In particular, we give efficient (deterministic) algorithms for identity testing of $\Sigma^{[2]}\Pi\Sigma\Pi^{[\deg_{x_i} \leq d]}$ circuits and testing if a sparse polynomial is an exact power. Hence, our algorithms rely on different techniques.

Keywords: Sparse Polynomials, Identity Testing, Derandomization, Factor-Sparsity, Multivariate Polynomial Factorization.

1 Introduction

Polynomial Factorization is one of the core problems in algebraic complexity: given a multivariate polynomial $f \in \mathbb{F}[x_1, x_2 \ldots, x_n]$ over a field $\mathbb{F}$, output all its irreducible factors. In addition to being a natural problem, its importance is highlighted by various applications such as: list decoding [Sud97, GS99], derandomization [KI04], cryptography [CR88] and others. In the seminal works of [Kal89, KT90], efficient randomized factorization algorithms were presented. Yet, coming up with an efficient *deterministic* factorization algorithm remains a long-standing open question.

Indeed, one aspect of the computational problem is the representation of the input polynomial. One natural way to represent a polynomial is by listing all its terms and coefficients. This is known as *dense* representation. Yet, even if the individual degree of every variable is bounded by a small constant d, the total number of terms can be exponentially large, reaching $(d+1)^n$. Nonetheless,

*Department of CSE, IIT Kanpur, India, pbisht@cse.iitk.ac.in

†Computer Science Department, Boston College, Chestnut Hill, MA. Email: ilya.volkovich@bc.edu

ISSN 1433-8092

in many applications [Zip79, GK85, BOT88, GJR10] the actual number of non-zero terms in a polynomial is much smaller - poly(n). Such polynomials are referred to as *sparse* polynomials, which will be the focus of our paper.

A key question that precedes the design of efficient factorization algorithms for sparse polynomial is whether a factor of a sparse polynomial is (itself) sparse. Indeed, this question was first studied by von zur Gathen and Kaltofen in [GK85] that gave a randomized factorization algorithm where the runtime depends on the number of terms in the output factors. In the same paper they provided an example inspired by geometric series (see below) of a family of polynomials that have factors with a super-polynomial (quasi-polynomial) number of terms. We denote by $\|f\|$ the *sparsity* of f. That is, the number of non-zero terms in f.

Example 1.1 ([GK85])**.** Let $n \geq 1$. Consider the polynomial $f(\overline{x}) = \prod_{i\in[n]} (x_i^n - 1)$ which can be written as a product of $g(\overline{x}) = \prod_{i\in[n]} (1 + x_i + \ldots + x_i^{n-1})$ and $h(\overline{x}) = \prod_{i\in[n]} (x_i - 1)$.
Observe that $\|f\| = \|h\| = 2^n$ while $\|g\| = n^n$, resulting in a quasi-polynomial blow-up[1].

Furthermore, for fields with finite characteristics the blow-up can be significantly larger:

Example 1.2 ([Vol15])**.** For a prime p, let $f \in \mathbb{F}_p[x_1, \ldots x_n]$, and let $0 < d < p$. Consider

$$f(\overline{x}) = (x_1 + x_2 + \ldots + x_n)^p = x_1^p + x_2^p + \ldots + x_n^p$$
$$g(\overline{x}) = (x_1 + x_2 + \ldots + x_n)^d$$

Notice that g is a factor of f, but $\|f\| = n$ and $\|g\| = \binom{n+d-1}{d} = n^{\Omega(d)}$.

Based on the above, we should first try to obtain a "sparsity-bound" on factors of sparse polynomials with constant (i.e. bounded) individual degree. More formally, for some fixed d, we require that $\deg_{x_i} \leq d$, for all variables x_i. The simplest case (when $d = 1$) corresponds to the so-called *multilinear* polynomials. In [SV10], it was shown that a factor of an s-sparse[2] multilinear polynomial is itself s-sparse. Subsequently, in [Vol17], this result was extended to the case of *multiquadratic* polynomials (i.e. when $d = 2$). In a recent work of [BSV20], a quasi-polynomial-size sparsity bound was given for *any* fixed d. Specifically, it was shown that a factor of an s-sparse polynomial with individual degree bounded by d is $s^{O(d^2 \log n)}$-sparse. In addition, [BSV20] designed a factorization algorithm whose runtime is efficient in terms of the sparsity bound. As a result they obtained a deterministic quasi-polynomial-time factorization algorithm for sparse polynomials with bounded individual degree. In the same paper it was also conjectured that the "true" sparsity bound should be polynomial rather than quasi-polynomial. More formally:

Conjecture 1.3. *There exists a universal constant $k \in \mathbb{N}$ such that for any $s, d \in \mathbb{N}$, any factor of an s-sparse polynomial with individual degree bounded by d has at most s^{d^k} terms.*

[1] Although g is not irreducible, this issue can be resolved using standard techniques. For example, by considering the product $f + yh = (g + y)h$ for a new variable y.

[2] A polynomial is s-sparse, it if contains at most s non-zero terms.

In this paper we provide supporting evidence for this conjecture by presenting deterministic polynomial-time algorithms for some problems that reduce to sparse polynomial factorization. It is to be noted that invoking the aforementioned factorization algorithm of [BSV20] with a polynomial-size sparsity bound would imply a (deterministic) *polynomial-time* algorithm for sparse polynomial factorization and hence polynomial-time algorithms for these problems. In the absence of a polynomial-size sparsity bound, we design our algorithms using new techniques.

1.1 Our Results

We will now describe our main results. In what follows, $\mathbb{F}$ is an arbitrary field (finite or otherwise).

1.1.1 Identity Testing for $\Sigma^{[2]}\Pi\Sigma\Pi^{[\deg_{x_i}\leq d]}$ Circuits

The Polynomial Identity Testing (PIT) problem asks to decide whether a given input polynomial is identically zero. The input is usually given in the form of an algebraic circuit (see Appendix A for definition). The PIT algorithm is called *white-box* if one can look 'inside' the circuit. The algorithm is called *black-box* if the circuit is given via an oracle access, where one is only allowed to evaluate the polynomial on a chosen set of input points. PIT is one of the few natural problems which have a simple efficient randomized algorithm [DL78, Sch80, Zip79] but lack a deterministic one. Indeed, it has been a long standing open question to come up with an efficient deterministic algorithm for this problem. Our first result is an efficient (deterministic) identity testing algorithm for the class of $\Sigma^{[2]}\Pi\Sigma\Pi^{[\deg_{x_i}\leq d]}$ circuits, where a $\Sigma^{[2]}\Pi\Sigma\Pi^{[\deg_{x_i}\leq d]}$ circuit C of size s computes a polynomial of the form:

$$C = \prod_{i=1}^{r} g_i + \prod_{j=1}^{m} h_j$$

where each polynomial (g_i and h_j) is an s-sparse polynomial with individual degree at most d (for some fixed d). Note, though, that r and m, and hence the total degree of of C, can be arbitrary (i.e. polynomially) large. In particular, the polynomial computed by C may not itself be sparse. This class generalizes the model considered in [Vol17], where $m = 1$ and the g_i-s are irreducible polynomials. For the formal definition of our circuit model and further discussion, see Section 3.1.

Observe that the identity testing problem for this circuit class reduces to polynomial factorization of sparse polynomials with bounded individual degree. Therefore, by invoking the factorization algorithm of [BSV20], we can get an algorithm whose runtime is efficient in terms of the sparsity bound. Plugging in the best bound of [BSV20], results in a quasi-polynomial-time algorithm. Our next result gives a *polynomial-time* algorithm for this model. In addition, our algorithm operates in the black-box setting, whereas the described factorization-based algorithm is a white-box algorithm.

Theorem 1. *There exists a deterministic algorithm that given a black-box access to a* $\Sigma^{[2]}\Pi\Sigma\Pi^{[\deg_{x_i}\leq d]}$ *circuit* C *of size* s *determines if* $C \equiv 0$, *in time* $\mathrm{poly}((sd)^d, n)$.

An important ingredient in our algorithm is a result that links the gcd of two polynomials, their subresultant and the resultant of their coprime parts - in the **multivariate** setting. See Section 3.2 for the formal definitions.

Theorem 2. *Let $A, B \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be two polynomials such that $A = f \cdot g$ and $B = h \cdot g$ and let x_i be a variable. Then*

$$S_{x_i}(d, A, B) = g \cdot \mathrm{Res}_{x_i}(f, h) \cdot \mathrm{lc}_{x_i}(g)^{m'+n'-1}$$

here $m = \deg_{x_i}(A)$, $n = \deg_{x_i}(B)$, $d = \deg_{x_i}(g)$, $m' = \deg_{x_i}(f) = m - d$ and $n' = \deg_{x_i}(h) = n - d$. In addition:

- *$\mathrm{Res}_{x_i}(f, h)$ is the resultant of f and h w.r.t the variable x_i.*
- *$\mathrm{lc}_{x_i}(g)$ is the leading coefficient of g when written as a polynomial in x_i*
- *And finally, $S_{x_i}(d, A, B)$ is the d-th subresultant of A and B.*

To put the result in context, consider two univariate polynomials $A, B \in \mathbb{F}[x]$. A classical result in the Theory of Resultants (see e.g. [GCL92, GG99, CLO15]) states that:

1. $\mathrm{Res}(A, B) \equiv 0$ if and only if $\gcd(A, B)$ is non-trivial.
2. The *j-th Subresultant* $S(j, A, B) \equiv 0$ whenever $j < \deg(\gcd(A, B))$.
3. There exist a non-zero field element $\alpha \in \mathbb{F}$ such that $S(j, A, B) = \alpha \cdot \gcd(A, B)$, when $j = \deg(\gcd(A, B))$.

In the multivariate setting one can always regard multivariate polynomials as polynomials in a single variable with coefficients being rational functions in the remaining variables. Yet, in this case α is no longer a mere 'field element' as it can now be an arbitrary rational function in the remaining variables! From that perspective, our result can be seen as explicitly expressing α as a polynomial (and not even a rational function) in the remaining variables. We believe that this explicit relation could be of interest in its own right.

1.1.2 Exact Powers

Our next result pertain to exact powers of polynomials. A polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ is an *exact power* if there exists (another) polynomial $g \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ and $e \in \mathbb{N}$ such that $f = g^e$. We note that despite the rich structure, the best known sparsity bound for exact roots (i.e. $\|g\|$ in terms of $\|f\|$) is the general sparsity bound of size $s^{O(d^2 \log n)}$ by [BSV20]. Hence, one can use the factorization algorithm of [BSV20] to test if a given sparse polynomial is an exact power, in quasi-polynomial time. Similarly, a polynomial-size sparsity bound, even for the case of exact roots, would imply a polynomial-time algorithm for exact-power testing problem. We provide a polynomial-time algorithm for exact-power testing that **does not** rely on this sparsity bound.

Theorem 3. *There is a deterministic algorithm that given a sparse polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of individual degree d as an input, decides whether $f = g^e$ for some polynomial $g \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ and $e \in \mathbb{N}$, in time* $\mathrm{poly}(s^{d^2}, n)$.

We remark that the algorithm only performs exact-power testing and **does not** output a "witness" polynomial g. Indeed, a polynomial-time algorithm that actually outputs g would imply a polynomial-size sparsity bound on exact roots! In addition, the runtime of our algorithm is polynomial in the bit-complexity of the field elements since it does not rely on univariate polynomial factorization. For instance, for finite fields we get the runtime of $\mathrm{poly}(\log |\mathbb{F}|)$ vs $\mathrm{poly}(|\mathbb{F}|)$.

1.1.3 Improved Sparsity Bounds for Co-factors of Multilinear Polynomials

Given two polynomials $f, h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ such that $f = gh$, g is called a *quotient polynomial* or a *co-factor* of h. We study the problem of multilinear co-factor sparsity: suppose f is s-sparse and h is multilinear. How sparse/dense can g be? We remark that any (even non-constructive) efficient upper bound on the sparsity of g allows us to compute g efficiently by interpolating the ratio f/h using a reconstruction algorithm for sparse polynomials (e.g. [KS01]) and verifying the result.

The motivation to study this problem is two-fold: first of all, by previous results (see e.g. [BSV20]) a multilinear factor of an s-sparse polynomial (of any degree) is itself s-sparse. This suggests more structure for multilinear co-factors we could potentially exploit. Second, a polynomial-size sparsity bound on multilinear co-factors g (even when the individual degree of g is $d = 2$) would imply a polynomial-size sparsity bound for (**all** factors of) polynomials with individual degree $d = 3$. We note that the multicubic ($d = 3$) case is the first instance where we do not have a polynomial-size factor-sparsity bound yet. Indeed, multilinear co-factors can be seen as the "bottle-neck" for this case. The formal argument is given in Section 1.4.

To state our result we need the following technical definition. We say that a polynomial $h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ has a *unique projection of length k* if there exist k variables $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ and k corresponding exponents $e_1, e_2, \ldots, e_k$ such that h has a unique monomial that contains the pattern $x_{i_1}^{e_1} x_{i_2}^{e_2} \cdot \ldots \cdot x_{i_k}^{e_k}$ (see Definition 5.5 for more details).

Theorem 4. *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial of sparsity s and individual degree at most d such that $f = gh$. Suppose, in addition, that h is a multilinear polynomial with a unique projection of length k. Then the sparsity of g is bounded by $s^{O(dk)}$.*

We remark that Example 1.2 with $d = p - 1$ (resulting in a lower bound of $n^{\Omega(p)}$) showcases the tightness of our result as here f is n-sparse and $h = x_1 + \ldots + x_n$ has a unique projection of length 1 (e.g. x_1) which results in an upper bound of $n^{O(p)}$ for g.

We can also extend Theorem 4 to the case of a co-factor of a power of a multilinear polynomial. See Theorem 5.25 for the formal statement. Subsequently, we show that every multilinear s-sparse polynomial has a unique projection of length $O(\log s)$ (see Lemma 5.9). By plugging in this result into Theorem 4, we obtain a new sparsity bound of size $s^{O(d \log s)}$ for all multilinear co-factors.

Corollary 1.4. *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial of sparsity s and individual degree at most d such that $f = gh$. Suppose, in addition, that h is a multilinear polynomial. Then the sparsity of g is bounded by $s^{O(d \log s)}$.*

The obtained bound is slightly better than the general sparsity bound of size $s^{O(d^2 \log n)}$ by [BSV20] when $s = \text{poly}(n)$. Although our overall improvement may seem incremental (e.g. it does not allow us to "get rid" of the $\log n$ in the exponent) our main contribution here is conceptual: identifying a combinatorial property - the length of the shortest unique projection - that governs the bound on the sparsity of multilinear co-factors.

1.2 Related works

For the sparse polynomial factorization problem, [BSV20] have shown that factors of an s-sparse polynomial of individual degree d, have their sparsity bounded by $s^{O(d^2 \log n)}$. Currently, this is the best known bound for factor-sparsity when $d \geq 3$. For restricted classes of symmetric polynomials, Bisht and Saxena [BS21] recently improved this bound to $s^{O(d^2 \log d)}$.

In [GK85], another problem was posed alongside the sparse factorization problem, in the hope that it might be easier. This problem is referred to as *testing sparse factorization*. Given $m + 1$ sparse polynomials $f, g_1, \ldots, g_m$, it asks to test whether $f = g_1 \cdot \ldots \cdot g_m$. The work of [SSS13] gives a polynomial-time algorithm for this problem, in the special case where every g_i is a sum of univariate polynomials. [Vol17] gives a polynomial-time algorithm when f (and therefore every g_i) has constant individual degree and each g_i is an irreducible polynomial. Our PIT result is connected to this problem. In Theorem 1, we give a polynomial-time algorithm to test whether $\prod_{i=1}^{r} f_i = \prod_{j=1}^{m} g_j$, where each f_i, g_j is a sparse polynomial with constant individual degree. Note that now LHS is also a product of polynomials. Moreover, there is no restriction placed on g_j-s except that they have bounded individual degree.

The depth-4 $\Sigma\Pi\Sigma\Pi$ circuit class (see Appendix A for definition) is extremely important in the context of the PIT problem, as it is known that a polynomial-time black-box PIT for this class implies a quasi-polynomial-time black-box PIT for general VP circuits [AV08, AGS19]. For a long time, no PIT algorithm better than the trivial $d^{O(n)}$ time algorithm was known for this class, until the recent breakthrough result of Limaye et al. [LST22], which gives a sub-exponential time algorithm. Various restricted versions of depth-4 circuits are studied to get close to polynomial-time PIT algorithms. For example, Peleg and Shpilka [PS21] give a polynomial-time PIT algorithm for $\Sigma^{[3]}\Pi\Sigma\Pi^{[2]}$ circuits, where the top fan-in is 3 and the bottom fan-in is 2. Recently, Dutta et al. [DDS21] gave a quasi-polynomial-time PIT for $\Sigma^{[k]}\Pi\Sigma\Pi^{[d]}$ circuits, where the top fan-in k and bottom fan-in d are allowed to be any fixed constants. In this model, the restriction on bottom fan-in implies that the bottom $\Sigma\Pi$ computes polynomials of total degree at most d. We give polynomial-time PIT algorithm for $\Sigma^{[2]}\Pi\Sigma\Pi^{[\deg_{x_i} \leq d]}$ model, where the top fan-in is 2 and the bottom $\Sigma\Pi$ computes polynomials with *individual* degree at most d. We note that the individual degree restriction

is much weaker than the total degree restriction. Indeed, even for the case of individual degree bounded by 1 (i.e. multilinear polynomials) the total degree can still be $\Omega(n)$! [SV18] gave a polynomial-time PIT algorithm for the class of multilinear $\Sigma^{[k]}\Pi\Sigma\Pi$ circuits, with constant top fan-in k, where every gate in the circuit computes a multilinear polynomial.

Another related problem is that of *divisibility testing*, which gives two multivariate polynomials f and h and asks to decide whether h divides f. [For15] gives a quasi-polynomial-time algorithm when f is sparse and h is a quadratic polynomial (and hence also sparse). We note that the quadratic restriction on h is much stronger than a constant individual degree restriction, although there is no constant degree restriction for f here. [Vol17] gives a polynomial-time algorithm when both f, h are sparse and have constant individual degree. In the proof of Corollary 1.4, we solve a 'search' version of the divisibility testing problem, i.e. we actually compute f/h in quasi-polynomial time, when f is sparse with constant individual degree and h is a multilinear factor of f.

1.3 Our Techniques & Proof Ideas

1.3.1 Identity Testing for $\Sigma^{[2]}\Pi\Sigma\Pi^{[\deg_{x_i}\leq d]}$ Circuits

Let $C = \prod_{i=1}^{r} g_i + \prod_{j=1}^{m} h_j$ where g_i-s and h_j-s are s-sparse polynomials in $\mathbb{F}[x_1, x_2, \ldots, x_n]$ of individual degree at most d. Clearly, if $C \equiv 0$ then it will evaluate to zero on any input. Now suppose $C \not\equiv 0$. Our goal is to find a point $\mathbf{a} \in \mathbb{F}^n$ such that $C(\mathbf{a}) \neq 0$. Our approach relies on the uniqueness of factorization property of the ring of multivariate polynomials. Specifically, we have that

$$\prod_{i=1}^{r} g_i \neq -\prod_{j=1}^{m} h_j$$

Consequently, wlog there exists a factor f of the LHS that does not divide the RHS. Our goal is to preserve this "situation" while reducing the number of variables. Clearly, a random projection will be sufficient. However, we wish to obtain a deterministic algorithm.

To this end, we are looking for a projection that does not introduce new dependencies between factors. That is, for every i, j: if $v \mid g_i$ and $u \mid h_j$ and $\gcd(u, v) = 1$ we need to ensure that $\gcd(u', v') = 1$, when u' and v' are the projections of u and v, respectively. The main tool for that is the *resultant*. Indeed, one of the fundamental properties of the resultant is that

$$\mathrm{Res}(A, B) \not\equiv 0 \text{ if and only if } \gcd(A, B) = 1.$$

In the multivariate setting, this condition translates into:

$$\forall x_k : \mathrm{Res}_{x_k}(u, v) \not\equiv 0 \iff \mathrm{Res}_{x_k}(u', v') \not\equiv 0.$$

In other words, we need to hit all the resultants of the form $\mathrm{Res}_{x_k}(u, v)$ when $v \mid g_i$ and $u \mid h_j$. By definition, $\mathrm{Res}_{x_k}(u, v)$ is a determinant of $2d \times 2d$ matrix where each entry is a coefficient of u or

v. Hence, $\mathrm{Res}_{x_k}(u,v)$ is $t^{O(d)}$-sparse polynomial with individual degree at most $O(d^2)$, where t is an upper bound on the sparsities of u and v. Consequently, we can use a hitting set generator for sparse polynomials (e.g. [KS01]) to hit the resultant. As u and v are factors of s-sparse polynomials of individual degree d, the best upper by [BSV20] will be $t = s^{O(d^2 \log s)}$. This will result in a quasi-polynomial-time algorithm. Another idea would be to use the multiplicative properties of the resultant and hit $\mathrm{Res}_{x_k}(h_j, g_i)$ instead. Indeed,

$$\mathrm{Res}_{x_k}(h_j, g_i) \not\equiv 0 \implies \mathrm{Res}_{x_k}(u,v) \not\equiv 0$$

and since g_i and h_j are s-sparse this would get a polynomial-time algorithm. The main issue is that we could have $\mathrm{Res}_{x_k}(u,v) \not\equiv 0$ while $\mathrm{Res}_{x_k}(h_j, g_i) \equiv 0$. For example, if $h_j = au$ and $g_i = av$ for the same polynomial a.

Going back, one may ask whether we could show a better sparsity bound on $\mathrm{Res}_{x_k}(u,v)$. While we do not quite do that, we instead show that $\mathrm{Res}_{x_k}(u,v)$ is a *factor* of an $s^{O(d)}$-sparse polynomial with individual degree at most $O(d^2)$. As the ring of polynomials forms an integral domain, this allows us to use a polynomial-size hitting set generator. Formally, we show that if $A, B \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ are two polynomials such that $A = f \cdot g$ and $B = h \cdot g$ then for each x_k there exists an $s^{O(d)}$-sparse polynomial T_k (and a polynomial W_k) such that:

$$T_k = g \cdot \mathrm{Res}_{x_k}(f,h) \cdot W_k$$

We believe that this explicit relation could of interest in its own right. For a more detailed version of the result, see Theorem 2.

1.3.2 Exact Power Testing

Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be an s-sparse polynomial of constant individual degree d. We show how to test whether $f = g^e$, for some other polynomial $g \in \mathbb{F}[x_1, \ldots, x_n]$ and some $e \in \mathbb{N}$. We utilize the notion of *reverse-monic* polynomials for this result. We call a polynomial h reverse-monic, if there exists some $i \in [n]$, such that $h|_{x_i=0} = 1$ (see Definition 5.1). If our input polynomial f is reverse-monic, we show that g is $s^{O(d)}$-sparse. Moreover, we also get an algorithm to compute this exact root g. We prove this in Lemmas 4.4 & 4.9 using a formal expansion that can be thought of as a generalization of the Binomial Expansion:

$$(1+x)^{\frac{1}{e}} = \sum_{i=0}^{\infty} \binom{\frac{1}{e}}{i} x^i.$$

In general though, our input polynomial f may not be reverse-monic. We first convert f into a reverse-monic polynomial $\hat{f}$ with respect to some variable x_i, using a known standard transformation (see Definition 4.10). This step only incurs a slight sparsity blow-up of s^d. One important property of this transformation is that it preserves the "exact power" structure. That is, if $f = g^e$,

then $\hat{f} = h^e$, for some polynomial h. We then compute this e-th root of the reverse-monic $\hat{f}$, as mentioned previously.

However, we are still not quite done. It can happen that a polynomial f which was not an exact power, may become an exact power after the reverse-monic transformation. We need an additional condition to get the converse implication. We show that if both $\hat{f}$ and $f_{|_{x_i=0}}$ are exact powers, then we can correctly conclude that f is also an exact power (Claim 4.12). This gives us a recursive algorithm, as $f_{|_{x_i=0}}$ is a polynomial in $(n-1)$ variables. This procedure is described formally in Algorithm 3.

1.3.3 Co-Factor Sparsity Bound

For the co-factor bounds, our results build on the division elimination techniques of [Str73]. Let us outline our approach. To this end, let $f, h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be s-sparse polynomials such that $h(0, \ldots, 0) = 1$ and suppose that $f = gh$ for some polynomial $g \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ with individual degree at most d. Consider the following formal expansion: $\frac{1}{(1-x)} = \sum_{j=0}^{\infty} x^j$. Then we have:

$$g = \frac{f}{h} = \frac{f}{(1-(1-h))} = \sum_{j=0}^{\infty} f(1-h)^j$$

when the equality is an equality of formal sums of monomials. The key observation is that $(1-h)$ does not contain any constants, hence total degree of every monomial in $(1-h)^j$ (and hence in every summand $f(1-h)^j$) is at least j. Consequently, we can "discard" the tail $\sum_{j=dn+1}^{\infty} f(1-h)^j$ since every monomial in g has a total degree of at most dn. Indeed, g will be formed by a subset of monomials of $\sum_{j=0}^{dn} f(1-h)^j$. This allows us to obtain an upper bound on the sparsity of g:

$$\|g\| \leq \sum_{j=0}^{dn} s^{j+1} \leq s^{dn+2}.$$

Clearly, the outlined approach has two major flaws:

1. It requires that $h(0, \ldots, 0) = 1$ (or more generally, $h(0, \ldots, 0) \neq 0$). And even then:
2. The obtained bound is exponential in n.

One way to address the former is by a random shift to the variable. However, this may significantly increase **both** the sparsity and the individual degree! We take a different approach. Our main observation is that the argument still works if we treat the polynomials as polynomials in "fewer" variables.

Formally, let $I \subseteq [n]$ of size $|I| = k$. We can regard the polynomials as polynomials in the variables x_I with coefficient in the remaining variables. In particular, suppose that $h|_{x_I=0_I} = 1$. In

this case we say that h is *I-reverse monic*. Observe that every monomial in $(1-h)^j$ contains at least one variable from x_I. That is, the total x_I-degree of $(1-h)^j$ is at least j and hence (as before) we can discard the tail. Yet now, g depends "only" on k variables and thus its "total" degree is kd (and not nd). This way we obtain a better upper bound on the sparsity of g, if k is "small":

$$\|g\| \leq \sum_{j=0}^{kd} s^{j+1} \leq s^{kd+2}.$$

Of course, our approach still relies on the assumption that h is I-reverse monic for a "small" subset I. Although we are unable to lift this assumption, we can weaken it. As was noted earlier, if $h(0,\ldots,0) = \alpha \neq 0$ (i.e. when $I = [n]$) we can just divide by α as it is a field element. However, this is no longer possible for an arbitrary I (especially, if I is a small set). Yet, we observe that if $h|_{x_I=0_I} = \alpha$ and α is a non-zero *single monomial* (in the remaining variables) we can transform h into an I-reverse monic polynomial $\hat{h}$ with the exact same sparsity. The idea is to apply the transformation $x_i = \alpha \cdot x_i$ for all $i \in I$. Note that since α is a single monomial, this transformation is reversible. Indeed, there is an 1-1 correspondence between the monomials of h and $\hat{h}$. Given this connection, we refer to such h as I-reverse *pseudo*-monic.

Our final ingredient is (yet) another observation that for multilinear polynomials we can weaken the assumption that h is I-reverse *pseudo*-monic further by considering *unique projections*. That is, monomials that have a "unique pattern". Formally, we want h to have exactly one monomial that contains the submonomial: $x_{i_1}^{e_1} x_{i_2}^{e_2} \cdots x_{i_k}^{e_k}$. We show that by "flipping" the variables in h we can transform it into another multilinear polynomial $\tilde{h}$ which is $\{i_1, i_2, \ldots, i_k\}$-reverse pseudo-monic. As a result, $\|g\| \leq s^{kd+2}$.

This is our main conceptual contribution: the upper bound on the sparsity of a multilinear co-factor g is governed by a *combinatorial property* of the set of monomials of h: the length of the shortest unique projection. As an application, we show that every s-sparse polynomial has a unique projection of length at most $\log s + 1$, thus we obtain a new, slightly stronger, sparsity bound on co-factors of multilinear polynomials.

1.4 Multilinear co-Factor Motivation

Theorem 4 and Corollary 1.4 in this paper apply to the factorization scenario of $f = gh$ where f is s-sparse and h is multilinear. First of all, note that by previous results (see [BSV20] and references within) h itself is s-sparse. So we are looking to bound the sparsity of g. As it turns out, this pattern is the "bottleneck" case for multicubic polynomials. In other words, showing a polynomial-size sparsity bound on g in this scenario would imply a polynomial-size sparsity bound on factors of general multicubic polynomials! In fact, it is sufficient to consider the case when the degree of g in every variable is exactly 2! We remark that getting polynomial-size sparsity bound is open for $d \geq 3$. The following lemma summarizes this formally.

Lemma 1.5. *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be an arbitrary s-sparse multicubic polynomial such that $f = gh$ and h is multilinear. Suppose there exists a universal constant $a \geq 1$ such that for any f as above, the co-factor g is s^a-sparse. Then any factor of any multicubic polynomial is s^a-sparse.*

Proof. We prove the following claim: Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be an s-sparse multicubic polynomial such that $f = uv$ and $v \not\equiv 0$. Then u is s^a-sparse. Note that the claim also covers the case when $f = u \equiv 0$.

The proof is by induction on n (the number of variables in f). The base case is when $n = 0$ (i.e. $u, f, v \in \mathbb{F}$) where the claim follows trivially. Suppose $n \geq 1$. We have the following cases to consider:

- There exists a variable x_i s.t. $\deg_{x_i}(u) \geq 1$ but $\deg_{x_i}(v) = 0$. Let $1 \leq d \leq 3$ be the degree of x_i in u. In this case we can write:

  $$(u_d x_i^d + \ldots + u_0)v = uv = f = f_d {x_i}^d + \ldots + f_0.$$

  Here, u_j, f_j and v do not depend on x_i. Formally: $f_j = u_j v$ for $j \in \{0, \ldots, d\}$. By the induction hypothesis, we have that $\|u_j\| \leq \|f_j\|^a$ for $j \in \{0, \ldots, d\}$ and hence:

  $$\|u\| = \sum_{j=0}^{d} \|u_j\| \leq \sum_{j=0}^{d} \|f_j\|^a \leq \left(\sum_{j=0}^{d} \|f_j\| \right)^a = \|f\|^a.$$

- There exists a variable x_i s.t. $\deg_{x_i}(v) \geq 1$, but $\deg_{x_i}(u) = 0$. Pick $\alpha \in \mathbb{F}$ such that $v|_{x_i=\alpha} \not\equiv 0$. We have that:

  $$u \cdot v|_{x_i=\alpha} = u|_{x_i=\alpha} \cdot v|_{x_i=\alpha} = f|_{x_i=\alpha}.$$

  By the induction hypothesis: $\|u\| \leq \|f|_{x_i=\alpha}\|^a \leq \|f\|^a$.

- There exists a variable x_i s.t. $\deg_{x_i}(u) = 1$. Wlog $\deg_{x_i}(v) \geq 1$. We can write

  $$(u_1 x_i + u_0)(v_d x_i^d + \ldots + v_e x_i^e) = uv = f = (f_{d+1} x_i^{d+1} + \ldots + f_e x_i^e).$$

  Here, $d > e$ and $v_d, v_e \not\equiv 0$. In particular, we have that $u_1 v_d = f_{d+1}$ and $u_0 v_e = f_e$. By the induction hypothesis: $\|u\| = \|u_1\| + \|u_0\| \leq \|f_{d+1}\|^a + \|f_e\|^a \leq \|f\|^a$.

- WLOG we are left with the case that for each $i \in [n]$ we have that: $\deg_{x_i}(u) = 2$ and $\deg_{x_i}(v) = 1$. Based on our assumption, in this case $\|u\| \leq \|f\|^a$ and we are done.

□

2 Preliminaries

Notations: We use the shorthand $[n]$ for the set $\{1, 2, \ldots, n\}$. We denote a vector $v = (v_1, \ldots, v_n)$ in short by $\mathbf{v}$ (as a column vector). We denote the n-fold Cartesian product of a set H by H^n. We will use $\log x$ for $\log_2 x$. We use the $\stackrel{\Delta}{=}$ symbol for definition.

Let $f \in \mathbb{F}[\mathbf{x}]$ be an n-variate polynomial. The *individual degree* of a variable x_i in f, denoted by $\deg_{x_i}(f)$, is defined as the maximum degree of that variable in f, while the *individual degree* of f is the maximum among all the individual degrees, $\max_{i\in[n]} \deg_{x_i}(f)$. We will use $\mathbf{x^e}$ to denote the monomial $x_1^{e_1}x_2^{e_2}\cdots x_n^{e_n}$. We define $\text{coeff}(\mathbf{x^e})(f)$ as the coefficient of monomial $\mathbf{x^e}$ in polynomial f. We define *support* of f as $\text{supp}(f) = \{\mathbf{e} \mid \text{coeff}(\mathbf{x^e})(f) \neq 0\}$. We define the *sparsity* of f as the number of non-zero terms in f. Let us denote *sparsity* of f as $\|f\|$, which is the same as $|\text{supp}(f)|$.

For a set $I \subseteq [n]$, we use x_I to denote the set of variables $\{x_i \mid i \in I\}$ and $x_{[n]\setminus I}$ to denote the set of remaining variables. We use the symbol $f|_{x_I=0_I}$ to denote the polynomial resulting from substituting 0 at all the x_I variables in f. For two polynomials g, h, we use the symbol $\gcd(g, h)$ to denote their greatest common divisor.

Using these notations, we can formulate the following known result, which can be found, for example in [BSV20].

Lemma 2.1. *Let $f, h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ where h is a multilinear polynomial and $h \mid f$. Then $\|h\| \leq \|f\|$.*

We state another lemma which is useful in extracting the coefficients when a given multivariate polynomial f is expressed as a polynomial w.r.t. a single variable. It follows from the standard trick of polynomial interpolation.

Lemma 2.2 (Folklore)**.** *Let $f \in \mathbb{F}[\mathbf{x}, y]$ be an s-sparse polynomial such that $\deg_y(f) = d$. Let $f = \sum_{i=0}^{d} f_i \cdot y^i$, where each $f_i \in \mathbb{F}[\mathbf{x}]$. Then, we can compute the coefficients $(f_0, \ldots, f_d)$ in $\text{poly}(s, n, d)$ $\mathbb{F}$-operations.*

3 PIT for $\Sigma^{[2]}\Pi\Sigma\Pi^{[\deg_{x_i}\leq d]}$ Circuits

In this section we prove Theorems 1 and 2. We refer the reader to Appendix A for the formal definition of an algebraic circuit and PIT algorithm. For the purpose of black-box PIT, we have the notion of hitting set generators (HSG) or simply generator in short.

Definition 3.1 (Generator)**.** *Let $\mathcal{C}$ be a class of n-variate polynomials. Consider $\mathcal{G} = (g_1, g_2, \ldots, g_n) : \mathbb{F}^k \to \mathbb{F}^n$, an n-tuple of k-variate polynomials where for each $i \in [n]$, $g_i \in \mathbb{F}[t_1, t_2, \ldots, t_k]$. Let $f(x_1, \ldots, x_n)$ be an n-variate polynomial. We define action of $\mathcal{G}$ on polynomial f by $f \circ \mathcal{G} = f(g_1, \ldots, g_n) \in \mathbb{F}[t_1, \ldots, t_k]$. We call $\mathcal{G}$ a k-seeded generator for class $\mathcal{C}$ if for every non-zero $f \in \mathcal{C}$, $f \circ \mathcal{G} \neq 0$. Degree of generator $\mathcal{G}$ is defined as $\deg(\mathcal{G}) \stackrel{\Delta}{=} \max\{\deg(g_i)\}_{i=1}^{n}$.*

For a polynomial-time PIT algorithm, k is kept constant. A generator $\mathcal{G}$ acts as a variable reduction map which converts an input polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ to $f \circ \mathcal{G} \in \mathbb{F}[t_1, \ldots, t_k]$ such that $f = 0$ if and only if $f \circ \mathcal{G} = 0$. Let D be the degree of $\mathcal{G}$, which makes $f \circ \mathcal{G}$ a polynomial of individual degree dD, where d is the individual degree of f. Thus, $\mathcal{G}$ gives us a hitting-set of size $(dD+1)^k$ by brute-force derandomization for $f \circ \mathcal{G}$ using PIT Lemma [Sch80, Zip79, DL78, Ore22]. In other words, we get a polynomial-time black-box PIT algorithm for f when k is constant, $\mathcal{G}$ can be designed in polynomial time and its degree is also polynomially bounded. See [SY10] for more on equivalence between hitting-sets and generators.

3.1 The $\Sigma^{[k]}\Pi\Sigma\Pi^{[\deg_{x_i} \leq d]}$ Model

A size s, depth-4 $\Sigma\Pi\Sigma\Pi$ circuit computes a polynomial of the form $f = \sum_{i=1}^{k} \prod_{j=1}^{m} f_{ij}$, where f_{ij} are s-sparse polynomials for each $i \in [k], j \in [m]$. For $s = \text{poly}(n)$, [LST22] gives the first deterministic sub-exponential time PIT for constant-depth (depth-4 also) which runs in $(sn)^{O(n^{\mu})}$-time, where $\mu > 0$ is any real number. While a polynomial-time PIT algorithm for general depth-4 circuit continues to be elusive, various restricted versions of this model have been attacked. One such restriction is to make the top fan-in k constant. For $k = 2$, even white-box PIT for $\Sigma^{[2]}\Pi\Sigma\Pi$ circuits is still open. A more restricted model is the class of $\Sigma^{[k]}\Pi\Sigma\Pi^{[d]}$ circuits, where the top fan-in k and the bottom fan-in d are constants. For a size-s circuit of this class, f_{ij}'s are s-sparse polynomials of constant total degree at most d. Even this restricted model seems to be quite non-trivial. Only very recently, [DDS21] gave a quasi-polynomial-time black-box PIT algorithm for this model. For $k = 3$ and $d = 2$ (f_{ij}'s are quadratic polynomials), [PS21] give a polynomial-time black-box PIT algorithm. For $k = 3$ and $d > 2$, coming up with a polynomial PIT algorithm remains an open question.

We now introduce, what we call the $\Sigma^{[k]}\Pi\Sigma\Pi^{[\deg_{x_i} \leq d]}$ model. In the $\Sigma^{[k]}\Pi\Sigma\Pi^{[d]}$ model, the sparse polynomials f_{ij}'s have constant total degree $\leq d$. We relax this restriction to f_{ij}'s being constant *individual* degree $\leq d$ polynomials in $\Sigma^{[k]}\Pi\Sigma\Pi^{[\deg_{x_i} \leq d]}$ model. This is a more general model, since f_{ij}'s can now have much higher total degree, like $O(n)$. In Section 3.3, we give a deterministic polynomial-time black-box PIT algorithm for this model when $k = 2$ and d is any constant. We also note that our PIT algorithm works for any field $\mathbb{F}$, while the works of [PS21, DDS21] do have certain field restrictions.

3.2 GCD, Resultants and Subresultants

Before we jump to our PIT algorithm, we discuss the classical tool of resultants and subresultants in this section. We prove an interesting connection in Theorem 3.9, that we will crucially use in our PIT algorithm later.

The polynomial ring $\mathbb{F}[x_1, \ldots, x_n]$ is a unique factorization domain (UFD). Hence, the gcd of two polynomials is well defined up to a multiplication by field element. We can also define gcd

with respect to a single variable x_i, where we treat rest of the variables as field elements. That is, $\gcd_{x_i}(f,g)$ is well defined up to multiplication by a rational function depending on the remaining variables. By convention we choose the normalized gcd. For example, let $f = x_1^2x_2 + x_1x_2^2$ and $g = x_1^2x_2^2$, then $\gcd(f,g) = x_1x_2$ while $\gcd_{x_2}(f,g) = x_2$. Technically, the former is gcd in $\mathbb{F}[x_1,x_2]$ and the latter is normalized gcd in $\mathbb{F}(x_1)[x_2]$.

Let $A(y), B(y) \in \mathcal{R}[y]$ be two non-zero polynomials with y-degree d and e respectively in an arbitrary UFD $\mathcal{R}$. Suppose $A(y) = \sum_{i=0}^{d} a_i \cdot y^i$ and $B(y) = \sum_{j=0}^{e} b_j \cdot y^j$. Consider the $(d+e) \times (d+e)$ *Sylvester matrix* M whose first e rows are the e shifts of the row vector $(a_d, \ldots, a_0, 0, \ldots, 0)$ and next d rows are the d shifts of the row vector $(b_e, \ldots, b_0, 0, \ldots, 0)$.

$$M = \begin{bmatrix} a_d & a_{d-1} & \ldots & a_1 & a_0 & & & \\ & a_d & a_{d-1} & \ldots & a_1 & a_0 & & \\ & & \ldots & \ldots & \ldots & \ldots & & \\ & & & a_d & a_{d-1} & \ldots & a_1 & a_0 \\ b_e & b_{e-1} & \ldots & b_1 & b_0 & & & \\ & b_e & b_{e-1} & \ldots & b_1 & b_0 & & \\ & & \ldots & \ldots & \ldots & \ldots & & \\ & & & b_e & b_{e-1} & \ldots & b_1 & b_0 \end{bmatrix}.$$

Definition 3.2 (Resultant)**.** *The resultant* $\mathrm{Res}_y(A,B) \in \mathcal{R}$ *is defined to be the determinant of this Sylvester matrix. That is,* $\mathrm{Res}_y(A,B) = \det(M)$.

In our setting, $\mathcal{R}$ will be a polynomial ring, say $\mathbb{F}[x_1,\ldots,x_n]$ and $\mathrm{Res}_y(A,B)$ will be a polynomial free of y-variable.

Lemma 3.3 (Properties of resultant)**.** *Let* $A, B \in \mathbb{F}[y, x_1, \ldots, x_n]$ *be two s-sparse polynomials with individual degrees at most d. Then,*

1. $\mathrm{Res}_y(A,B) \in \mathbb{F}[x_1,\ldots,x_n]$ *is an* $(2ds)^{2d}$*-sparse polynomial with individual degrees at most* $2d^2$.
2. *For every* $\mathbf{a} \in \mathbb{F}^n$: $\mathrm{Res}_y(A|_{\mathbf{x}=\mathbf{a}}, B|_{\mathbf{x}=\mathbf{a}}) = \mathrm{Res}_y(A,B)(\mathbf{a})$.
3. $\gcd_y(A,B) \neq 1$ *iff* $\mathrm{Res}_y(A,B) \equiv 0$. *That is, A and B share a non-trivial gcd w.r.t. variable y ($\deg_y(\gcd(A,B)) > 0$) iff resultant of A, B w.r.t. y is 0.*

We now study few useful sub-matrices of the Sylvester matrix below.

Definition 3.4 (j-th principal resultant)**.** *Let M_j be the submatrix of M formed by deleting last j rows of A terms, last j rows of B terms and the last $2j$ columns. We call M_j to be the j-th principal resultant of A and B.*

Note that $\mathrm{Res}_y(A,B) = M = M_0$.

Lemma 3.5 (Thm 7.3 of [GCL92]). *Let $A(y), B(y) \in \mathcal{R}[y]$ be two non-zero polynomials. Then, degree of y in $\gcd_y(A, B)$ equals to the smallest $j \geq 0$ such that $\det(M_j) \neq 0$.*

We can now define the subresultant polynomial as follows.

Definition 3.6 (Subresultant). *Let M_{ij} be the $(d + e - 2j) \times (d + e - 2j)$ submatrix of Sylvester matrix M formed by deleting:*

- *rows $e - j + 1$ to e (each having coefficients of $A(y)$),*
- *rows $d + e - j + 1$ to $d + e$ (each having coefficients of $B(y)$),*
- *columns $d + e - 2j$ to $d + e$, except for column $d + e - i - j$.*

Note that the j-th principal resultant M_j is exactly M_{jj}.

For $0 \leq j \leq e$, the j-th subresultant of $A(y), B(y) \in \mathcal{R}[y]$ is the polynomial in $\mathcal{R}[y]$ of degree j defined by

$$S_y(j, A, B) = \det(M_{0j}) + \det(M_{1j}) \cdot y + \ldots + \det(M_{jj}) \cdot y^j.$$

By Definition 3.6, Lemma 3.5 and Lemma 3.3, we get the following useful properties for subresultant.

Lemma 3.7 (Properties of subresultant). *Let $A, B \in \mathbb{F}[y, x_1, \ldots, x_n]$ be two s-sparse polynomials with individual degrees at most d. Then,*

1. *$S_y(j, A, B) \in \mathbb{F}[y, x_1, \ldots, x_n]$ is an $(2ds)^{2d+1}$-sparse polynomial with individual degrees at most $2d^2$.*
2. *For every $\mathbf{a} \in \mathbb{F}^n$: $S_y(j, A|_{\mathbf{x}=\mathbf{a}}, B|_{\mathbf{x}=\mathbf{a}}) = S_y(j, A, B)(\mathbf{a})$.*
3. *$\gcd_y(A, B)$ has y-degree j iff $\deg_y(S(j, A, B)) = j$.*

We state below a known result in the theory of subresultants, which will be useful for us.

Lemma 3.8 (Lem 7.1 of [GCL92]). *Let $A(x), B(x) \in \mathcal{R}[x]$ be two polynomials over an arbitrary UFD $\mathcal{R}$. Suppose*

$$A(x) = Q(x) \cdot B(x) + R(x)$$

with $\deg(A) = m$, $\deg(B) = n$, $\deg(Q) = m - n$, $\deg(R) = k$ and $m \geq n > k$. Let b and r denote the leading coefficients of $B(x)$ and $R(x)$ respectively. Then

$$S_x(j, A, B) = (-1)^{(m-j)(n-j)} \begin{cases} b^{m-k} \cdot S_x(j, B, R) & 0 \leq j < k \\ b^{m-k} \cdot r^{n-k-1} \cdot R(x) & j = k \\ 0 & k < j < n - 1 \\ b^{m-n+1} \cdot R(x) & j = n - 1. \end{cases}$$

Theorem 2 is a special case of the following result which, we believe, could be interesting in its own right.

Theorem 3.9. *Let $A(x), B(x) \in \mathcal{R}[x]$ be two polynomials over an arbitrary UFD $\mathcal{R}$. Suppose $A(x) = f(x) \cdot g(x)$ and $B(x) = h(x) \cdot g(x)$ with $\deg(A) = m, \deg(B) = n, \deg(g) = d, \deg(f) = m' = m - d$ and $\deg(h) = n' = n - d$. Then*

$$S_x(d, A, B) = g \cdot \mathrm{Res}_x(f, h) \cdot \mathrm{lc}(g)^{m'+n'-1}$$

Proof. Consider Euclidean division of A by B so that we get $A(x) = Q(x) \cdot B(x) + R(x)$, for some polynomials Q, R such that $\deg(R) < \deg(B)$. Note that since g divides both A and B, it must also divide R. Therefore, $R = g \cdot p$ for some polynomial $p(x)$. Thus, we also get

$$f(x) = Q(x) \cdot h(x) + p(x) \tag{3.10}$$

Let $\deg(R) = k$ for some $k < n$ and let $\deg(p) = k' = k - d$. Now, we prove the theorem by induction on $\deg(p)$.

Base case: $\deg(p) = k' = 0$. In other words, $\deg(R) = k = d$. Thus using second case of Lemma 3.8, we get that:

$$\begin{aligned}
S_x(d, A, B) &= (-1)^{(m-d)(n-d)} \cdot b^{m-k} \cdot r^{n-k-1} \cdot R \\
&= (-1)^{m'.n'} \cdot \mathrm{lc}(h)^{m-k} \cdot \mathrm{lc}(g)^{m-k} \cdot \mathrm{lc}(p)^{n-k-1} \cdot \mathrm{lc}(g)^{n-k-1} \cdot pg \\
&= (-1)^{m'.n'} \cdot g \cdot \mathrm{lc}(h)^{m-k} \cdot \mathrm{lc}(p)^{n-k} \cdot \mathrm{lc}(g)^{m+n-2k-1} \\
S_x(d, A, B) &= (-1)^{m'.n'} \cdot g \cdot \mathrm{lc}(h)^{m-k} \cdot \mathrm{lc}(p)^{n-k} \cdot \mathrm{lc}(g)^{m'+n'-1}
\end{aligned} \tag{3.11}$$

The second last step above follows because $p = \mathrm{lc}(p)$ when $\deg(p) = 0$. Now, we shall compute $\mathrm{Res}_x(f, h)$. Note that $\mathrm{Res}_x(f, h) = S_x(0, f, h)$ by definition of subresultant. Considering (3.10) with $\deg(p) = 0$, we can use second case of Lemma 3.8 to get:

$$\begin{aligned}
S_x(0, f, h) &= (-1)^{(\deg(f)-0).(\deg(h)-0)} \cdot \mathrm{lc}(h)^{\deg(f)-\deg(p)} \cdot \mathrm{lc}(p)^{\deg(h)-\deg(p)-1} \cdot p \\
&= (-1)^{m'.n'} \cdot \mathrm{lc}(h)^{m'} \cdot \mathrm{lc}(p)^{n'-1} \cdot p \quad [\text{as } \deg(p) = 0] \\
&= (-1)^{m'.n'} \cdot \mathrm{lc}(h)^{m'} \cdot \mathrm{lc}(p)^{n'} \quad [\text{as } p = \mathrm{lc}(p)] \\
\mathrm{Res}_x(f, h) &= (-1)^{m'.n'} \cdot \mathrm{lc}(h)^{m-k} \cdot \mathrm{lc}(p)^{n-k}
\end{aligned} \tag{3.12}$$

(3.11) and (3.12) together yield $S_x(d, A, B) = g \cdot \mathrm{Res}_x(f, h) \cdot \mathrm{lc}(g)^{m'+n'-1}$ for the base case.

Induction step: Now, we assume $\deg(p) = k' > 1$. In other words, $\deg(R) = k > d$. Therefore, by first case of Lemma 3.8:

$$S_x(d, A, B) = (-1)^{(m-d)(n-d)} \cdot b^{m-k} \cdot S_x(d, B, R)$$

$$= (-1)^{m'.n'} \cdot \mathrm{lc}(h)^{m-k} \cdot \mathrm{lc}(g)^{m-k} \cdot S_x(d, B, R) \tag{3.13}$$

Now consider Euclidean division of B by R to get

$$B(x) = Q'(x) \cdot R(x) + R'(x) \tag{3.14}$$

for some polynomial $R'(x)$ with $\deg(R') < deg(R)$. Since g divides both B and R, we deduce that g must also divide R'. Let $R' = g \cdot p'$ for some polynomial p'. Thus from (3.14), we also get

$$h(x) = Q'(x) \cdot p(x) + p'(x) \tag{3.15}$$

In (3.14) since $\deg(R') < \deg(R)$ or equivalently $\deg(p') < \deg(p)$, we can use induction hypothesis to deduce that,

$$S_x(d, B, R) = g \cdot \mathrm{Res}_x(h, p) \cdot \mathrm{lc}(g)^{n'+k'-1} \tag{3.16}$$

Note that $\deg(p) = k' > 0$ in induction step, thus we can use first case of Lemma 3.8 on (3.10) to get

$$\begin{aligned}
\mathrm{Res}_x(f, h) &= S_x(0, f, h) \\
&= (-1)^{(\deg(f)-0)(\deg(h)-0)} \cdot \mathrm{lc}(h)^{\deg(f)-\deg(p)} \cdot S_x(0, h, p) \\
&= (-1)^{m'.n'} \cdot \mathrm{lc}(h)^{m'-k'} \cdot \mathrm{Res}_x(h, p) \\
\mathrm{Res}_x(h, p) &= \frac{\mathrm{Res}_x(f, h)}{(-1)^{m'.n'} \cdot \mathrm{lc}(h)^{m'-k'}}.
\end{aligned} \tag{3.17}$$

Substituting (3.17) in (3.16), we get:

$$S_x(d, B, R) = g \cdot \frac{\mathrm{Res}_x(f, h)}{(-1)^{m'.n'} \cdot \mathrm{lc}(h)^{m'-k'}} \cdot \mathrm{lc}(g)^{n'+k'-1} \tag{3.18}$$

Substituting (3.18) back into (3.13), we get

$$\begin{aligned}
S_x(d, A, B) &= (-1)^{m'.n'} \cdot \mathrm{lc}(h)^{m-k} \cdot \mathrm{lc}(g)^{m-k} \cdot g \cdot \frac{\mathrm{Res}_x(f, h)}{(-1)^{m'.n'} \cdot \mathrm{lc}(h)^{m'-k'}} \cdot \mathrm{lc}(g)^{n'+k'-1} \\
&= \mathrm{lc}(g)^{m-k} \cdot g \cdot \mathrm{Res}_x(f, h) \cdot \mathrm{lc}(g)^{n'+k'-1} \quad [\text{as } m-k = m'-k'] \\
&= g \cdot \mathrm{Res}_x(f, h) \cdot \mathrm{lc}(g)^{m-k+n'+k'-1} \\
&= g \cdot \mathrm{Res}_x(f, h) \cdot \mathrm{lc}(g)^{m'+n'-1} \quad [\text{as } m-k+k' = m-d = m']
\end{aligned}$$

This completes the proof of induction step, as well as that of the theorem. □

3.3 The PIT Algorithm

We finally get to our PIT algorithm for the circuit class $\Sigma^{[2]}\Pi\Sigma\Pi^{[\deg_{x_i} \leq d]}$. Let f be a polynomial having a size-s circuit in this class. Then, $f = \prod_{i=1}^{r} g_i + \prod_{j=1}^{m} h_j$, where each g_i, h_j is an s-sparse

polynomial of individual degree bounded by d. It is easy to see that a polynomial-time factorization algorithm for sparse polynomials, will then yield a white-box PIT algorithm, since one can factor every g_i and h_j and compare the irreducible factorization of both LHS and RHS in (3.20).

To get a black-box algorithm, we need a more careful argument, for which we use the tool of resultant. If $f \neq 0$, then (3.20) holds and by uniqueness of factorization, there exists a g_i on LHS with an irreducible factor that does not divide any h_j on RHS. Thus, we want a generator $\mathcal{G}$ such that we 'preserve' this irreducible factor, i.e. the irreducible factor of $g_i \circ \mathcal{G}$ does not divide $h_j \circ \mathcal{G}$. This will be our certificate of non-zeroness for $f \circ \mathcal{G}$. Let $g_i = g_i' \cdot q_{ij}$ and $h_j = h_j' \cdot q_{ij}$, where $q_{ij} = \gcd(g_i, h_j)$. In the proof, we show that it suffices to preserve the co-primality of g_i', h_j' for every $j \in [m]$. Therefore, we only need to *hit* the resultant of g_i', h_j' for every $j \in [m]$, i.e. construct a generator $\mathcal{G}$ such that $\mathrm{Res}_{x_\ell}(g_i', h_j') \circ \mathcal{G} \neq 0$, where x_ℓ is some variable in $\mathrm{supp}(g_i')$. Note that g_i', h_j' are not known to be sparse, though they are factors of sparse polynomials. At this point, we can actually invoke [BSV20] to deduce that both g_i' and h_j' are $s^{O(d^2 \log n)}$-sparse and hence $\mathrm{Res}_{x_\ell}(g_i', h_j')$ is $s^{\mathrm{poly}(d) \log n}$-sparse. We can then invoke the sparse PIT algorithm of [KS01] to get an $s^{\mathrm{poly}(d) \log n}$-time (quasi-poly time) black-box PIT.

The above argument also shows that an $s^{\mathrm{poly}(d)}$ factor-sparsity bound will imply a polynomial-time black-box PIT algorithm for this model, for constant d. In our proof, we show how to get a polynomial-time black-box PIT algorithm for this model without even proving a factor-sparsity bound. As discussed earlier, we need a generator that hits $\mathrm{Res}_{x_\ell}(g_i', h_j')$. Even though this resultant polynomial may not be sparse, we show that it is factor of some other sparse polynomial (Theorem 3.9). This polynomial is the subresultant $S_{x_\ell}(k, g_i, h_j)$, which is non-zero for $k = \deg(\gcd(g_i, h_j))$ and is $(sd)^{O(d)}$-sparse by Lemma 3.7. Since a factor of a non-zero polynomial is also non-zero, we can hit the desired resultant by hitting this sparse subresultant.

Theorem 3.19. *Let $\mathcal{C}$ be the class of size-s, n-variate $\Sigma^{[2]}\Pi\Sigma\Pi^{[\deg_{x_i} \leq d]}$ circuits. Then, there is a deterministic black-box PIT algorithm for $\mathcal{C}$ that runs in $\mathrm{poly}((sd)^d, n)$-time.*

Proof. Let $f \in \mathcal{C}$ be a non-zero polynomial of the form $f = \prod_{i=1}^{r} g_i + \prod_{j=1}^{m} h_j$, where $g_i, h_j \in \mathbb{F}[\mathbf{x}]$ are s-sparse polynomials with individual degree d, for each $i \in [r]$ and $j \in [m]$. Then,

$$\prod_{i=1}^{r} g_i \;\neq\; -\prod_{j=1}^{m} h_j \tag{3.20}$$

We wish to design a generator $\mathcal{G}$ such that the inequality in (3.20) is preserved for $f \circ \mathcal{G}$ also. Observe that if inequality (3.20) holds, then by uniqueness of factorization there exists a g_i on LHS with an irreducible factor, say u, such that u does not divide h_j, for every $j \in [m]$ (or vice versa). Let $g_i = g_i' \cdot q_{ij}$ and $h_j = h_j' \cdot q_{ij}$, for some co-prime polynomials g_i' and h_j' such that $q_{ij} \stackrel{\Delta}{=} \gcd(g_i, h_j)$. We claim that a generator $\mathcal{G}$ that preserves the co-primality of g_i' and h_j', will suffice. Formally, we want that for every $j \in [m]$,

$$\gcd(g_i' \circ \mathcal{G}, h_j' \circ \mathcal{G}) \;=\; 1. \tag{3.21}$$

If we know the exact g_i, we only need to preserve (3.21) for all $j \in [m]$, but since we are in the black-box setting, we don't know the exact i beforehand. Therefore, we will later iterate over all $i \in [n]$ also, as done in Algorithm 1. We handle this properly later but for now we proceed assuming that we know the correct g_i.

Note that the irreducible factor u is a part of g_i' and not q_{ij}, since u does not divide h_j. And if (3.21) holds, then $u \circ \mathcal{G}$ does not divide $h_j \circ \mathcal{G}$ for every h_j, otherwise it will contradict the co-primality of $g_i' \circ \mathcal{G}$ and $h_j' \circ \mathcal{G}$.

Let x_ℓ be a variable in the support of u. By Lemma 3.3, $\mathrm{Res}_{x_\ell}(g_i', h_j') \neq 0$ since $\gcd_{x_\ell}(g_i', h_j') = 1$. By Theorem 3.9, this resultant polynomial is a factor of the subresultant polynomial, $S_{x_\ell}(k, g_i, h_j)$, where $k \triangleq \deg(q_{ij}) \leq d$. Since g_i, h_j are s-sparse polynomials, this subresultant is non-zero and $(2ds)^{2d+1}$-sparse by Lemma 3.7.

Let $\mathcal{G} = (g_1, \ldots, g_{\ell-1}, g_{\ell+1}, \ldots, g_n)$ be the generator for the class of $(n-1)$-variate, $(2ds)^{2d+1}$-sparse polynomials (Lemma A.1). We extend $\mathcal{G}$ to a generator $\mathcal{G}_\ell$ as follows:

$$\mathcal{G}_\ell \triangleq (g_1, \ldots, g_{\ell-1}, x_\ell, g_{\ell+1}, \ldots, g_n). \tag{3.22}$$

Then observe that $S_{x_\ell}(k, g_i, h_j) \circ \mathcal{G}_\ell \neq 0$, for any $j \in [m]$, by the sparsity argument above. Thus, we deduce that $\mathrm{Res}_{x_\ell}(g_i', h_j') \circ \mathcal{G}_\ell \neq 0$, since it is a factor of $S_{x_\ell}(k, g_i, h_j) \circ \mathcal{G}_\ell$. Then by Lemma 3.7, we get that $\mathrm{Res}_{x_\ell}(g_i', h_j') \circ \mathcal{G}_\ell = \mathrm{Res}_{x_\ell}(g_i' \circ \mathcal{G}_\ell, h_j' \circ \mathcal{G}_\ell) \neq 0$. This means that $\gcd_{x_\ell}(g_i' \circ \mathcal{G}_\ell, h_j' \circ \mathcal{G}_\ell) = 1$. Hence, there is a factor of $g_i \circ \mathcal{G}_\ell$ (with variable x_ℓ in its support), which does not divide $h_j \circ \mathcal{G}_\ell$, for every $j \in [m]$. By uniqueness of factorization,

$$\prod_{i=1}^{r} (g_i \circ \mathcal{G}_\ell) \;\neq\; -\prod_{j=1}^{m} (h_j \circ \mathcal{G}_\ell) \tag{3.23}$$

Hence, $f \circ \mathcal{G}_\ell \neq 0$. Since $\mathcal{G}_\ell$ is a generator for sparse polynomials, using Lemma A.1, we note that $\mathcal{G}_\ell$ is efficient as it has $\mathrm{poly}((sd)^d, n)$ degree and is of seed length 2.

Recall that we picked l to be a number such that x_ℓ was a variable in support of some g_i'. Since, input polynomial is only given as a black-box, we don't know this variable beforehand. Therefore, we iterate l over all the n variables and get a corresponding generator $\mathcal{G}_\ell$ for each iteration. If for any generator $\mathcal{G}_\ell$, we get that $f \circ \mathcal{G}_\ell \neq 0$, then we deduce that f is non-zero. This process only adds a factor of n in the time complexity. We outline this overall process in Algorithm 1. □

Algorithm 1: black-box PIT algorithm for class $\Sigma^{[2]}\Pi\Sigma\Pi^{[\deg_{x_i}\leq d]}$

Input: A polynomial $f(x_1,\ldots,x_n) \in \Sigma^{[2]}\Pi\Sigma\Pi^{[\deg_{x_i}\leq d]}$, where bottom $\sum\prod$ computes s-sparse polynomials of individual degrees $\leq d$.

Output: ZERO, if f is identically zero and NON-ZERO, otherwise.

```
Call Lemma A.1 to get generator 𝒢 of seed-length 1 for (n−1)-variate polynomials of
 sparsity ≤ (2ds)^{2d+1}.
for i ← 1 to n do
    Construct 𝒢_i of seed-length 2 by inserting an extra (seed) variable x_i at the i-th position
     in 𝒢 as shown in (3.22).
    Compute bivariate polynomial p ← f ∘ 𝒢_i.
    Use brute-force black-box PIT for bivariate p.
    if p ≠ 0 then
        return NON-ZERO.
    end
end
return ZERO.
```

We now analyze correctness and time-complexity of Algorithm 1.

Correctness: By proof argument in Theorem 3.19, $f \neq 0$ iff $\exists i \in [n], f \circ \mathcal{G}_i \neq 0$. Indeed the algorithm outputs ZERO only when $f \circ \mathcal{G}_i = 0$ for every $i \in [n]$, otherwise it outputs NON-ZERO, as desired.

Time complexity: By Lemma A.1, degree of generator $\mathcal{G}_i$ is $\text{poly}((sd)^d, n)$. Hence, testing non-zeroness of the bivariate polynomial $f \circ \mathcal{G}_i$ takes only $\text{poly}((sd)^d, n)$ time. The n iterations add a factor of n and we still get an overall black-box PIT algorithm of time complexity $\text{poly}((sd)^d, n)$.

4 Exact Power Testing

In this section we prove Theorem 3. A polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ is an *exact power* if there exists (another) polynomial $g \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ and $e \in \mathbb{N}$ such that $f = g^e$.

We first show an $s^{O(d)}$ sparsity bound for g, when f is an s-sparse, reverse-monic polynomial of individual degree d (Lemma 4.4). Moreover, we also get an algorithm to compute g for this case in Algorithm 2. For the general case, when f is not reverse-monic, we use a standard transformation to convert it into a reverse-monic polynomial $\hat{f}$ (Definition 4.10). This transformation preserves the exact power structure of f, i.e. $f = g^e$ implies $\hat{f} = h^e$, for some suitable polynomial h. One can then invoke Lemma 4.4 to get a nice sparsity bound for h. Unfortunately, we do not recover a sparsity bound for g from this. The main reason is that this transformation is not exactly reversible,

as there is no 1-1 correspondence between monomials of f and $\hat{f}$; there is an s^d-sparsity blow-up in $\hat{f}$. Intuitively, there is a 'loss of information' in this transformation. However, we still make use of the 'available information' to get a recursive algorithm for determining whether f is an exact power in Algorithm 3.

4.1 Preliminaries

Notations: Let $f \in \mathbb{F}[x_1, \ldots, x_n]$. For some $i \in [n]$, let x_i be a variable in the support of f. For the sake of convenience, we slightly abuse the notation $\mathbf{x}$ to denote the set $\{x_1, \ldots, x_n\} \setminus \{x_i\}$ throughout this section. Let $f = \sum_{j=0}^{d} f_j \cdot x_i^j$ such that $\forall j$, $f_j \in \mathbb{F}[\mathbf{x}]$ and $f_d \not\equiv 0$. The leading coefficient of f w.r.t x_i is defined as $\mathrm{lc}_{x_i}(f) \stackrel{\Delta}{=} f_d$. Polynomial f is called monic w.r.t variable x_i, if $\mathrm{lc}_{x_i}(f) = 1$. We say that f is x_i-reverse-monic if $f|_{x_i=0} = f_0 = 1$. We say that f is reverse-monic if there exists some variable $x_i \in \mathrm{supp}(f)$ such that f is x_i-reverse monic.

Definition 4.1 (Chapter 6 [GG99])**.** *Let R be a unique factorization domain and K be its field of fractions. Let $g \in K[y]$ be a polynomial over K such that $g = \sum_{i=0}^{m} (g_i/b) \cdot y^i \in K[y]$, where $b \in R$ is the common denominator. The content of g is defined as $\mathrm{cont}(g) = \gcd(g_0, \ldots, g_m)/b$. We define primitive part of g as $\mathrm{pp}(g) = g/\mathrm{cont}(g)$. Observe that $\mathrm{cont}(g) \in K$, while $\mathrm{pp}(g) \in R[y]$.*

Examples: For $R = \mathbb{Z}$ and $K = \mathbb{Q}$, consider $g = 3y + 9/2 \in \mathbb{Q}[y]$. Then $g = (6y + 9)/2$ and $\mathrm{cont}(g) = \gcd(6, 9)/2 = 3/2 \in \mathbb{Q}$. And $\mathrm{pp}(g) = g/\mathrm{cont}(g) = 2y + 3 \in \mathbb{Z}[y]$. Let us now consider a bi-variate example. Let $R = \mathbb{F}[x]$ and $K = \mathbb{F}(x)$. Consider $g = (x^2 - 1) \cdot y + (x - 1)/(x + 1) \in \mathbb{F}(x)[y]$. Then, $\mathrm{cont}(g) = \gcd((x^2 - 1)(x + 1), (x - 1))/(x + 1) = (x - 1)/(x + 1) \in \mathbb{F}(x)$. And $pp(g) = (x + 1)^2 \cdot y + 1 \in \mathbb{F}[x][y]$.

The following lemma will be useful to us later on.

Lemma 4.2. *Let R be a unique factorization domain and K be its field of fractions. Let $f \in R[y]$ and $g \in K[y]$ such that $f = g^e$. Then, $g \in R[y]$.*

Proof. By definitions of content and primitive parts in Definition 4.1, we know that $\mathrm{cont}(g) \in K$, while $\mathrm{pp}(g) \in R[y]$ for $g = \mathrm{cont}(g) \cdot \mathrm{pp}(g)$.

Gauss's Lemma states that the product of two primitive polynomials is also primitive. From this, one can derive that for two polynomials $g, h \in K[y]$, $\mathrm{cont}(\mathrm{gh}) = \mathrm{cont}(g) \cdot \mathrm{cont}(h)$ and $\mathrm{pp}(gh) = \mathrm{pp}(g) \cdot \mathrm{pp}(h)$. In particular, $\mathrm{cont}(g^e) = \mathrm{cont}(g)^e$ and $\mathrm{pp}(g^e) = \mathrm{pp}(g)^e$. Since $f = g^e$, we get that

$$\mathrm{cont}(f) = \mathrm{cont}(g)^e. \tag{4.3}$$

Since $f \in R[y]$, we know that $\mathrm{cont}(f) \in R$ by definition. We will now use this to prove that $\mathrm{cont}(g) \in R$ also. This will suffice to prove that $g = \mathrm{cont}(g) \cdot \mathrm{pp}(g) \in R[y]$. Note that we can write $\mathrm{cont}(g)$ in the simplest form as $\mathrm{cont}(g) = \frac{a}{b}$, where $a, b \in R$ and $\gcd(a, b) = 1$. Let

$d \triangleq \text{cont}(f) \in R$. Using (4.3), we get that $d = \left(\frac{a}{b}\right)^e = \frac{a^e}{b^e}$. Now, let p be an irreducible factor of b in R. Since $d \in R$, p must divide the numerator a^e. If p divides a^e, then p must divide a also. This contradicts with the fact that $\gcd(a, b) = 1$. This means, that the denominator b must be one and hence, $\text{cont}(g) \in R$. Thus, $g \in R[y]$. □

4.2 Exact Power Testing

We start with the case when our input polynomial f is reverse-monic w.r.t. some variable. We generalize it to the general case in the section after.

4.2.1 Reverse Monic Case

We use Newton's Binomial Theorem to get the sparsity bound for $f^{1/e}$ below. This tool has been used before to bound the size of e-th roots for the classes of algebraic circuits, formulas and ABPs. See for example [Dut18, ST21]. Although, these works assume $\text{char}(\mathbb{F})$ to be 0 or a non-divisor of e, we prove our result below for fields of arbitrary characteristic.

Lemma 4.4. *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be an s-sparse polynomial of individual degree d which is x_i-reverse monic, for some $i \in [n]$. If $f = g^e$ for some polynomial $g \in \mathbb{F}[x_1, \ldots, x_n]$ and $e \in \mathbb{N}$, then g is $s^{d/e+1}$-sparse.*

Proof. We can write g as $g = f^{1/e} = (1 + (f-1))^{1/e}$. By Newton's Binomial Theorem, this gives

$$g = (1 + (f-1))^{1/e} = \sum_{i=0}^{\infty} \binom{1/e}{i} (f-1)^i. \tag{4.5}$$

We first focus on the case when $\text{char}(\mathbb{F})$ is either zero or it does not divide e. In that case $1/e$ is well defined in $\mathbb{F}$ and so are all the binomial coefficients appearing in (4.5). Since f is reverse-monic in x_i, $f|_{x_i=0} = 1$. This means that $(f-1)$ has x_i-degree ≥ 1. Since g has x_i-degree $= d/e$, (4.5) becomes a finite sum modulo the ideal $\langle x_i \rangle^{d/e+1}$. Thus,

$$g = \sum_{i=0}^{d/e} \binom{1/e}{i} (f-1)^i \mod \langle x_i \rangle^{d/e+1}. \tag{4.6}$$

Since $\|f - 1\| \leq s$, it is easy to see that $G = \sum_{i=0}^{d/e} \binom{1/e}{i} (f-1)^i$ is $s^{d/e+1}$-sparse. Since $g = G \bmod \langle x_i \rangle^{d/e+1}$ and going mod $\langle x_i \rangle^{d/e+1}$ can only decrease sparsity, we get that g is also $s^{d/e+1}$-sparse.

Now we handle the case when $\text{char}(\mathbb{F})$ divides e. Let p be the characteristic of $\mathbb{F}$, for some prime p. Let $e = p^k \cdot q$, where p^k is the highest power of p which divides e, for some integer $k \geq 1$ and $p \nmid q$. Then by the famous *Frobenius endomorphism*, we know that:

$$g(x_1, \ldots, x_n)^p = g(x_1^p, \ldots, x_n^p)$$

$$g(x_1, \ldots, x_n)^{p^k \cdot q} = \left(g(x_1^{p^k}, \ldots, x_n^{p^k})\right)^q. \tag{4.7}$$

Since $f = g^e = g^{p^k \cdot q}$, we can use the variable transformation $y_j \leftarrow x_j^{p^k}$, for each $j \in [n]$ to get that

$$f = g(y_1, \ldots, y_n)^q.$$

Observe that x_i-degree in every non-zero monomial of f is a multiple of $e = p^k \cdot q$, therefore f is a proper polynomial in $\mathbb{F}[y_1, \ldots, y_n]$. Moreover f is still s-sparse as the transformation does not affect sparsity. We further note that if f was reverse-monic in x_i, it will also be reverse-monic in y_i. Thus, we have reduced to the previous case, since p does not divide q. Moreover, the individual degree of f is now reduced, specifically y_i-degree of f is $d' = d/p^k$. This implies that $g(y_1, \ldots, y_n)$ has sparsity $\leq s^{d'/q+1} = s^{d/e+1}$. Since this transformation does not affect sparsity, we deduce that our original g is also $s^{d/e+1}$-sparse. □

Remark 4.8. Lemma 4.4 is also true for a monic f of sparsity s such that $f = g^e$. This is because we can convert a monic f into a reverse-monic $\hat{f}$ by the reversal transformation, $\hat{f} = \text{rev}_i^d[f]$ (see Definition 5.3). Observe that if f is monic in x_i, then $\hat{f}$ is reverse-monic in x_i. By definition, this transformation is invertible. In fact, $f = \text{rev}_i^d[\hat{f}]$, thus given $\hat{f}$, we can recover f. We also get that $\|f\| = \|\hat{f}\| = s$. Since $\hat{f} = \hat{g}^e$ and $\hat{g}$ is $s^{d/e+1}$-sparse using Lemma 4.4, we also get that g is $s^{d/e+1}$-sparse.

In fact, Lemma 4.4 gives rise to an algorithm to compute the e^{th} root of a reverse-monic f, as shown below.

Lemma 4.9. *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be an s-sparse polynomial of individual degree d which is x_i-reverse monic for some $i \in [n]$. If $f = g^e$ for some polynomial $g \in \mathbb{F}[x_1, \ldots, x_n]$ and $e \in \mathbb{N}$, then there is a deterministic algorithm to compute g in* $\text{poly}(s^{d/e}, n, d)$ $\mathbb{F}$*-operations.*

Proof. Algorithm 2 below computes the required g when $f = g^e$, for some reverse-monic f.

Correctness: Follows from Lemma 4.4.

Time Complexity: Steps 2 to 6, except Step 4 can be done in $O(d)$ time. Step 4 will take $\text{poly}(s, n, d)$ time as f is s-sparse. Steps 8 and 9, each take $\text{poly}(s^{d/e})$ $\mathbb{F}$-operations as $\|g\| \leq \|G\| \leq s^{d/e+1}$. Thus, total complexity is $\text{poly}(s^{d/e}, n, d)$ $\mathbb{F}$-operations. □

Remark. Using Remark 4.8, Lemma 4.9 also works for a monic f by first making it reverse-monic, computing its e^{th} root and then returning the reversal of that.

Algorithm 2: To compute e^{th} root of f:

Input: Polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ of individual degree $\leq d$, s-sparse and reverse-monic in variable x_i such that $f = g^e$.

Output: Root g.

Let $p \stackrel{\Delta}{=} \text{char}(\mathbb{F})$.
if $p > 0$ *and* $p \mid e$ **then**
  Let $e = p^k \cdot q$, where p^k is the highest power of p that divides e and $p \nmid q$.
  $f \leftarrow f\left(x_1^{1/p^k}, \ldots, x_n^{1/p^k}\right)$.
  $d \leftarrow d/p^k$.
  $e' \leftarrow e$ and $e \leftarrow e/p^k$. `/* Saving value of` e `in` e' `and updating` e `to` q `*/`
end
$G \longleftarrow \sum_{i=0}^{d/e} \binom{1/e}{i} (f-1)^i$.
$g \longleftarrow G \pmod{x_i^{d/e+1}}$.
if $p > 0$ *and* $p \mid e'$ **then**
  $g \leftarrow g\left(x_1^{p^k}, \ldots, x_n^{p^k}\right)$.
end
return g.

4.2.2 General Case

Now, we handle the case where input f is not monic or reverse-monic in any variable. In this case, we are not able to compute the exact root, but we can solve the decision version of this problem, that is we show how to efficiently test if $f = g^e$, for some g and $e \geq 1$.

We first give a standard trick to convert a polynomial into a reverse-monic polynomial. The properties mentioned below are fairly straightforward to prove, see for example [BSV20, Lem 5.5].

Definition 4.10 (Reverse-monic transformation)**.** *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be an s-sparse polynomial of individual degree at most d. Pick any variable $x_i \in \text{supp}(f)$ such that $f|_{x_i=0} \not\equiv 0$. Set $x_i = y$ and let $f_0 \stackrel{\Delta}{=} f|_{y=0}$. We define*

$$\hat{f} = \frac{1}{f_0} \cdot f(\mathbf{x}, f_0 \cdot y).$$

This transformation has some nice properties:

1. *$\hat{f}$ is reverse-monic in y. Moreover $\hat{f}$ is a proper polynomial in $\mathbb{F}[x_1, \ldots, x_n]$ (not a rational function).*
2. *$\|\hat{f}\| \leq s^d$.*
3. *Individual degree of $\hat{f}$ is at most d^2. However, $\deg_y(\hat{f}) = \deg_{x_i}(f) \leq d$.*

We remark that it could be the case that the trailing coefficient $f_0 = 0$ for every x_i above. We show how to handle that case in Step 3 of Algorithm 3. So without loss of generality, we can always convert our polynomial f into reverse-monic $\hat{f}$.

By definition of this transformation, one can easily show that if $f = g^e$, then $\hat{f} = h^e$, for some suitable h. However, the converse is not always true. For example, consider $f(x, z) = z(x+1)^2$. It is not an exact power but if we make it reverse-monic w.r.t. x we get $\hat{f}(y, z) = 1/z \cdot f(zy, z)$. It turns out to be $\hat{f} = (zy+1)^2$, which is an exact power. For testing whether f is an exact power, we need a converse also. In the two claims below, we find the extra condition on trailing coefficient, which gives us a suitable converse. This will amount to a recursive algorithm for exact power testing in Algorithm 3.

Claim 4.11 ($\Rightarrow$)**.** *If $f = g^e$ in $\mathbb{F}[\mathbf{x}, x_i]$, then $\hat{f} = h^e$ in $\mathbb{F}[\mathbf{x}, y]$ for some polynomial h and $f_0 = g_0^e$ in $\mathbb{F}[\mathbf{x}]$, where $g_0 \triangleq g|_{x_i=0}$.*

Proof. Let $f = f_k \cdot x_i^k + \ldots + f_1 \cdot x_i + f_0$ and $g = g_m \cdot x_i^m + \ldots + g_1 \cdot x_i + g_0$. If $f = g^e$, then $k = em$ and $f_0 = g_0^e$. Thus,

$$\begin{aligned}\hat{f} &= \frac{1}{f_0} \cdot f(\mathbf{x}, f_0 \cdot y) \\ &= \frac{1}{f_0} \cdot g(\mathbf{x}, f_0 \cdot y)^e \\ &= \left(\frac{g(\mathbf{x}, f_0 \cdot y)}{g_0}\right)^e = h^e,\end{aligned}$$

for $h \triangleq \frac{g(\mathbf{x}, f_0 \cdot y)}{g_0}$. By definition of the reverse-monic transformation, $\hat{f} \in \mathbb{F}[\mathbf{x}, y]$ is a proper polynomial in this ring. Clearly, h is in $\mathbb{F}(\mathbf{x})[y]$ by definition. Also $\hat{f} = h^e \in \mathbb{F}[\mathbf{x}, y]$, therefore h also belongs to $\mathbb{F}[\mathbf{x}, y]$, by Lemma 4.2. □

Claim 4.12 ($\Leftarrow$)**.** *If $\hat{f} = h^e$ in $\mathbb{F}[\mathbf{x}, y]$ and $f_0 = b^e$ in $\mathbb{F}[\mathbf{x}]$, then $f = g^e$ in $\mathbb{F}[\mathbf{x}, x_i]$.*

Proof. Observe that,

$$\begin{aligned}f(\mathbf{x}, x_i) &= f_0 \cdot \hat{f}\left(\mathbf{x}, \frac{x_i}{f_0}\right) \\ &= f_0 \cdot h\left(\mathbf{x}, \frac{x_i}{f_0}\right)^e \\ &= \left(b \cdot h\left(\mathbf{x}, \frac{x_i}{f_0}\right)\right)^e \\ &= \big(g(\mathbf{x}, x_i)\big)^e,\end{aligned}$$

for $g \triangleq b \cdot h(\mathbf{x}, \frac{x_i}{f_0})$. Clearly, $g \in \mathbb{F}(\mathbf{x})[x_i]$ from above. But since $f \in \mathbb{F}[\mathbf{x}][x_i]$ and $f = g^e$, this implies $g \in \mathbb{F}[\mathbf{x}, x_i]$ by Lemma 4.2. □

Let $r_{\mathbb{F}}(a, e)$ denote the time complexity of deciding whether $a = b^e$, for $a, b \in \mathbb{F}$ and for some $e \in \mathbb{N}$. Then,

- For a finite field $\mathbb{F} = \mathbb{F}_q$, $r_{\mathbb{F}} = \text{poly}(\log q)$ $\mathbb{F}$-opertions (Lemma 4.14).
- For the field of rationals $\mathbb{F} = \mathbb{Q}$, $r_{\mathbb{F}} = \text{poly}(e, \log a)$ $\mathbb{F}$-operations. For an integer (or rational number), it is easy to even compute the e-th root by binary search, or one can simply invoke univariate factorization ([LLL82]) for $x^e - a$ to compute $a^{1/e}$.

Theorem 3 follows from the next lemma.

Lemma 4.13. *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be an s-sparse polynomial of individual degree d. There is a deterministic algorithm to test whether $f = g^e$ for some polynomial $g \in \mathbb{F}[x_1, \ldots, x_n]$ and $e \in \mathbb{N}$. The algorithm takes $\text{poly}(s^{d^2}, n, d) + r_{\mathbb{F}}(f(0, \ldots, 0), e)$ $\mathbb{F}$-operations.*

Proof. The $e = 1$ case is trivial. Run the Algorithm 3 below for each $e \in \{2, \ldots, d\}$. If any such e exists such that $f = g^e$, that is Algorithm 3 outputs YES, then f is an exact power. Otherwise, if for every e Algorithm 3 outputs NO, then f is not an exact power.

Algorithm 3: Exact power testing

Input: An s-sparse polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ with individual degree d and an integer $e \in \{2, \ldots, d\}$.

Output: YES, if $f = g^e$ for some polynomial g and NO, otherwise.

1 For each $i \in [n]$, check whether f is reverse-monic in variable x_i. If such an i exists, then set $\hat{f} \stackrel{\Delta}{=} f, y \stackrel{\Delta}{=} x_i$ and go to Step 8 directly, else go to Step 2.

2 Choose any $i \in [n]$. Set $f_0 \stackrel{\Delta}{=} f|_{x_i=0}$ and $x_i \stackrel{\Delta}{=} y$.

3 **if** $f_0 = 0$ **then**

4 Let k be the highest power of x_i such that x_i^k divides f.

5 If $e \nmid k$ then output NO and return, otherwise set $f = f/x_i^k$ and $f_0 = f|_{x_i=0}$.

6 **end**

7 Define $\hat{f} \stackrel{\Delta}{=} \frac{1}{f_0} \cdot f(\mathbf{x}, f_0 \cdot y)$.

8 Invoke Algorithm 2 for $\hat{f}$ which is reverse-monic in variable y to get candidate root $\hat{g}$.

9 Check whether $\hat{f} = \hat{g}^e$, by multiplying out. If it is, go to Step 10, otherwise output NO.

10 For the $(n-1)$-variate polynomial $f_0 \stackrel{\Delta}{=} f|_{x_i=0} \in \mathbb{F}[\mathbf{x}]$, recursively check whether f_0 is e^{th} power of some polynomial. If it is, then output YES, otherwise output NO.

We discuss the correctness and time-complexity of Algorithm 3 below.

Correctness: If f is indeed equal to g^e for some $e \in \{2, \ldots, d\}$, then by Claim 4.11, $\hat{f} = h^e$ for some h and $f_0 = b^e$, for $b = g|_{x_i=0}$. Thus, by Lemma 4.9, Step 8 will compute the correct root $\hat{g}$ and

in Step 10, the algorithm will output YES. If f is not an exact power for any $e \in [d]$, the algorithm will output NO in either Step 5 or Step 9 or Step 10. This follows due to Claim 4.12 (consider contrapositive).

Time Complexity: Step 1 takes $\text{poly}(s, n, d)$ $\mathbb{F}$-operations as we only have to check whether $f|_{x_i=0} = 1$ at most n times. In Steps 2-6, we are required to compute the trailing coefficient of f w.r.t x_i-variable, which takes $\text{poly}(s, n, d)$ $\mathbb{F}$-operations by Lemma 2.2. Step 7 takes at most $\text{poly}(s^d, n, d)$ time. Step 8 takes at most $\text{poly}(\|\hat{f}\|^{d/e}, n, d)$ $\mathbb{F}$-operations by Lemma 4.9 as y-degree of $\hat{f}$ is still d. Since $\|\hat{f}\| \leq s^d$, this step takes at most $\text{poly}(s^{d^2/e}, n, d)$ $\mathbb{F}$-operations. Multiplying out in Step 9 will take take at most $\text{poly}(s^{(d^2/e)\cdot e}) = \text{poly}(s^{d^2})$ $\mathbb{F}$-operations as $\|\hat{g}\| \leq s^{d^2/e+1}$. Note that in Step 10, we recurse on f_0, which has sparsity $\leq s$, therefore there is no blow-up of sparsity in the recursion. Hence, this step takes $\text{poly}(s^{d^2}, n, d)$ $\mathbb{F}$-operations to reach the base case of deciding whether the field element $f(0, \ldots, 0)$ has an e-th root. The total complexity is thus, $\text{poly}(s^{d^2}, n, d) + r_{\mathbb{F}}(f(0, \ldots, 0), e)$ $\mathbb{F}$-operations. □

Using the standard theory of finite fields, we show how to test whether a finite field element is an exact power. In other words, we show that $r_{\mathbb{F}} = \text{poly}(\log q)$ for a finite field F_q. This is required for the base case of Algorithm 3, when working over finite fields.

Lemma 4.14 (Folklore). *For a finite field $\mathbb{F}_q$, we can decide whether an element $a \in \mathbb{F}_q$ is a k^{th} power residue, i.e. $a = b^k$ for some $b \in \mathbb{F}_q$ in $\text{poly}(\log q)$ $\mathbb{F}_q$-operations.*

Proof. We will focus on $\mathbb{F}_q^*$ since $0 = 0^k$ trivially. We will prove that $a = b^k$ for some $b \in \mathbb{F}_q^*$ and $k \geq 1$, if and only if $a^{\frac{q-1}{d}} = 1$ in $\mathbb{F}_q^*$, where $d = \gcd(k, q-1)$. Having proved that, we can simply test this by computing $a^{\frac{q-1}{d}}$ in $\text{poly}(\log q)$ $\mathbb{F}_q$-operations using repeated squaring. Now we prove both the directions for

$$a = b^k \Leftrightarrow a^{\frac{q-1}{d}} = 1.$$

($\Rightarrow$) Observe that $a = b^k \Rightarrow a^{\frac{q-1}{d}} = b^{\frac{k(q-1)}{d}}$. Since $d = \gcd(k, q-1)$, d divides k, hence $\frac{k}{d}$ is an integer. Thus, we get that $a^{\frac{q-1}{d}} = b^{\frac{k(q-1)}{d}} = 1$ by using the generalization of Fermat's Little Theorem, i.e. $x^{q-1} = 1$ for all $x \in \mathbb{F}_q^*$.

($\Leftarrow$) We know that $\mathbb{F}_q^*$ is a cyclic group of order $q-1$. Let g be its generator such that $a = g^r$, for some $r \in [q-2]$ (For $r = 0$, we know that $1 = 1^k$ trivially). Now, if $a^{\frac{q-1}{d}} = 1$, we get that $g^{\frac{r(q-1)}{d}} = 1$. This implies that $\frac{r}{d}$ is an integer or that d divides r. By Bezout's identity, we know that $d = \gcd(\mathrm{k}, \mathrm{q}-1) = sk + t(q-1)$ for some integers s, t. Since $d \mid r$, we get that $r = s'k + t'(q-1)$. This proves that a is a k^{th} power residue as:

$$a = g^r = g^{s'k+t'(q-1)} = g^{s'k} = b^k$$

in $\mathbb{F}_q^*$ for $b = g^{s'}$. □

5 Co-Factor Polynomial Sparsity

In this section we prove Theorem 4 as well as Corollary 1.4. In fact, we prove somewhat technically stronger versions of these theorems with more explicit parameters (not just in terms of big-Oh). We begin with some technical definitions. Some of these have been used implicitly in the previous sections.

5.1 Multivariate Reversal operation

Definition 5.1 (Reverse Monic, Reverse Pseudo-Monic)**.** *We say that a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ is* reverse monic *if there exists a variable $x_i \in \mathrm{supp}(f)$ such that $f|_{x_i=0} = 1$. If we know the variable beforehand, we say that f is x_i-*reverse monic*. We can extend this definition to a set of variables x_I, for some arbitrary $I \subseteq [n]$. We say that f is I-*reverse monic*, if $f|_{x_I=0_I} = 1$. We say that f is* reverse pseudo-monic*, x_i-*reverse pseudo-monic*, I-*reverse pseudo-monic *respectively, when instead of* 1 *the result of setting variables to* 0 *is a non-zero field element or a single monomial.*

In other words, the constant term of a reverse monic polynomial is 1, when regarded as a polynomial in the remaining variables. The following are immediate connections between some of the previously defined concepts.

Observation 5.2. *Let $h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial, $i \in [n]$ and $I \subseteq [n]$. Then:*

- $\mathrm{supp}(h|_{x_i=0}) = \{\mathbf{e} \in \mathrm{supp}(h) \mid e_i = 0\}$.
- *h is I-reverse pseudo-monic if and only if $|\mathrm{supp}(h|_{x_I=0_I})| = 1$.*

The following transformation will be useful later on to convert specific polynomials into I-reverse pseudo-monic polynomials.

Definition 5.3 (Reversal Transformation)**.** *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial and let $\ell \in \mathbb{N}$. We define the* reversal operation *on f with respect to a variable x_i as follows:*

$$\mathrm{rev}_i^{\ell}[f] \stackrel{\Delta}{=} x_i^{\ell} \cdot f|_{x_i=\frac{1}{x_i}} = x_i^{\ell} \cdot f(x_1, \ldots, x_{i-1}, \frac{1}{x_i}, x_{i+1}, \ldots, x_n).$$

By iteration, we can extend this definition to a set of variables x_I, for some arbitrary $I = \{i_1, \ldots, i_r\} \subseteq [n]$.

$$\mathrm{rev}_I^{\ell}[f] \stackrel{\Delta}{=} \mathrm{rev}_{i_1}^{\ell}\left[\mathrm{rev}_{i_2}^{\ell}[\ldots \mathrm{rev}_{i_r}^{\ell}[f]]\right].$$

Alternatively:

$$\mathrm{rev}_I^{\ell}[f] \stackrel{\Delta}{=} x_{i_1}^{\ell} \cdots x_{i_r}^{\ell} \cdot f(y_1, \ldots, y_n),$$
$$\text{where } y_j = \begin{cases} \frac{1}{x_j}, & \text{if } x_j \in I \\ x_j, & \text{if } x_j \notin I. \end{cases}$$

For intuition, express f as a polynomial in x_i such that $f = f_d x_i^d + f_{d-1} x_i^{d-1} + \ldots + f_1 x_i + f_0$, where each coefficient f_j is a polynomial in variables other than x_i. Then, $\mathrm{rev}_i^d[f]$ reverses the order of coefficients in this representation. That is, $\mathrm{rev}_i^d[f] = f_0 x_i^d + f_1 x_i^{d-1} + \ldots + f_{d-1} x_i + f_d$. In particular, if f is monic in x_i then $\mathrm{rev}_i^d[f]$ is x_i-reverse monic.

The following lemma summarizes some of the basic, yet useful properties of the reversal transformation. Subsequently, we will use these properties implicitly.

Lemma 5.4. *Let $f, g, h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ such that $f = g \cdot h$. Let $i \in [n]$ and suppose that $d \geq \deg_{x_i}(f)$. Then:*

1. $\mathrm{rev}_i^d[f]$ *is a polynomial (and not a rational function).*
2. $\deg_{x_i}(\mathrm{rev}_i^d[f]) \leq d$.
3. $\|\mathrm{rev}_i^d[f]\| = \|f\|$.
4. *Let a, b such that $d = a + b$. Then* $\mathrm{rev}_i^d[f] = \mathrm{rev}_i^a[g] \cdot \mathrm{rev}_i^b[h]$.

5.2 Unique Projections

Definition 5.5. *Let $V \subseteq \mathbb{N}^n$. A* unique projection *of V of* length k *is a set $\{(i_1, e_1), (i_2, e_2), \ldots, (i_k, e_k)\}$ such that there exists a unique vector $\mathbf{v} \in V$ satisfying $\forall j \in [k] : v_{i_j} = e_j$.*
A unique projection of a polynomial $h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ is defined as a unique projection of $\mathrm{supp}(h)$.

In other words, there exists a unique monomial in the monomial representation of h that contains the pattern $x_{i_1}^{e_1} x_{i_2}^{e_2} \cdots x_{i_k}^{e_k}$. The following is immediate from the definition.

Observation 5.6. *Let $h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial and let $\{(i_1, e_1), (i_2, e_2), \ldots, (i_k, e_k)\}$ be a unique projection of h. Pick $j \in [k]$ and let $\ell \geq e_j$. Then*

$$\{(i_1, e_1), (i_2, e_2), \ldots, (i_{j-1}, e_{j-1}), (i_j, \ell - e_j), (i_{j+1}, e_{j+1}), \ldots, (i_k, e_k)\}$$

is a unique projection of $\mathrm{rev}_{i_j}^{\ell}[h]$.

Subsequently, we demonstrate the usefulness of unique projections.

Lemma 5.7. *Let $h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial with a unique projection of the form $\{(i_1, 0), (i_2, 0), \ldots, (i_k, 0)\}$ (i.e. $\forall j \in [k] : e_k = 0$). Then h is $\{i_1, i_2, \ldots, i_k\}$-reverse pseudo-monic.*

Proof. Let $I = \{i_1, i_2, \ldots, i_k\}$. By iterative application of Part 1 of Observation 5.2, we obtain that

$$\mathrm{supp}(h|_{x_I = 0_I}) = \left\{ \mathbf{e} \in \mathrm{supp}(h) \;\middle|\; \forall j \in [k] : e_{i_j} = 0 \right\}$$

As I corresponds to a unique projection, the set of the RHS contains exactly one vector and the claim follows from Part 2 of Observation 5.2. □

Lemma 5.8. *Let $h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a multilinear polynomial and let $\{(i_1, e_1), (i_2, e_2), \ldots, (i_k, e_k)\}$ be a unique projection of h. Furthermore, let $J = \{i_j \mid e_j = 1\}$. That is, the set of all indices i_j for which $e_j = 1$. Then $\tilde{h} \triangleq \mathrm{rev}_J^1[h]$ is $\{i_1, i_2, \ldots, i_k\}$-reverse pseudo-monic.*

Proof. First, note that since h is a multilinear polynomial, we have that $e_j = 0$ for indices $j \in \{i_1, \ldots, i_k\} \setminus J$. Subsequently, by iterative application of Observation 5.6, we obtain that $\tilde{h}$ is a multilinear polynomial with a unique projection $\{(i_1, 0), (i_2, 0), \ldots, (i_k, 0)\}$. Note that $\tilde{h}$ is a proper polynomial (and not a rational function) by iterative application of Part 1 in Lemma 5.4. The claim then follows from Lemma 5.7. □

We conclude this section by showing that every set contains a unique projection of (at most) logarithmic size and a relation of unique projections with δ-entropy polynomials that were defined in [BS21].

Lemma 5.9. *Let $V \subseteq \mathbb{N}^n$ of size $|V| \leq s$. Then V has a unique projection of length at most $\log s + 1$.*

Proof. The proof is by induction on the size of V. For the base case $|V| = 1$ there exists a unique projection of length 1. Now assume $|V| \geq 2$. Therefore, V contains at least two different vectors $\mathbf{u} \neq \mathbf{w}$. Let i be such that $u_i \neq w_i$. Let us denote $a = u_i$ and $b = w_i$. Partition V into $V_a \triangleq \{\mathbf{v} \in V \mid v_i = a\}$ and $V_b \triangleq \{\mathbf{v} \in V \mid v_i = b\}$. We have that $|V_a| + |V_b| \leq |V|$. Hence, wlog $1 \leq |V_a| \leq s/2$. By the induction hypothesis, V_a has a unique projection of length at most $\log(s/2) + 1 = \log s$. We now add the index i and $e_i = a$ to the set to obtain a unique projection for V of size $\log s + 1$. □

Recently, [BS21] defined a class of polynomials called 'low-entropy' polynomials and showed an $(nd)^{O(d\delta)}$ sparsity upper bound for the factors of a δ-entropy polynomial. We quickly give their definition of a δ-entropy set and then show a combinatorial connection of entropy with our notion of unique projections below. We note this connection between these two combinatorial concepts but our results are incomparable from those in [BS21].

Definition 5.10 ([BS21])**.** *A vector $\mathbf{v} \in \mathbb{N}^n$ has entropy δ if it has the same value in $(n - \delta)$ of its coordinates. A set $V \subseteq \mathbb{N}^n$ is called a δ-entropy set if for every $\mathbf{v} \in V$, $\mathbf{v}$ has entropy $\leq \delta$.*

Lemma 5.11. *Let $V \subseteq \mathbb{N}^n$ be a δ-entropy set. Then V has a unique projection of length at most $2\delta + 1$.*

Proof. Let $\mathbf{u} \in V$ be a vector with maximum entropy in V. Let $m(\mathbf{u})$ denote the majority value in $\mathbf{u}$. In other words, $\mathbf{u}$ has the largest number of non-majority values among all vectors in V. Let $u_{i_1}, \ldots, u_{i_k}$ be all the non-majority values in $\mathbf{u}$. Since $\mathbf{u}$ has entropy $\leq \delta$, the length of this sequence k is at most δ. We note that the remaining elements of $\mathbf{u}$ outside this sequence have the same value (equal to $m(\mathbf{u})$). We extend the sequence to length $k + \delta + 1$ by adding any $\delta + 1$ elements from these remaining elements of $\mathbf{u}$ to get: $u_{i_1}, \ldots, u_{i_k}, u_{i_{k+1}}, \ldots, u_{i_{k+\delta+1}}$. We claim that $\{(i_1, u_{i_1}), \ldots, (i_{k+\delta+1}, u_{k+\delta+1})\}$ is a unique projection of the set V.

We need to show that if $\mathbf{v}$ is another vector in V such that $v_{i_j} = u_{i_j}$, for each $j \in [k+\delta+1]$ (values agree on projection), then $\mathbf{v} = \mathbf{u}$. Observe that $u_{i_{k+1}} = \ldots = u_{i_{k+\delta+1}} = m(\mathbf{u})$, by definition of k. This means $v_{i_{k+1}} = \ldots = v_{i_{k+\delta+1}} = m(\mathbf{u})$ also. We deduce that at least $\delta+1$ coordinates of $\mathbf{v}$ have value $m(\mathbf{u})$. We also know that $\mathbf{v}$ has entropy $\leq \delta$ and note that for a $\leq \delta$-entropy vector, if any $\delta+1$ coordinates have the same value, that value is the majority value. Hence, $m(\mathbf{v}) = m(\mathbf{u})$. Now suppose for the sake of contradiction that there exists some coordinate i_r outside the projection ($r > k+\delta+1$) for which $v_{i_r} \neq u_{i_r}$. Since all the non-majority values of $\mathbf{u}$ have already appeared in the projection coordinates, we deduce that $u_{i_r} = m(\mathbf{u}) = m(\mathbf{v})$. This means that $v_{i_r} \neq m(\mathbf{v})$. In that case, v_{i_r} is another element in $\mathbf{v}$ apart from $v_{i_1}, \ldots, v_{i_k}$ which is distinct from $m(\mathbf{v})$. This is a contradiction to our assumption that $\mathbf{u}$ is a vector with maximum entropy. Hence, $v_{i_j} = u_{i_j}$ for all $j > k+\delta+1$. By our premise, they also agree on the $k+\delta+1$ projection coordinates. Hence $v_j = u_j$, for all $j \in [n]$ and thus, $\mathbf{v} = \mathbf{u}$. Moreover, since $k \leq \delta$, length of this unique projection is $k+\delta+1 \leq 2\delta+1$. □

5.3 Proofs of Theorem 4 and Corollary 1.4

We are now ready to state and prove the technical results of this section that will imply Theorem 4 and Corollary 1.4. In what follows, let $f, h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be two s-sparse polynomials such that $f = gh$.

Lemma 5.12. *Suppose that h is reverse monic and the individual degree of g is at most d. Then g is s^{d+2}-sparse.*

Proof. By hypothesis, let h be reverse monic w.r.t. some variable $x_i \in \text{supp}(h)$. Express h as a univariate in x_i with coefficients as polynomials in the remaining variables. Since h is x_i-reverse monic, the constant term, $h|_{x_i=0}$ is 1. Therefore, every term in $(1-h)$ has x_i-degree ≥ 1. We use this observation in a division-elimination argument as follows:

$$g = \frac{f}{h} = \frac{f}{1-(1-h)} = \sum_{j=0}^{\infty} f(1-h)^j. \tag{5.13}$$

Let x_i-degree of g be d_i. Then, we can safely truncate the infinite sum in Equation (5.13) as follows:

$$g = \sum_{j=0}^{d_i} f(1-h)^j \bmod \langle x_i^{d_i+1} \rangle. \tag{5.14}$$

Equation (5.14) helps us in bounding sparsity of g. Note that going mod $\langle x_i^{d_i+1} \rangle$ can only decrease sparsity, so we focus only on the sparsity of finite sum in (5.14). Since g is a factor of f, its individual degree d_i is also upper bounded by d. Also note that both $\|f\|, \|(1-h)\| \leq s$. Therefore, we get that $\|g\| \leq \sum_{j=0}^{d} s^{j+1} \leq s^{d+2}$. □

Generalizing this observation we obtain:

Lemma 5.15. *Suppose that h is I-reverse monic and the individual degrees of the variables of g in x_I are at most d. Then g is $s^{d|I|+2}$-sparse.*

Proof. We follow the same template as in proof of Lemma 5.12, with the change that h is reverse monic with respect to a set I of variables instead of just a single variable. Express h as a polynomial in x_I variables with coefficients as polynomials in the remaining $n - |I|$ variables. Since h is I-reverse monic, $h|_{x_I=0_I}$ (the constant term of h) is 1. Therefore, every term in $(1-h)$ has total x_I-degree ≥ 1. We then get the same Equation (5.13) for g. Let $I = \{i_1, \ldots, i_k\} \subseteq [n]$, where $k = |I|$. Let individual degree of variable x_{i_j} in g be d_j for each $j \in [k]$. Then, we can truncate the infinite sum as follows:

$$g = \sum_{j=0}^{dk} f(1-h)^j \bmod \langle x_{i_1}^{d_1+1}, \ldots, x_{i_k}^{d_k+1} \rangle. \tag{5.16}$$

By the premises, for each $j \in [k]$ each individual degree d_j is upper bounded by d. Therefore, we only need to sum up to $j = dk$ in (5.16) as the total degree in x_I variables is upper bounded by dk. Therefore, we get that $\|g\| \leq \sum_{j=0}^{dk} s^{j+1} \leq s^{dk+2} = s^{d|I|+2}$. □

The next lemma transforms a pseudo-monic polynomial into a monic polynomial while maintaining the sparsity and the multiplicative properties.

Lemma 5.17. *Let $f = gh$. Suppose that h is I-reverse pseudo-monic and the individual degrees of the variables of g in x_I are at most d. Then there exists polynomials $\tilde{f}, \tilde{g}, \tilde{h} \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ such that:*

1. *$\tilde{h}$ is I-reverse monic.*
2. *$\tilde{f} = \tilde{g}\tilde{h}$.*
3. *$\|\tilde{f}\| = \|f\|, \|\tilde{g}\| = \|g\|, \|\tilde{h}\| = \|h\|$.*
4. *The individual degrees of the variables of $\tilde{g}$ in x_I are at most d.*

Proof. Let $\alpha \triangleq h|_{x_I=0_I}$. We first define $\hat{f}, \hat{g}$ and $\hat{h}$ by setting $x_i \triangleq x_i \cdot \alpha$ for all $i \in I$, into f, g and h, respectively. Next, we set $\tilde{f} \triangleq \hat{f}$, $\tilde{g} \triangleq \hat{g} \cdot \alpha$ and $\tilde{h} = \hat{h}/\alpha$. We will now prove each part of the claim.

1. First, observe that $\tilde{h}$ is, indeed, a polynomial (and not a rational function). This is due to the fact that α divides $\hat{h}$. Next, $\tilde{h}|_{x_I=0_I} = \hat{h}|_{x_I=0_I}/\alpha = h|_{x_I=0_I}/\alpha = 1$.
2. $\tilde{f} = \hat{f} = \hat{g}\hat{h} = (\hat{g} \cdot \alpha)(\hat{h}/\alpha) = \tilde{g}\tilde{h}$.
3. Since α is a monomial or a field element there is $1-1$ correspondence between the monomials of f, g, h and $\tilde{f}, \tilde{g}, \tilde{h}$, respectively.
4. By definition, $\alpha \in \mathbb{F}[x_{[n]\setminus I}]$. Hence, multiplication or division by α does not affect the degrees of the variables in I. □

By transforming a pseudo-monic polynomial into a monic polynomial we can generalize Lemma 5.15 to the pseudo-monic case.

Corollary 5.18. *Suppose that h is I-reverse pseudo-monic and the individual degrees of the variables of g in x_I are at most d. Then g is $s^{d|I|+2}$-sparse.*

Proof. Apply Lemma 5.15 on $\tilde{f}, \tilde{g}$ and $\tilde{h}$ from Lemma 5.17. We obtain that $\tilde{g}$ and hence g is $s^{d|I|+2}$-sparse. □

Remark 5.19. In the context of exact-root sparsity, we can extend the result of Lemma 4.4 from the reverse monic to the I-reverse pseudo-monic case. It is done in the exact same fashion as we moved from Lemma 5.12 to Corollary 5.18 above. The formal statement is given in Theorem 5.20 below. In addition, we observe that if $f = g^e$ then f is I-reverse pseudo-monic iff g is I-reverse pseudo-monic (for the exact same I).

Theorem 5.20. *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial of sparsity s and individual degree at most d such that $f = g^e$ for some (other) polynomial $g \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ and $e \in \mathbb{N}$. In addition, suppose that f is I-reverse pseudo-monic for some $I \subseteq [n]$. Then the sparsity of g is bounded by $s^{O(d \cdot |I|/e)}$.*

Theorem 4 and Corollary 1.4 follow from the next two claims.

Theorem 5.21. *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be an s-sparse polynomial, with a multilinear factor h such that $f = g \cdot h$. Suppose that the individual degree of g is at most d and h has a unique projection of length at most k. Then g is s^{dk+2}-sparse.*

Proof. Let $\{(i_1, e_1), (i_2, e_2), \ldots, (i_k, e_k)\}$ be the guaranteed unique projection of h and let $J = \{i_j \mid e_j = 1\}$. We define:

$$\tilde{f} \triangleq \mathrm{rev}_J^{d+1}[f] \ , \ \tilde{g} \triangleq \mathrm{rev}_J^{d}[g] \text{ and } \tilde{h} \triangleq \mathrm{rev}_J^{1}[h].$$

By Lemma 5.4, we have that $\tilde{f} = \tilde{g} \cdot \tilde{h}$, where $\tilde{f}$ is an s-sparse polynomial, $\tilde{g}$ is a polynomial with individual degree at most d and $\tilde{h}$ is a multilinear polynomial. Furthermore, by Lemma 2.1, $\|\tilde{h}\| \leq \|\tilde{f}\| \leq s$. Finally, by Lemma 5.8, $\tilde{h}$ is $\{i_1, i_2, \ldots, i_k\}$-reverse pseudo-monic. Consequently, by Corollary 5.18, we obtain that $\tilde{g}$ and hence g are s^{dk+2}-sparse. □

Corollary 5.22. *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be an s-sparse polynomial, such that $f = g \cdot h$ where h is multilinear polynomial and g is a polynomial with individual degree at most d. Then g is $s^{d(\log s+1)+2}$-sparse.*

Proof. By Lemma 2.1, $\|h\| \leq \|f\| \leq s$. Consequently, by Lemma 5.9, h has a unique projection length at most $\log s + 1$. Further using Theorem 5.21, we deduce that $\|g\| \leq s^{d(\log s+1)+2}$. □

Similarly, by plugging in Lemma 5.11 into Theorem 5.21 we obtain the following relation to low-entropy polynomials.

Corollary 5.23. *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be an s-sparse polynomial, such that $f = g \cdot h$ where h is a δ-entropy, multilinear polynomial and g is a polynomial with individual degree at most d. Then g is $s^{d(2\delta+1)+2}$-sparse.*

Remark 5.24. By using the formal expansion:

$$\frac{1}{(1-x)^\ell} = \sum_{j=0}^{\infty} \binom{j+\ell-1}{j} x^j$$

for division elimination in the proof of Lemma 5.15, we can get somewhat stronger versions of Theorem 4 below.

Theorem 5.25. *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial of sparsity s and individual degree at most d such that $f = gh^\ell$ for some $\ell \in \mathbb{N}$. Suppose, in addition, that h is a multilinear polynomial with a unique projection of length k. Then the sparsity of g is bounded by $s^{O((d-\ell)k)}$.*

6 Future Directions

A lot of interesting open problems arise in the context of this work:

- Design a polynomial-time PIT algorithm for $\Sigma^{[k]}\Pi\Sigma\Pi^{[\deg_{x_i} \leq d]}$ circuits with bounded k and d, for $k \geq 3$. To the best of our knowledge, the smallest open case is $k = 3$ and $d = 1$!
- Prove a polynomial-size sparsity bound (Conjecture 1.3) even for the special cases of exact-roots and multilinear co-factors.
  - In particular, improve the sparsity bound in Corollary 1.4. Ideally, get rid of the $\log s$ term in the exponent. One can start by studying the structure of polynomials with non-constant or log-sized unique projections.
  - Likewise, generalize Theorem 5.20 to work for any general f with bounded individual degree d. The smallest open case here is $d = 4$ and $e = 2$, in other words prove that square-root is sparse.

Acknowledgments

The authors would like to thank the anonymous referees for their useful comments that improved the presentation of the results.

References

[AGS19] M. Agrawal, S. Ghosh, and N. Saxena. Bootstrapping variables in algebraic circuits. *Proceedings of the National Academy of Sciences*, 116(17):8107–8118, 2019. 6

[AV08] M. Agrawal and V. Vinay. Arithmetic circuits: A chasm at depth four. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 67–75, 2008. 6

[BOT88] M. Ben-Or and P. Tiwari. A Deterministic Algorithm for Sparse Multivariate Polynomial Interpolation. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 301–309, 1988. 2

[BS21] P. Bisht and N. Saxena. Derandomization via symmetric polytopes: Poly-time factorization of certain sparse polynomials. 2021. 6, 30

[BSV20] V. Bhargava, Sh. Saraf, and I. Volkovich. Deterministic Factorization of Sparse Polynomials with Bounded Individual Degree. *J. ACM*, 67(2):8:1–8:28, 2020. 2, 3, 4, 5, 6, 8, 10, 12, 18, 24

[CLO15] D. A. Cox, J. Little, and D. O'Shea. *Ideals, varieties, and algorithms - an introduction to computational algebraic geometry and commutative algebra (4. ed.)*. Undergraduate texts in mathematics. Springer, 2015. 4

[CR88] B. Chor and R. L. Rivest. A knapsack-type public key cryptosystem based on arithmetic in finite fields. *IEEE Transactions on Information Theory*, 34(5):901–909, 1988. 1

[DDS21] P. Dutta, P. Dwivedi, and N. Saxena. Deterministic identity testing paradigms for bounded top-fanin depth-4 circuits. In *36th Conference on Computational Complexity (CCC 2021)*, volume 5, page 9, 2021. 6, 13

[DL78] R. A. DeMillo and R. J. Lipton. A Probabilistic Remark on Algebraic Program Testing. *Inf. Process. Lett.*, 7(4):193–195, 1978. 3, 13

[Dut18] P. Dutta. Discovering the roots: Unifying and extending results on multivariate polynomial factoring in algebraic complexity. *Master's thesis, Chennai Mathematical Institute*, 2018. 22

[For15] M. A. Forbes. Deterministic divisibility testing via shifted partial derivatives. In *FOCS*, 2015. 7

[GCL92] K. O. Geddes, S. R. Czapor, and G. Labahn. *Algorithms for computer algebra*. Kluwer, 1992. 4, 15

[GG99] J. von zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, 1999. 4, 21

[GJR10] E. Grigorescu, K. Jung, and R. Rubinfeld. A local decision test for sparse polynomials. *Inf. Process. Lett.*, 110(20):898–901, 2010. 2

[GK85] J. von zur Gathen and E. Kaltofen. Factoring Sparse Multivariate Polynomials. *Journal of Computer and System Sciences*, 31(2):265–287, 1985. 2, 6

[GS99] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon codes and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999. 1

[Kal89] E. Kaltofen. Factorization of polynomials given by straight-line programs. In S. Micali, editor, *Randomness in Computation*, volume 5 of *Advances in Computing Research*, pages 375–412. JAI Press Inc., Greenwhich, Connecticut, 1989. 1

[KI04] V. Kabanets and R. Impagliazzo. Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds. *Computational Complexity*, 13(1-2):1–46, 2004. 1

[KS01] A. Klivans and D. Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 216–223, 2001. 5, 8, 18, 38

[KT90] E. Kaltofen and B. M. Trager. Computing with Polynomials Given by Black Boxes for Their Evaluations: Greatest Common Divisors, Factorization, Separation of Numerators and Denominators. *J. of Symbolic Computation*, 9(3):301–320, 1990. 1

[LLL82] A.K. Lenstra, H.W. Lenstr, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen,*, 261(4):515–534, 1982. 26

[LST22] N. Limaye, S. Srinivasan, and S. Tavenas. Superpolynomial lower bounds against low-depth algebraic circuits. In *FOCS 2021*, 2022. 6, 13

[Ore22] Ø. Ore. Über höhere kongruenzen. *Norsk Mat. Forenings Skrifter*, 1(7):15, 1922. 13

[PS21] S. Peleg and A. Shpilka. Polynomial time deterministic identity testing algorithm for Σ [3] $\Pi\Sigma\Pi$ [2] circuits via Edelstein–Kelly type theorem for quadratic polynomials. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 259–271, 2021. 6, 13

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980. 3, 13

[SSS13] C. Saha, R. Saptharishi, and N. Saxena. A Case of Depth-3 Identity Testing, Sparse Factorization and Duality. *Computational Complexity*, 22(1):39–69, 2013. 6

[ST21] A. Sinhababu and T. Thierauf. Factorization of polynomials given by arithmetic branching programs. *computational complexity*, 30(2):1–47, 2021. 22

[Str73] V. Strassen. Vermeidung von Divisionen. *J. of Reine Angew. Math.*, 264:182–202, 1973. 9

[Sud97] M. Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997. 1

[SV10] A. Shpilka and I. Volkovich. On the Relation between Polynomial Identity Testing and Finding Variable Disjoint Factors. In *Automata, Languages and Programming, 37th International Colloquium (ICALP)*, pages 408–419, 2010. Full version at https://eccc.weizmann.ac.il/report/2010/036. 2

[SV18] S. Saraf and I. Volkovich. Blackbox Identity Testing for Depth-4 Multilinear Circuits. *Combinatorica*, 38(5):1205–1238, 2018. 7

[SY10] A. Shpilka and A. Yehudayoff. Arithmetic Circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010. 13, 37

[Vol15] I. Volkovich. Deterministically Factoring Sparse Polynomials into Multilinear Factors and Sums of Univariate Polynomials. In *APPROX-RANDOM*, pages 943–958, 2015. 2

[Vol17] I. Volkovich. On some Computations on Sparse Polynomials. In *APPROX-RANDOM*, pages 48:1–4:21, 2017. 2, 3, 6, 7

[Zip79] R. Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, pages 216–226, 1979. 2, 3, 13

A Basics of algebraic complexity

In this section, we formally define various algebraic models of computation. We refer the reader to the excellent survey of [SY10] for a detailed discussion.

A.1 Algebraic computational models:

An *algebraic circuit* or an arithmetic circuit is a directed acyclic graph with input leaves at the bottom and a single output node at top, where the computation is done bottom-up. The leaves are labeled with variables or field constants while the internal nodes are either addition or multiplication gates. A directed edge between two nodes $u \to v$ is labeled with field constants, which gets multiplied to the polynomial computed at node u before feeding it to node v. The in-degree of a node is called its *fan-in* and out-degree is called *fan-out*. *Size* of the circuit is simply size of the directed graph, which is the maximum among number of edges and number of nodes. *Depth* of the circuit is length of the longest path from a leaf to the output node. *Degree* of the circuit is maximum degree of a polynomial computed at any node in the circuit.

A *depth-2* $\Sigma\Pi$ *circuit* of size s computes a sum of s-many monomials. Thus, depth-2 circuits compute the class of sparse polynomials. A size s, depth-4 $\Sigma\Pi\Sigma\Pi$ circuit computes a polynomial of the form $f = \sum_{i=1}^{k} \prod_{j=1}^{m} f_{ij}$, where f_{ij} are s-sparse polynomials for each $i \in [k], j \in [m]$. The much more general class of $\text{poly}(n)$-sized and $\text{poly}(n)$ degree algebraic circuits is called VP, which is considered the algebraic analog of complexity class P. The class VNP is considered the algebraic analog of complexity class NP. It is the class of polynomials which can be expressed as an exponential sum of a projection of a VP circuit family.

An algebraic circuit where fan-out of each node is one is called an *algebraic formula*. The class of polynomial sized formulas is called VF.

An *algebraic branching program (ABP)* is a layered directed graph with a unique source and sink vertex. Each edge is directed from one layer to the next with a linear polynomial associated to it as its weight. The weight of a path is the product of edge weights along the path. The ABP then computes the sum of all weighted paths from source to sink, as its output polynomial. The *length* of an ABP is the length of the longest path from source to sink and *width* of an ABP is the maximum number of vertices in any layer. The *size* of ABP is the product of its length and width. The class of all polynomial sized ABPs is called VBP.

A.2 PIT-Preliminaries

The problem of PIT asks for determining whether a given input polynomial is identically zero or not. The input polynomial is given in the form of some algebraic circuit. In *white-box* PIT, one can look 'inside' the input circuit while in *black-box* PIT, the input is given as a black-box and one can only evaluate the given circuit on field points. Therefore, in black-box PIT for a class of n-variate polynomials $\mathcal{C}$, we are asked to provide a set $\mathcal{H} \in \mathbb{F}^n$ such that for any non-zero $f \in \mathcal{C}$, there exists at least one point $\boldsymbol{\alpha} \in \mathcal{H}$ such that $f(\boldsymbol{\alpha}) \neq 0$. Such a set $\mathcal{H}$ is called *hitting-set* for class $\mathcal{C}$.

There is also a notion of hitting set generator (HSG) or simply generator in short, which is equivalent to a hitting set and is easier to work with PIT algorithms. We frame the PIT result in this work using generators. Definition 3.1 gives the formal definition of a generator.

Below, we mention the sparse PIT map of [KS01] which gives efficient deterministic black-box PIT for the class of sparse polynomials and few other folklore results.

Lemma A.1 (Sparse HSG; [KS01])**.** *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a non-zero polynomial of individual degree at most d, such that $\|f\| \leq m$. Let p be a prime larger than $\max(d, mn+1)$. Then, there exists a $k \in [mn+1]$ such that the univariate polynomial $f'(y) = f(y, y^{k^1 \bmod p}, \ldots, y^{k^{n-1} \bmod p})$ is non-zero. This yields an HSG $\mathcal{G} : \mathbb{F} \to \mathbb{F}^n$ of seed-length 1 for the class of m-sparse polynomials such that $\deg(\mathcal{G}) = \text{poly}(m, n, d)$.*

Lemma A.2 (Folklore)**.** *Let $f = \prod_{i=1}^{k} f_i$ be a non-zero polynomial, where for each $i \in [k]$, $f_i \in \mathcal{C}$, for some circuit class $\mathcal{C}$. Let $\mathcal{G}$ be a generator for $\mathcal{C}$. Then, $f \circ \mathcal{G} \neq 0$.*

Lemma A.3 (Folklore)**.** *Let $f \in \mathcal{C}$ be a non-zero polynomial in circuit class $\mathcal{C}$. Let $\mathcal{G}$ be a generator for $\mathcal{C}$. If f has a non-zero factor g, then $g \circ \mathcal{G} \neq 0$.*

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il