A primer on the closure of algebraic complexity classes under factoring

C.S. Bhargav and Prateek Dwivedi and Nitin Saxena

Abstract Polynomial factorization is a fundamental problem in computational algebra. Over the past half century, a variety of algorithmic techniques have been developed to tackle different variants of this problem. In parallel, algebraic complexity theory classifies polynomials into complexity classes based on their perceived 'hardness'. This raises a natural question: Do these classes afford efficient factorization?

In this survey, we revisit two pivotal techniques in polynomial factorization: Hensel lifting and Newton iteration. Though they are variants of the same theme, their distinct applications across the literature warrant separate treatment. These techniques have played an important role in resolving key factoring questions in algebraic complexity theory. We examine and organise the known results through the lens of these techniques. We also discuss their equivalence while reflecting on how their use varies with the context of the problem.

We focus on four prominent complexity classes: circuits of polynomial size ($\mathsf{VP}_{\mathrm{nb}}$), circuits with both polynomial size and degree (VP and its border $\overline{\mathsf{VP}}$), verifier circuits of polynomial size and degree (VNP), and polynomial-size algebraic branching programs (VBP). We also examine more restricted models, such as formulas and bounded-depth circuits. Along the way, we list several open problems that remain unresolved.

C.S. Bhargav
IIT Kanpur, India, e-mail: bhargav@cse.iitk.ac.in

Prateek Dwivedi
ITU Copenhagen, Denmark, e-mail: prdw@itu.dk

Nitin Saxena
IIT Kanpur, India, e-mail: nitin@cse.iitk.ac.in

Contents

1 Introduction

The problem of finding a nontrivial factor of a polynomial is a classical and fundamental one, with a rich history spanning centuries [Gat06]. Kaltofen [Kal82; Kal90; Kal92] gave a thorough treatment of several foundational results, while von zur Gathen and Panario [GP01] focused on factorization over finite fields. Comprehensive textbook treatments are contained in [GG13] and [Sho09].

In this survey, we focus on the core techniques and ideas that drive factorization results within algebraic circuit complexity. Our aim is to illustrate how these tools have contributed to resolving long-standing questions in the field. Notably, Forbes and Shpilka [FS15] have provided an accessible exposition of the high-level ideas behind some factoring algorithms, emphasising their relevance to other problems in algebraic complexity. Building on this, our survey aims to delve into the technical intricacies of these results by organising them according to the underlying methods. More than algorithms, we will be concerned with whether the factors of a 'structured' polynomial are also structured. We begin with a gentle introduction to each technique, followed by a discussion of the key results that it enables. The simplest case, perhaps, is when the polynomial is of a single variable.

1.1 Univariate factoring: finite fields are key

Problem 1 (Univariate factoring) Given a univariate polynomial $f(x)$ over a field $\mathbb{F}$, compute pairwise distinct irreducible polynomials $f_1, \ldots, f_r \in \mathbb{F}[x]$ such that

$$f \;=\; f_1^{e_1} \cdots f_r^{e_r},$$

where $(e_1, \ldots, e_r) \in \mathbb{N}^r$.

Considered over the rational numbers $\mathbb{Q}$, the polynomial $x^2 - 2$ is irreducible, whereas it factors as $(x-3)(x-4) \mod 7$. Evidently, the problem critically depends on the field $\mathbb{F}$. We begin by considering factorization over a *finite field* $\mathbb{F}_q$ of order $q = p^a$, for some prime p. Despite its simplicity, this case involves several non-trivial ideas that serve as the foundation for more general algorithms, including those over the rationals and algebraic number fields.

The input polynomial $f(x) = \sum_{i=0}^{d} c_i x^i \in \mathbb{F}_q[x]$ is given in the *dense representation* as a list of (coefficient, exponent) pairs (c_i, i) for all $0 \le i \le d$. Let $t \in \mathbb{F}_p[y]$

be some irreducible polynomial of degree a, such that $\mathbb{F}_q \cong \mathbb{F}_p[y]/\langle t(y)\rangle$. Given such a t, arithmetic operations in $\mathbb{F}_q$ can be performed in time $\mathrm{poly}(\log q)$. Hence, we assume that the irreducible polynomial t is part of the input to the algorithm. A thorough exposition of computational issues in finite fields is available in [GS13] and [AB09, Section A.4].

An important subroutine is computing the greatest common divisor (GCD) of two polynomials, a task that underlies many factoring algorithms. The classical Euclidean algorithm can be used to efficiently compute the gcd [GG13, Section 3]. For the sake of brevity, we will use the notation $\mathrm{poly}(n)$ to denote any function of the form $O(n^c)$ for some constant c.

Lemma 1 (GCD) *Let $\mathbb{F}$ be a field and $f, g \in \mathbb{F}[x]$ be polynomials of degree at most d. Then, $\gcd(f, g)$ can be computed in $\mathrm{poly}(d)$ operations over $\mathbb{F}$. Moreover, the algorithm returns $a, b \in \mathbb{F}[x]$ such that $a \cdot f + b \cdot g = \gcd(f, g)$.*

Most factoring algorithms over finite fields, and beyond, follow a standard three-step structure (see Figure 1):

1. **Square-free Factorization:** Note that each $e_i \geq 1$ in Problem 1. The *square-free* part of f, denoted by $\mathrm{rad}(f) = f_1 \cdots f_r$, is called the *radical* of f. This can be efficiently computed by repeatedly applying $f/\gcd(f, \partial_x f)$. If $\partial_x f = 0$ then f is a perfect p-th power. We can apply the Frobenius automorphism to extract its p-th root, effectively computing $f^{1/p}$.
2. **Distinct Degree Factorization:** The goal in this step is to decompose $\mathrm{rad}(f)$ into a product of polynomials, each consisting of irreducible factors of the same degree. That is, we aim to compute $\tilde{f}_1, \ldots, \tilde{f}_d$ such that each factor $\tilde{f}_i$ is the product of all irreducible degree-i factors. A key result from field theory proves that $\tilde{f}_i = \gcd(x^{q^i} - x, f)$, which can be efficiently computed using repeated squaring and Lemma 1. In practice, for high-degree random polynomials, this step often dominates the running time of the overall factoring process.
3. **Equal Degree Factorization:** After the above preprocessing, we are left with the task of factoring a polynomial into irreducibles of the same degree. Two classical algorithms for this purpose are Cantor–Zassenhaus and Berlekamp's algorithm. While Berlekamp's method is deterministic, Cantor–Zassenhaus is randomized and tends to scale better for large finite fields. Both approaches fundamentally exploit the structure given by the Chinese Remainder Theorem.

In the remainder of this section, we focus on algorithms for *Equal Degree Factorization*. After performing the two preprocessing steps, we are left with factoring a

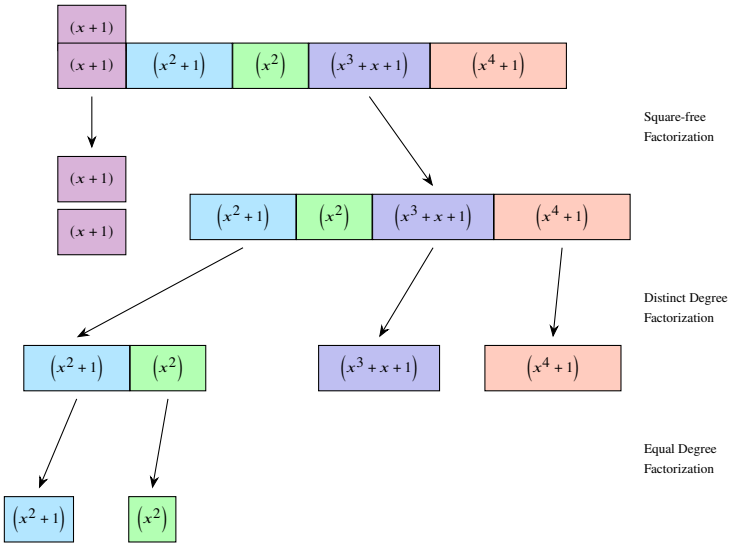

Fig. 1 Polynomial factorization steps. Adapted from [GS92].

polynomial of the form $f = f_1 \cdot f_2 \cdots f_r$, where each f_i is an irreducible polynomial of the same degree.

1.1.1 Berlekamp's algorithm

At the core of Berlekamp's algorithm lies a fundamental number-theoretic observation: suppose there exists a polynomial $h \in \mathbb{F}_q[x]$ such that

$$h^p - h \equiv 0 \pmod{f},$$

and $1 \leq \deg(h) < \deg(f)$. Then, by the identity $h^p - h = \prod_{\alpha \in \mathbb{F}_q}(h - \alpha)$, the polynomial f can be factored by computing $\gcd(f, h - \alpha)$ for all $\alpha \in \mathbb{F}_q$.

The existence of such a polynomial h can be proved using the Chinese Remainder Theorem. To find such an h, the algorithm considers the $\mathbb{F}_p$-vector space

$$V := \{h(x) : \deg(h) \leq d, h^p - h = 0 \pmod{f}\}. \tag{1}$$

This is a subspace of $\mathbb{F}_q[x]/\langle f \rangle$ of dimension at most $\deg(f) \cdot a$, and a basis for V can be efficiently computed using standard linear algebra over $\mathbb{F}_p$. Any non-trivial (i.e., non-constant) basis element $h \in V$ yields a non-trivial factor of f via GCD computations.

To avoid q-many GCD computations, elements of $\mathbb{F}_q$ are represented as a vectors over $\mathbb{F}_p$. Notably, Berlekamp's algorithm does not require the input polynomial to have irreducible factors of the same degree; thus, the distinct-degree factorization step can be bypassed. With careful implementation, the total runtime is bounded by $\mathrm{poly}(p, d, \log q)$, making it a deterministic polynomial-time algorithm in small characteristic, e.g., when $p = (da)^{O(1)}$. For an accessible exposition, see the lecture notes by Kopparty [Kop14, Lecture 10].

1.1.2 Cantor-Zassenhaus Algorithm

When working over large fields, Berlekamp's algorithm—though deterministic—may become inefficient. The Cantor–Zassenhaus algorithm offers a more efficient alternative for factoring over fields of large characteristic by leveraging randomness. The key idea is to choose a polynomial $h \in \mathbb{F}_q[x]$ of degree at most $d-1$ uniformly at random, and check whether the following gcd yields a non-trivial factor:

$$\gcd\left(f,\ h^{\frac{q^d-1}{2}} - 1 \bmod f\right).$$

To see why this works, recall that $f = f_1 \cdot f_2 \cdots f_r$, obtained after the preprocessing steps. Since each f_i is irreducible of degree at most d, the Chinese Remainder Theorem yields an isomorphism:

$$\frac{\mathbb{F}_q[x]}{\langle f\rangle} \cong \frac{\mathbb{F}_q[x]}{\langle f_1\rangle} \times \cdots \times \frac{\mathbb{F}_q[x]}{\langle f_r\rangle}.$$

Let $h \in_R \mathbb{F}_q[x]/\langle f\rangle$ map to $(h_1, \ldots, h_r)$ under this isomorphism. Then, $h^{q^d} - h = 0$. Assuming q is odd, it follows that

$$h \cdot \left(h^{\frac{q^d-1}{2}} - 1\right) \cdot \left(h^{\frac{q^d-1}{2}} + 1\right) = 0.$$

Furthermore, we know that $h^{q^d} - h = \prod_{\alpha \in \mathbb{F}_{q^d}} (h - \alpha)$. This implies that, besides 0, exactly half the elements of $\mathbb{F}_{q^d}$ satisfy $h^{\frac{q^d-1}{2}} = 1$, and the other half satisfy $h^{\frac{q^d-1}{2}} = -1$. Consequently, with probability at least $1/2$, the image of $h^{\frac{q^d-1}{2}} - 1$ under the CRT map will be zero in at least one coordinate. Therefore, computing $\gcd(f, h^{\frac{q^d-1}{2}} - 1)$ yields a non-trivial factor of f with good probability. To eliminate the possibility of h getting mapped to the identity element in all coordinates (yielding

a trivial GCD), we begin by checking whether our randomly chosen h is already a non-trivial factor of f.

When q is even, we instead compute $h' = h + h^2 + h^{2^2} + \cdots + h^{2^{rd-1}}$ and check whether $\gcd(f, h')$ is non-trivial. This approach works because one can show that $h'(h'+1) = h^{2^{rd}} + h$, enabling a similar probabilistic argument. The algorithm relies primarily on computing gcds and repeated squaring. The probabilistic bounds imply that, in expectation, only a constant number of repetitions (at most two) are required. Thus, the overall complexity of the Cantor–Zassenhaus algorithm is $\text{poly}(d, \log q)$.

Question 1 (Open) Is there a derandomization result for the finite field factoring algorithm?
E.g. Given a, p in the input, can we find $\sqrt{a} \bmod p$ in deterministic $\text{poly}(\log p)$ time? See [Iva+12] for a detailed survey on this line of work.

1.2 Fine-grained developments

A finer analysis of the algorithms discussed so far reveals that the overall complexity of univariate factoring using the Cantor-Zassenhaus algorithm is $O(d^2 + o(1) \log q)$ operations in $\mathbb{F}_q$. By improving on the Equal-degree Factorization step, von zur Gathen and Shoup got it down to $O(d^{2+o(1)} + d^{1+o(1)} \log q)$ operations in $\mathbb{F}_q$ [GS92]. Kaltofen and Lobo then improved Berlekamp's algorithm to match the complexity of von zur Gathen and Shoup's algorithm in the black-box linear algebra model [KL94]. Kaltofen and Shoup breached the quadratic barrier by giving an algorithm with complexity $O(d^\omega + d^{1+o(1)} \log q)$ operations in $\mathbb{F}_q$, where ω is the matrix multiplication exponent [KS98]. By giving a significantly novel approach for computing modular composition, which is used to compute h' in the Cantor-Zassenhaus algorithm, Kedlaya and Umans improved the complexity of univariate factoring to $O(d^{1.5+o(1)} + d^{1+o(1)} \log q)$ operations in $\mathbb{F}_q$ [KU11].

2 Model of computation

We will mostly focus on *arithmetic/algebraic circuits*, a model very natural for computing multivariate polynomials in the variables $\boldsymbol{x} \coloneqq (x_1, \ldots, x_n)$[1] over a

[1] We will use bold letters to denote tuples of variables.

field $\mathbb{F}$. Introduced by Valiant [Val79] to develop the algebraic analogue of NP-completeness [Val82], it has led to the development of a rich and varied theory of algebraic complexity (see [Bür24]).

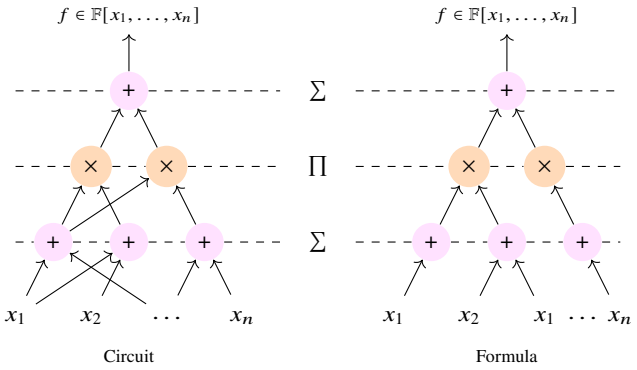

Fig. 2 An algebraic circuit and a formula of depth 3.

Definition 1 (Algebraic Circuits and Formulas) An *algebraic circuit*, defined over a field $\mathbb{F}$, is a *layered directed acyclic graph* with alternating layers of '+' and '×' gates, and a single root, called the 'output' gate. The 'input' *leaf* gates are labeled by either a variable from $x_1, \ldots, x_n$ or a constant from $\mathbb{F}$. If the graph is a *tree*, then we call it a *formula*. See Figure 2 for an illustration.

A circuit computes a polynomial $f \in \mathbb{F}[\boldsymbol{x}]$ in the natural way: a '+' gate sums up the polynomials from its children, whereas a '×' gate computes their product, with the root finally computing f. The *size* of a circuit is the total number of vertices in the graph. The *depth* of the circuit is the number of layers in the circuit, or equivalently, the length of the longest path from the root to a leaf.

A *complexity class* in the algebraic world is a family/sequence of polynomials $(f_n)_{n\in\mathbb{N}}$ where f_n is a multivariate polynomial over some field $\mathbb{F}$. We will mostly be interested in families where the number of variables in f_n grows as a polynomial function of n.

In Boolean complexity, the notion of efficient computation is captured by the class P of problems solvable in polynomial time. The algebraic analogue over a field $\mathbb{F}$ is $\mathsf{VP}_{\mathbb{F}}$ (for Valiant's P, called p-computable by Valiant) and consists of all polynomial families where f_n has degree $\mathrm{poly}(n)$ and the smallest circuit (over $\mathbb{F}$) computing f_n has size $\mathrm{poly}(n)$ (Definition 3). We will usually drop the field from the notation when

the context makes it clear. Note that the notion of computation is *non-uniform* – the circuits of f_n for different n need not be related to one another. A prime example of a polynomial family in VP is (Det_n), defined by the *determinant* of the $n \times n$ symbolic matrix $(x_{ij})_{1 \le i,j \le n}$:

$$\mathsf{Det}_n = \sum_{\sigma \in S_n} \left(\mathrm{sgn}(\sigma) \prod_{i=1}^{n} x_{i,\sigma(i)} \right).$$

The algebraic analogue of the class NP is called VNP. Informally[2], it consists of polynomial families which are 'explicit', in the sense that given a monomial of f_n, we can compute the corresponding coefficient efficiently, say in polynomial time. It is not hard to show that $\mathsf{VP} \subseteq \mathsf{VNP}$, and the long-standing conjecture of Valiant [Val79] is that there are explicit polynomial families that cannot be computed efficiently, i.e., $\mathsf{VP} \subsetneq \mathsf{VNP}$. A prominent 'explicit' candidate for this separation is the family of *permanents*,

$$\mathsf{Per}_n = \sum_{\sigma \in S_n} \left(\prod_{i=1}^{n} x_{i,\sigma(i)} \right).$$

The determinant and permanent families essentially characterize the classes VP and VNP, respectively. Hence, Valiant's conjecture is also sometimes called the Permanent versus Determinant problem [Agr06]. It is the algebraic version of Cook's hypothesis [Coo71], the famous P vs. NP problem (see [AB09] for more details). There is a formal sense in which the VP vs. VNP problem is a 'stepping stone' towards the P vs. NP problem [Bür00b]. For details on the connection between Valiant's and Cook's hypotheses, and the progress on Valiant's conjecture, we encourage readers to consult [BCS97; Bür99; Bür00a; SY10; CKW10; Mah14; Sap21].

2.1 Structural results

Algebraic circuits impose a combinatorial structure on the polynomials being computed. We will now list (without proofs) some structural properties of algebraic circuits that showcase the robustness of the model and will be useful for us in the future. For proofs, see [SY10; Sap21]. To begin with, we can extract coefficients (with respect to a single variable) of a polynomial computed by a small circuit efficiently.

[2] For the formal version, see Definition 6.

Lemma 2 (Interpolation) *Let $\mathbb{F}$ be a field with $|\mathbb{F}| > k$, and $f \in \mathbb{F}[\boldsymbol{x}, y]$ be a polynomial with $\deg_y(f) = k$. Suppose that $f(\boldsymbol{x}, y) = \sum_{j=0}^{k} f_j(\boldsymbol{x}) y^j$ where $f_j \in \mathbb{F}[\boldsymbol{x}]$ for all $j \in \{0, 1, \ldots, k\}$.*

If $f(\boldsymbol{x}, y)$ can be computed by a circuit of size s and depth Δ, then for all $j \in \{0, 1, \ldots, k\}$, $f_j(\boldsymbol{x})$ can be computed by a circuit of size $O(sk)$ and depth Δ.

As we are concerned only with multivariate *polynomials*, we can always formally define the partial derivative of a polynomial with respect to a variable (and by extension, multiple variables) over any field $\mathbb{F}$. Computing the partial derivatives of a circuit with respect to a variable of bounded individual degree is an efficient operation.

Lemma 3 (Partial Derivatives) *Let $\mathbb{F}$ be a field with $|\mathbb{F}| > r$, and $f \in \mathbb{F}[\boldsymbol{x}, y]$ be a polynomial with $\deg_y(f) = r$. If $f(\boldsymbol{x}, y)$ can be computed by a circuit of size s and depth Δ, then for all $0 \leq j \leq r$, the partial derivative $\partial_y^j f(\boldsymbol{x}, y)$ can be computed by a circuit of size $O(sr^3)$ and depth Δ.*

The observant reader will have noticed the conspicuous absence of divisions in our definition of algebraic circuits. It turns out that our notions of complexity do not depend on this exclusion, for the most part. Division can be eliminated efficiently [Str73].

Lemma 4 (Division Elimination) *Let $f \in \mathbb{F}[\boldsymbol{x}]$ be a polynomial of degree d computed by a circuit (with division gates) of size s. Then, there exists a circuit without division gates of size $\mathrm{poly}(s, d, n)$ that computes f.*

In the univariate factoring algorithm, we saw that gcd computation was an essential ingredient. The following lemma shows that it can be computed efficiently in the algebraic circuit model. Refer to [KSS15, Lemma 2.9] for the complete proof.

Lemma 5 (GCD in circuits) *Let $f, g \in \mathbb{F}[\boldsymbol{x}]$ be two n-variate polynomials of degree at most d computed by circuits of size s. Then, the $\gcd(f, g)$ can be computed by a circuit of size $\mathrm{poly}(s, d, n)$.*

3 Applications of factoring

Polynomial factoring is not only a great mathematical problem. The techniques developed for its solution have a wide range of applications in various areas of computer science. We briefly describe some of them.

3.1 Equivalence to identity testing

Consider a class of polynomials C that can be computed by algebraic circuits. The *polynomial identity testing* (PIT) problem for C asks whether a given polynomial $f \in C$ is identically zero. Due to the PIT Lemma [Zip79; DL78; Sch80; Ore22], a simple randomized algorithm for this problem has been known for decades. However, designing a polynomial-time deterministic algorithm for PIT for the class VP remains a long-standing open problem. Nevertheless, several efficient deterministic algorithms have been developed for restricted circuit classes. The problem is studied in two settings: the *black-box* setting, where only polynomial evaluation is permissible, and the *white-box* setting, where the internals of the circuit are accessible.

The importance of this fundamental problem stems from its applications to equivalence checking, perfect matching, primality testing, and more. The assumption of efficient identity testing algorithms has led to a variety of derandomization results. Most notably, PIT is tightly connected to proving strong lower bounds for algebraic circuits (more in Section 3.2). For a comprehensive treatment, we refer the reader to classical surveys such as [SY10; Sax09; Sax14], as well as the recent exposition in [DG24] and the references therein.

PIT was first linked to factorization when Shpilka and Volkovich [SV10] observed that the polynomial $f(\boldsymbol{x})+yz$ has two irreducible factors over disjoint sets of variables if and only if $f(\boldsymbol{x})$ is identically zero. This observation implies that a deterministic algorithm for multivariate polynomial factorization would suffice to derandomize PIT.

The connection between the two problems was further solidified by the work of Kopparty, Saraf, and Shpilka [KSS15], who showed that a derandomized PIT algorithm also leads to a deterministic multivariate factoring algorithm. Together, these results establish the equivalence between derandomizing PIT and polynomial factorization in both black-box and white-box settings.

A natural and simpler question related to polynomial factoring is that of *divisibility testing*: given two polynomials f and g, determine whether g divides f. One could factor both f and g followed by comparing the irreducible factors using PIT to solve this problem using randomization. However, since non-trivial deterministic PIT algorithms are known in several restricted settings, it is natural to ask whether divisibility testing is easier in such settings.

This question was first studied by Saha, Saptharishi, and Shpilka [SSS13], who reduced the problem of testing divisibility of sparse polynomial by a linear polynomial

to PIT for expressions of the form $\sum_i \boldsymbol{x}^{\boldsymbol{a}_i} \cdot f_i(\boldsymbol{x})^{d_i}$, where each $\boldsymbol{x}^{\boldsymbol{a}_i}$ is a monomial and $\deg f_i \leq 1$. Known PIT algorithms for such structured polynomials were applied to obtain efficient and deterministic divisibility testing algorithms (see [RS05; FS13]). Later, Forbes [For15] extended this line of work to test whether a sparse polynomial divides a quadratic polynomial. The problem in this case was reduced to PIT for almost similar polynomials of the form $\sum_i \boldsymbol{x}^{\boldsymbol{a}_i} \cdot f_i(\boldsymbol{x})^{d_i}$ with $\deg f_i \leq 2$, and gave a quasipolynomial-time algorithm for this case. For certain classes of polynomials, Forbes also showed a general reduction to PIT [For15, Section 7]. For a more comprehensive discussion on these equivalences, we refer the reader to the survey by Shpilka and Forbes [FS15, Section 5].

3.2 Hardness vs Randomness

In the previous section, we hinted at the tight connection between the derandomization of PIT and strong lower bounds for algebraic circuits. Informally, this connection implies that PIT can be efficiently derandomized if and only if there exist explicit polynomials of high circuit complexity. Interestingly, factoring results on the algebraic circuit classes play a surprising yet pivotal role in establishing this connection.

Heintz and Schnorr [HS80], and later Agrawal [Agr05], showed that a black-box PIT algorithm for algebraic circuits of size s, running in time $\mathrm{poly}(s)$, would imply the existence of a polynomial whose coefficients are computable in polynomial space (PSPACE) but which requires algebraic circuits of size at least $\exp(\mathrm{poly}(s))$. This yields an exponential lower bound from a polynomial-time PIT algorithm. A central open question is whether the complexity of computing the coefficients can be reduced to $\#\mathsf{P}/\mathsf{poly}$, as this would imply a separation of VP from VNP [SY10, Open Problem 17]. Kabanets and Impagliazzo [KI04] further strengthened this direction of the connection by proving that derandomizing PIT, even in the weaker white-box setting, would imply either $\mathsf{VP} \neq \mathsf{VNP}$ or $\mathsf{NEXP} \not\subset \mathsf{P}/\mathrm{poly}$.

Kabanets and Impagliazzo further strengthened the connection by studying the reverse direction, drawing inspiration from analogous results in the Boolean setting by Nisan and Wigderson [NW94]. To describe this, we introduce an alternative notion of black-box identity testing. Consider a polynomial map $\mathcal{G} \coloneqq (g_1, \ldots, g_n) : \mathbb{F}^r \to \mathbb{F}^n$, where the seed $r < n$. The map $\mathcal{G}$ is called a *hitting-set generator* for a class of polynomials C if, for every non-zero $f \in C$, the composed polynomial $f \circ \mathcal{G} = f(g_1(\boldsymbol{y}), \ldots, g_n(\boldsymbol{y}))$ is also non-zero. For a detailed discussion on the

connections between hitting-set generators and PIT, see [For14, Section 3.2.2]. The high-level idea is that, if one can construct a hitting-set generator $\mathcal{G}$ with a seed length $r = O(1)$, then identity testing for any $f \in C$ reduces to checking whether the constant-variate polynomial $f \circ \mathcal{G}$ is identically zero–a problem efficiently solvable using the PIT lemma.

To illustrate the idea of a hitting-set generator in action, consider a generator with seed length $(n-1)$ defined as $\mathcal{G}(\boldsymbol{y}) = (y_1, \ldots, y_{n-1}, g(\boldsymbol{y}))$, where g is a hard polynomial. This generator extends $r = n-1$ variables by one additional coordinate using a hard polynomial. Now, suppose $f(\boldsymbol{y}, x)$ is a non-zero polynomial in C. For the sake of contradiction, assume that $f \circ \mathcal{G} = f(y_1, \ldots, y_{n-1}, g(\boldsymbol{y})) = 0$. This implies that $(x - g)$ divides f. As we will discuss later (Section 4.4), if f is computable by a small algebraic circuit, then so are all of its factors. Therefore, $(x - g)$ must also be computable by a small circuit, contradicting our assumption that g has large circuit complexity. Hence, $\mathcal{G}$ is as a valid hitting-set generator for C.

To derandomize PIT using the above approach, one requires a generator that stretches a small seed r to significantly more than just one additional variable. Kabanets and Impagliazzo [KI04], drawing inspiration from the Nisan-Wigderson design [NW94], constructed such a generator $\mathcal{G}$ using combinatorial techniques and a hard polynomial g. Their construction yields a generator that stretches a seed of length $r = \text{poly}(\log n)$ to n variables. Applying the same reasoning as before, this generator leads to a quasipolynomial-time PIT algorithm. Crucially, to establish that their construction is indeed a hitting-set generator, they use a hybrid argument and fundamentally rely on factor closure of circuits–which will be explored in detail in the upcoming parts of this survey.

Theorem 1 (Combinatorial Hardness Implies PIT) *Let $k \coloneqq O(1)$. Suppose there exists a family $\{f_n\}_{n \in \mathbb{N}}$ of n-variate multilinear polynomials such that each f_n requires algebraic circuits of size $2^{\Omega(n)}$. Then, there exists a black-box PIT algorithm running in time $2^{\text{poly}(\log n)}$ for the class of n-variate degree-$\text{poly}(n)$ polynomials computable by circuits of size $\text{poly}(n)$.*

Later, Andrews [And20] showed that the derandomization of PIT from hardness assumptions holds even over fields of characteristic p. This improvement hinged on a key result: the p-th root of a circuit computing p-th power can be computed efficiently by a small circuit, provided the number of variables is bounded.

Theorem 1 was subsequently improved with better parameters by the authors of [Guo+22]. They showed that a constant-variate hard polynomial can imply a

polynomial-time black-box PIT algorithm. A key contribution in their work is the use of an *algebraic* generator, in contrast to the *combinatorial design*-based generator used in [KI04]. The combinatorial design inherently suffers from limitations–most notably, it cannot yield better than a quasipolynomial-time PIT algorithm.

Theorem 2 (Non-combinatorial Hardness-to-PIT) *Let $k \coloneqq O(1)$. Suppose $\{f_d\}_{d\in\mathbb{N}}$ is a family of k-variate polynomials of degree d, and each f_d requires algebraic circuits of size at least $d^{0.1}$. Then, there exists a black-box PIT algorithm running in time $s^{O(k^2)}$ for the class of s-variate, degree-s polynomials computable by algebraic circuits of size s.*

We conclude this section by highlighting an important application of the hardness versus randomness paradigm–namely, the phenomenon of *bootstrapping* [AGS19; KST23; Guo+22]. This idea relies on recursively leveraging the connection between hardness and polynomial identity testing (PIT). Remarkably, it shows that a complete derandomization of PIT can be achieved from even a mildly non-trivial derandomization of PIT.

Theorem 3 (Bootstrapping) *Suppose there exists a black-box PIT algorithm that runs in time $\left(s^k - 1\right)$ for the class of k-variate polynomials of individual degree s, computable by algebraic circuits of size $s^{0.1}$. If s is sufficiently large, then there exists a black-box PIT algorithm running in time $s^{O(k^2)}$ for s-variate polynomials computable by algebraic circuits of size s.*

For a more detailed exposition of this connection between factoring and its implications for the hardness versus randomness frontier, we refer the reader to the lucid survey by Kumar and Saptharishi [KS19].

3.3 Other applications

The field of *coding theory* [GRS23] deals with developing *error-correcting codes* – ways of adding (minimal) redundancy to data such that even if parts of it get corrupted during transmission, one can recover the original information. Reed-Solomon codes are particularly ubiquitous and also 'optimal', in a sense. They treat the original message as a univariate polynomial and the encoding is the evaluation of this polynomial at various points over some finite field. When the number of errors is too large, we cannot decode a corrupted message uniquely, but we can produce a small

list of potential decodings (also known as *list decoding*). The list decoding algorithm of Sudan [Sud97] and the later improvement by Guruswami and Sudan [GS99] crucially use polynomial factorization. We point the reader to the survey of Forbes and Shpilka [FS15, Section 3.1] for more details.

The problem of learning algebraic circuits is called *reconstruction* [SY10, Chapter 5]. We are given black box access to a polynomial computed by a circuit C from some nice family of circuits $\mathcal{C}$, and we need to 'learn' an arithmetic circuit computing the same polynomial as C. Efficient polynomial factorization plays an important role in many reconstruction algorithms [KS09; Sin16; Sin22; SS25]. Polynomial factorization is also helpful in algebraic property testing [AS03] and the construction of pseudorandom generators for low-degree polynomials [Bog05; DGV24].

In *proof complexity*, a central problem is to prove that certain propositional tautologies need extremely lengthy proofs, even in very powerful proof systems. An important work of Cook and Rechow [CR79] showed that such proof complexity lower bounds, provided we are able to show them for every propositional proof system, would separate the complexity classes NP and coNP, and in turn, also P from NP. Closure of a class under factoring is another way of saying that multiples of hard polynomials from the class remain hard (more generally, one can study the complexity of ideals [Gro20]). Using such hard multiples, Forbes, Shpilka, Tzameret, and Wigderson [For+21] showed lower bounds against certain algebraic proof systems. For more about proof complexity, see [Kra95; Juk12; Kra19].

Polynomial factorization also has applications to various other problems in mathematics, such as derandomizing Noether's Normalization Lemma [Mul17], the primary decomposition of polynomial ideals [GTZ88], and isomorphism of algebras [KS06; Iva+12]. In cryptography, polynomial factorization is often used as a subroutine—for example, in index calculus algorithms and public-key encryption [MOV97, Chapter 3], and in factoring integers [Bre00]. Polynomial factorization algorithms—and the tools developed alongside them—have proven invaluable in the cryptanalysis of lattice-based schemes [NV10] as well as post-quantum cryptosystems [DPS20].

4 Factoring via Hensel lifting

Hensel lifting was first introduced by Kurt Hensel in a series of papers [Hen97; Hen04; Hen08; Hen18], although an earlier form of it was known to Gauß [Fre07]. For an element p in a ring $\mathcal{R}$, Hensel lifting gives a method to compute factorization

modulo p^ℓ (for any $\ell > 0$) from the factorization modulo p. The term *lifting* refers to improving the approximation of the factors with each iteration. The explicit algorithm has applications in various areas of mathematics, including number theory, and computer algebra.

Theorem 4 (Hensel Lifting) *Let $\mathcal{R}$ be a ring and $\mathcal{I}$ be an ideal in $\mathcal{R}$. Consider elements $f, g, h \in \mathcal{R}$ such that $f \equiv g \cdot h \pmod{\mathcal{I}}$ and there exist $a, b \in \mathcal{R}$ such that $a \cdot g + b \cdot h \equiv 1 \pmod{\mathcal{I}}$. Then, we have:*

1. **Existence.** *There exists $g', h' \in \mathcal{R}$ such that $f \equiv g' \cdot h' \pmod{\mathcal{I}^2}$ and*

$$g' \equiv g \pmod{\mathcal{I}} \quad \text{and} \quad h' \equiv h \pmod{\mathcal{I}}$$

2. **Pseudo-Coprimality.** *For some $a' \equiv a \pmod{\mathcal{I}}$ and $b' \equiv b \pmod{\mathcal{I}}$, we have $a' \cdot g' + b' \cdot h' \equiv 1 \pmod{\mathcal{I}^2}$.*
3. **Uniqueness.** *If any other $\tilde{g}, \tilde{h}$ satisfy the above conditions, then there must be a $u \in I$ such that $\tilde{g} \equiv g'(1+u) \pmod{\mathcal{I}^2}$ and $\tilde{h} \equiv h'(1-u) \pmod{\mathcal{I}^2}$.*

The polynomials g' and h' are called the lifts of g and h, respectively. The theorem is not only existential but also constructive–it yields an explicit algorithm to compute the lifts. Let $e \coloneqq f - gh$. Then the lifts can be obtained as follows:

$$g' \coloneqq g + e \cdot b \quad \text{and} \quad h' \coloneqq h + e \cdot a. \tag{2}$$

To use Hensel lifting for polynomial factoring, we consider $\mathcal{R} = \mathbb{F}[x, y]$ and $\mathcal{I} = y^k$ for some $k \geq 1$. In the case of a multivariate polynomial, we transform it to a bivariate polynomial. We will discuss this in more detail in the upcoming sections. When working over a ring of polynomials, a monic version of the theorem further guarantees the uniqueness of the monic lifts.

Theorem 5 (Monic Hensel Lifting) *Let $\mathbb{F}[x, y]$ be a ring of polynomials. Consider monic polynomials $f, g, h \in \mathbb{F}[x, y]$ such that $f \equiv g \cdot h \pmod{y}$ and there exist $a, b \in \mathbb{F}[x, y]$ such that $a \cdot g + b \cdot h \equiv 1 \pmod{y}$. Then, we have:*

1. **Existence.** *There exists monic polynomials $g', h' \in \mathbb{F}[x, y]$ such that $f \equiv g' \cdot h' \pmod{y^2}$ and*

$$g' \equiv g \pmod{y} \quad \text{and} \quad h' \equiv h \pmod{y}$$

2. **Uniqueness.** *If any other monic $\tilde{g}, \tilde{h}$ satisfy the above conditions, then $\tilde{g} \equiv g' \pmod{y^2}$ and $\tilde{h} \equiv h' \pmod{y^2}$.*

3. **Pseudo-Coprimality.** *For some* $a' \equiv a \pmod{y}$ *and* $b' \equiv b \pmod{y}$*, we have* $a' \cdot g' + b' \cdot h' \equiv 1 \pmod{y^2}$.

Once again let $e \coloneqq f - gh$, and define $\hat{g}, \hat{h}$ as in Equation (2). Compute the following expressions using the division algorithm:

$$u \coloneqq \frac{\hat{g} - g}{y} = q \cdot g + r.$$

Then the unique monic lifts can be computed as follows:

$$g' \coloneqq g + y \cdot r \quad \text{and} \quad h' \coloneqq \hat{h} \cdot (1 + q \cdot y) \tag{3}$$

An expository proof of Hensel lifting can be found in [KSS15, Lemma 3.4].

4.1 Dense representation

To demonstrate the use of Hensel lifting in factoring, consider the problem of factoring a bivariate polynomial $f \in \mathbb{F}_q[x, y]$ of degree at most d, where $q = p^a$. The input is given in the dense representation, as in Section 1.1. In fact, the univariate factoring algorithms will be used as subroutines in the bivariate setting. For convenience, it is useful to view the input polynomial f as an element of $\left(F_q[x]\right)[y]$, thereby treating the variable y informally as a constant.

Resultants.

We saw in Section 1.1 that GCD is an important tool for univariate factoring algorithms. A closely related polynomial called the *resultant* plays an important role in several factoring algorithms.

Definition 2 (Resultant) Consider two n-variate polynomials $f, g \in \mathbb{F}[\boldsymbol{x}][y]$ as follows:

$$f(\boldsymbol{x}, y) = \sum_{i=0}^{d_1} f_i(\boldsymbol{x}) \cdot y^i \quad \text{and} \quad g(\boldsymbol{x}, y) = \sum_{i=0}^{d_2} g_i(\boldsymbol{x}) \cdot y^i.$$

Define *Sylvester matrix* of f and g as the following $(d_1 + d_2) \times (d_1 + d_2)$ matrix:

$$\mathbf{S}_y(f,g) \;=\; \begin{pmatrix} f_0 & & & & g_0 & & & \\ f_1 & f_0 & & & g_1 & g_0 & & \\ f_2 & f_1 & \ddots & & \vdots & g_1 & \ddots & \\ \vdots & \vdots & \ddots & & g_{d_2} & \vdots & \ddots & \ddots \\ f_{d_1} & f_{d_1-1} & & f_0 & & g_{d_2} & & \ddots \; g_0 \\ & f_{d_1} & \ddots & f_1 & & & \ddots & \quad g_1 \\ & & \ddots & \vdots & & & \ddots \; \ddots & \vdots \\ & & & f_{d_1} & & & & \ddots \; g_{d_2} \end{pmatrix}$$

Then the resultant of the two polynomials with respect to y is defined as the determinant of the Sylvester matrix as:

$$\mathsf{Res}_y(f,g) \;\coloneqq\; \mathsf{Det}(\mathbf{S}_y(f,g)).$$

The resultant is a polynomial of degree at most $2(d_1 \cdot d_2)$ in the coefficients of f and g. The following lemma captures the one of several properties of the resultant that are useful in factoring algorithms. See [GG13, Section 6.3] for the detailed proof of the following lemma.

Lemma 6 (Resultant and GCD) *Let $f, g \in \mathbb{F}[x][y]$ such that degree with respect to y is positive. Then the following is true:*

1. $\mathsf{Res}_y(f,g) = 0$ *if and only if f and g share a common factor of positive degree in y.*
2. *There exists a, b such that $a \cdot f + b \cdot g = \mathsf{Res}_y(f,g)$.*

Bivariate Factoring.

The high-level idea of bivariate factoring is to first obtain factorization of f modulo y, and then lift them using Hensel lifting. But before we can do that, we need to ensure that the f satisfy certain properties, as described below.

1. $f(x,0)$ **is square-free:** For $a \in \mathbb{F}_q$, the univariate polynomial $f(x,a)$ is square-free if and only if
$$\gcd\Big(f(x,a), \partial_x f(x,a)\Big) \;\neq\; 1.$$

From Lemma 6, this is equivalent to $\mathsf{Res}_x(f, \partial_x f)|_{y=a} \neq 0$. Since the degree of the resultant in y is at most $2d^2$, it suffices to test $O(2d^2)$ values of $a \in \mathbb{F}_q$ (or a suitable extension) to find one for which $f(x, a)$ is square-free. A simple linear transformation will ensure that the resultant is non-zero at $a = 0$. Further, this would also imply that f is square-free.

2. f **is monic in** x: Let $f_d(x)$ be the leading coefficient of f of degree at most d. Then,
$$\tilde{f} \;=\; f_d^{d-1} \cdot f\left(x/f_d, y\right)$$
is a monic polynomial in x.

For simplicity, we denote the monic and square-free polynomial obtained after the above transformations by f. The significance of ensuring that $f(x, 0)$ is square-free is that it allows us to apply univariate factoring algorithms Section 1.1 to compute polynomials g_0 and h_0 such that

$$f \equiv g_0 \cdot h_0 \mod y,$$

where g_0 is a monic irreducible factor in x. We now use Hensel lifting Theorem 5, repeatedly t times to obtain g_k and h_k satisfying

$$f \;\equiv\; g_k \cdot h_k \mod y^{2^t}.$$

The lifting process is continued until $2^t > 2d^2$, a bound that hints at a connection with resultant polynomials. We address this connection in the proof of the following claim.

Proposition 1 *If the input polynomial f is reducible then there exists a non-trivial factor f_1 of f such that $f_1 \equiv g_k \cdot \ell \pmod{y^{2^k}}$. Further, $\deg_x(f_1), \deg_y(f_1)$ is at most $\deg_x(f)$ and $\deg_y(f)$ respectively.*

Proof. First let us prove that such a factor exists. Since g_0 is an irreducible factor of $f \pmod y$, we know that g_0 must divide some irreducible factor of f, say f_1. Let $f = f_1 \cdot h$. Then, for some ℓ_0,

$$f_1 \equiv g_0 \cdot \ell_0 \mod y.$$

By Theorem 5, we get $f_1 \equiv g_k' \cdot \ell_k \pmod{y^{2^k}}$. Multiplying h both sides gives

$$f \equiv g_k' \cdot h_k' \mod y^{2^k}.$$

From the uniqueness part of Theorem 5, we must have $g'_k = g_k$, and hence $f_1 \equiv g_k \cdot \ell_k \pmod{y^{2^k}}$ exists.

To see that f_1 is non-trivial, let us assume for the sake of contradiction that $\gcd_x(f, f_1) = 1$. Then there exist u, v such that

$$u \cdot f + v \cdot f_1 = \mathsf{Res}_x(f, f_1).$$

Substituting the lifted expressions, we get

$$g_k \cdot (u\, h_k + v\, \ell_k) \equiv \mathsf{Res}_x(f, f_1) \mod y^{2^k}.$$

Since g_k is monic, and $\mathsf{Res}_x(f, f_1) \in \mathbb{F}_q[y]$, it follows that

$$u\, h_k + v\, \ell_k \equiv 0 \mod y^{2^k}.$$

Since 2^k is greater than the degree of the resultant $2d^2$, it follows that $\mathsf{Res}_x(f, f_1) = 0$. This yields a contradiction, and thus $\gcd_x(f, f_1) \neq 1$. □

The above claim suggests that a non-trivial factor of f, if it exist, can be obtained by solving the following linear system:

$$g \equiv g_k \cdot \ell \mod 2^{y^k}, \tag{4}$$

where the degree bounds are same as before.

Finally, we recall that the algorithm factors a polynomial that results from a sequence of preprocessing steps. Hence, it is necessary to undo these steps—particularly the monic transformation—in order to recover a factor of the original input polynomial. By applying the Cantor–Zassenhaus algorithm for univariate factorization and analyzing the cost of lifting factors via Hensel lifting, it can be observed that the overall complexity of bivariate factoring is $\mathrm{poly}(d, \log q)$ operations over $\mathbb{F}_q$.

Naturally, the bivariate factoring algorithm extends to the multivariate setting. However, as the number of variables increases, the runtime of the algorithm degrades rapidly. Later, we will explore how to address this challenge while still reusing insights from the factoring algorithms discussed thus far.

4.2 Factoring polynomials over integers

The finite field factoring algorithm can be extended to factoring polynomials over integers and thereby over rationals as well. Although the details of it is outside the scope of this survey, we will briefly discuss the high-level idea of the algorithm. Consider a degree d input polynomial $f \in \mathbb{Z}[x]$ with coefficients of bit-length at most 2ℓ. Therefore, the coefficients of f are between -2^{ℓ} and 2^{ℓ}, and the goal is to obtain a non-trivial factor of f in $\text{poly}(d, \ell)$ time.

The algorithm for factoring polynomials over the integers closely follows the same template as the bivariate factoring algorithm. It begins by choosing a sufficiently large prime p such that $f \bmod p$ is square-free. The prime p plays a role analogous to the variable y in the bivariate setting. The polynomial f is then factored modulo p using a univariate factoring algorithm (see Section 1.1). The resulting factors are lifted to a factors modulo p^{2^k} via Hensel lifting. As in the bivariate case, it can be shown that $\log(n^3\ell)$ iterations of the lifting step suffice.

The most challenging part of the algorithm lies in solving the linear system similar to Equation (4), under the additional constraint that the coefficients of the factor g are small. Although this is not immediately obvious, the problem reduces to finding a shortest vector in a lattice—-a task that is known to be NP-hard in general.

Nevertheless, the celebrated algorithm of Lenstra, Lenstra, and Lovász [LLL82] provides an efficient *weak-approximation* algorithm for this problem, which turns out to be sufficient for the purposes of polynomial factoring. For a detailed exposition, see [Sap17, Chapter 13].

4.3 p-adic factoring: Chistov, Cantor-Gordon algorithm

There is no *topology*, nor a geometric interpretation, inherent in a finite field. Classically, this motivated mathematicians to 'extend' a finite field $\mathbb{F}_q$ (q is a power of a prime p) to a characteristic *zero* field. The latter is called an *unramified p-adic field* construction.

p-adics. For simplicity, consider the object $\mathbb{Z}_p$, the ring of *p-adic integers*. Its elements are infinite series of the type $(a_0 + a_1 \cdot p + a_2 \cdot p^2 + \cdots)$ with the *digits* a_i's in $[0 \ldots p-1]$. The notion of *convergence* here requires an *ultra*-metric, defined via prime-powers, $|p^i|_u := p^{-i}$. (Exercise: Check that it defines a *metric* in the space $\mathbb{Z}_p \to \mathbb{Q}$.) This forces an *un*real comparison of real numbers: $1 > p > p^2 > \cdots >$

$p^\infty = 0$! The convergence allows certain integers to become invertible or *units*, e.g. $1/(1-p) = 1 + p + p^2 + \cdots \in \mathbb{Z}_p$. It is easy to show: (1) $\mathbb{Z}_p$ is an integral domain, so it has a field of fractions called *p-adic rationals* $\mathbb{Q}_p$, which has non-integer elements like $1/p$ and $1/(p^2 + 2p^3)$. (2) The characteristic of $\mathbb{Z}_p$ and $\mathbb{Q}_p$ is 0.

Given a polynomial $f(x) \in \mathbb{Q}_p[x]$ as a binary string of size n, can we factor it, over $\mathbb{Q}_p$, in time poly(n)? A randomized poly-time algorithm was provided by [Chi87; CG00]. The two papers are written in very different styles. We will sketch the basic ideas common to them, by working through an example.

Consider the 2-adic polynomial $f(x) = (x-2)^2(x-4) + 32 \in \mathbb{Q}_2[x]$. Note that $f \equiv x^4 \bmod 2$ has no coprime factors over $\mathbb{F}_2$, so the algorithm of Section 1.1.2 cannot be applied to get factors, or check irreducibility, 'lifted' to $\mathbb{Z}/4\mathbb{Z}$. This forces us to work modulo higher 2-powers and to perform some new operations. Seeing broadly, there are two major algebraic themes in the algorithm, which we oversimplify for the sake of presentation as follows.

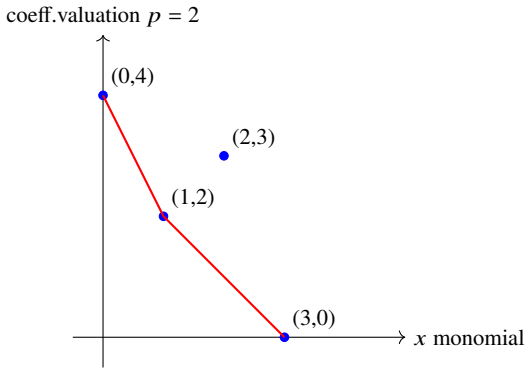

Fig. 3 The Newton Diagram for $f(x)$.

Theme 1 (Newton-Hensel). Consider the *Newton diagram* of $f = x^3 - 8x^2 + 20x + 16$, by plotting the monomial-exponents in the X-axis (namely, $\{0, 1, 2, 3\}$) and the 2-adic valuation of the integral-coefficients in the Y-axis (respectively, $\{4, 2, 3, 0\}$). The lower-boundary of this diagram has two edges of slopes 1 and 2 respectively, suggesting that the $\overline{\mathbb{Q}}_2$-roots of $f(x)$ are (exactly) divisible by 2^1 and 2^2 respectively. This gives us the 'transformation' $f(2x)$ to study, and to find the first factor by a version of *Hensel lifting*. Thus, $f(2x)/8 = (x-2)(x-1)^2 + 4 = (x - 2 + 2^2 + 2^5 + \cdots) \cdot (x^2 - (2 + 2^2 + 2^5 + \cdots)x + 1 - 2^4)$. The key property we use here is the coprimality

of the two factors $(x-2), (x-1)^2 \bmod 2$; so, the algorithm of Section 1.1.2 can be applied.

An inverse transformation $(x \mapsto x/2)$ gives us the two factors of f as, $g_1 := (x - 2^2 + 2^3 + 2^6 + \cdots)$ and $g_2 := (x^2 - (2^2 + 2^3 + 2^6 + \cdots)x + (2^2 - 2^6 + \cdots))$, respectively. Clearly, g_1 is an *irreducible* 2-adic factor of f. What about g_2?

Theme 2 (p-adic ramification). Again from the Newton diagram of g_2 we learn that it has two distinct $\overline{\mathbb{Q}}_2$-roots, both (exactly) divisible by 2^1. So, we study the new transformation $g_2(2x)/4 = (x^2 - (2 + 2^2 + 2^5 + \cdots)x + 1 - 2^4) =: T(x)$. Clearly, $T \equiv (x-1)^2 \bmod 4$ with no coprime factors mod2. In this case both the tricks of Newton and Hensel fail.

We have $T \equiv x^2 - 6x + 1 \equiv (x-3)^2 - 8 \bmod 16$. From here we learn two properties: (1) T is irreducible (yielding a *certificate* of irreducibility for g_2), as $\sqrt{8}$ does not exist in $\mathbb{Q}_2$; and (2) T suggests a more intricate transformation to progress to a deeper root of f, namely, $(x-3) \mapsto (x-3)\sqrt{2}$ over the (*ramified*) field $\mathbb{Q}_2(\sqrt{2})$.

In general, if a repeated number of these steps achieve a ramification degree equal to that of the degree of f, then we have a certificate of irreducibility of f over $\mathbb{Q}_p$.

The above two algebraic themes give a randomized poly-time algorithm to factor $f(x) \in \mathbb{Q}_p[x]$. The time-complexity is based on analyzing the (Galois) symmetries of both the ramified and unramified field extensions of $\mathbb{Q}_p$ that the algorithm constructs.

Question 2 (Open) Given an integral polynomial $f(x) \in \mathbb{Z}[x]$ of degree d and a prime-power p^k, can we factor $f \bmod p^k$, in randomized poly$(dk \log p)$ time?

In this case, there is no unique factorization property, and we cannot use division by p, as it is now a zerodivisor. See [DMS21; CDS24] for a detailed survey.

4.4 Small algebraic circuits

The factoring algorithms discussed in earlier sections assume that the input polynomial is given in the dense representation, that is, as an explicit list of its coefficients. However, algebraic circuits—-introduced in Section 2 provide a much more succinct way to represent polynomials. From now on, we will work over fields of characteristic zero unless otherwise specified. The results also holds if the characteristic is sufficiently greater than the degree of the polynomial under consideration.

Let $f \in \mathbb{F}[\boldsymbol{x}]$ be a n-variate polynomial computed by an algebraic circuit C of size s. In the black-box model, access to C is restricted to evaluation queries only

(see Figure 4). Given such a circuit C and the degree bound $d = \deg(f)$, the goal is to compute the irreducible factors of f.

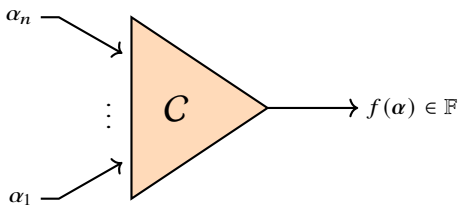

Fig. 4 Black-box access to polynomial via circuit C

A special case of multivariate factorization arises when $f = g^e$, where g is an irreducible polynomial and $e \geq 1$. For fields for large characteristic, the binomial theorem gives a simple way to compute g from f. Start with performing a linear transformation on f to ensure that $f(0, \ldots, 0) = 1$. Then we have

$$\begin{aligned} g \;=\; f^{1/e} \;&=\; \left(1 + (f-1)\right)^{1/e} \\ &=\; \sum_{i=0}^{d} \binom{1/e}{i} \cdot (f-1)^i \mod \langle \boldsymbol{x} \rangle^{d+1} . \end{aligned} \tag{5}$$

The last equality holds because $(f-1)^i$ contributes larger than degree d terms for $i > d$. Algebraic circuits can be efficiently added and multiplied, and further the division required to compute Equation (5) can be eliminated using Lemma 4. Overall, given only blackbox access to C computing f, we can obtain the circuit for g of size $\mathrm{poly}(s, n, d)$.

Blackbox multivariate factoring.

Remarkably, Kaltofen and Trager [KT90] showed that in the black-box setting, efficient circuit factorization is achievable. To describe it, we need to state Hilbert's Irreducibility Theorem, which guarantees that irreducible factors of f maintain their irreducibility profile when restricted to the random subspace.

Theorem 6 (Effective Hilbert Irreducibility Theorem) *Let $S \subseteq \mathbb{F}$ be a large enough finite subset and $g \in \mathbb{F}[\boldsymbol{x}, y]$ be a monic polynomial of degree d, and $\partial_y g$ is*

non-zero. If $|S|$ is at least d^6 and g is irreducible, then for most of the $\overline{\alpha}, \overline{\beta} \in S^n$, the polynomial $g(y, \alpha_1 x + \beta_1, \ldots, \alpha_n x + \beta_n)$ is irreducible.

The effective version of Hilbert's Irreducibility Theorem was first proved by Kaltofen [Kal95, Section 3]. For a clear and accessible exposition, see Sudan's lecture notes [Sud98, Lecture 9].

Consider the polynomial f restricted to a suitable subspace so that the theorem above holds:

$$f_{\alpha,\beta} \coloneqq f\big(y, \alpha_1 x + \beta_1, \ldots, \alpha_n x + \beta_n\big).$$

Let ℓ be the number of irreducible factors of $f_{\alpha,\beta}$ obtained using bi-variate factoring algorithm (see Section 4.1). Assuming f is monic in y and $\partial_y f$ is non-zero, one can prove using Theorem 6 that f has ℓ irreducible factors with high probability. Note that, since the bivariate factoring algorithm requires the input polynomial to be in dense representation, the coefficients of $f_{\alpha,\beta}$ are computed via interpolation (ref Lemma 2).

Given a point $(a, b_1, \ldots, b_n) \in \mathbb{F}^{n+1}$ and an index i, we want to compute the i-th factor $f_i(a, b_1, \ldots, b_n)$. For this define a trivariate polynomial which captures both projection of f on $\overline{\alpha}, \beta$ and the evaluation point $(a, b_1, \ldots, b_n)$ as follows:

$$\hat{f}(y, z_1, z_2) \coloneqq f\big(y, \alpha_1 z_1 + \beta_1 + (b_1 - \beta_1) z_2, \ldots, \alpha_n z_1 + \beta_n + (b_n - \beta_n) z_2\big),$$

where each $x_i \mapsto \alpha_i z_1 + \beta_i + (b_i - \beta_i) z_2$. Note that $\hat{f}(a, 0, 1) = f(a, b_1, \ldots, b_n)$ and $\hat{f}(y, x, 0) = f_{\alpha,\beta}(y, x)$. Using interpolation described in Lemma 2, we can obtain the coefficients of $\hat{f}$. Kaltofen [Kal85] proved that factoring a constant variate polynomial, in particular the trivariate polynomial $\hat{f}$, is efficiently doable using univariate and bivariate factorizations. Let $\left\{\hat{f}_i(y, x_1, x_2) \;:\; i \in [\ell']\right\}$ be the set of irreducible factors of $\hat{f}$. Find an index j such that $\hat{f}_j(y, x, 0) = \tilde{f}_i(y, x)$ and output $\hat{f}_j(a, 0, 1)$.

Circuit factoring

Black-box factoring is a strong notion of factorization, wherein the algorithm is given oracle access to a polynomial and is required to construct oracle access to its factors–without relying on or even knowing the underlying representation or model of computation. Remarkably, this idea can be adapted to show that if a polynomial is computed by a small algebraic circuit, then all its irreducible factors can be computed by small circuits [Kal89]. The complexity class VP contains all polynomial families that can be computed by 'small' circuits.

Definition 3 (VP) A polynomial family $f = (f_n)$ is in the class VP over the field $\mathbb{F}$ if both the number of variables and degree of f_n are bounded by $\text{poly}(n)$ and moreover, the size of the smallest circuit over $\mathbb{F}$ computing f_n, denoted $\text{size}_{\mathbb{F}}(f_n)$ is bounded by $\text{poly}(n)$.

Note that polynomials of degree $\exp(n)$ can be computed by circuits of size $\text{poly}(n)$ by repeated squaring. Such families are not in VP due to the degree restriction in its definition. There are good reasons for imposing this restriction [Gro13]. We will also encounter small circuits of high degree in Section 5.2. The main result of this section is the closure of VP under taking factors.

Over characteristic zero, we can without loss of generality assume that $f = g^e \cdot h$, where g and h are co-prime and $e \geq 1$. The special case of $h = 1$ can be handled as before using Equation (5).

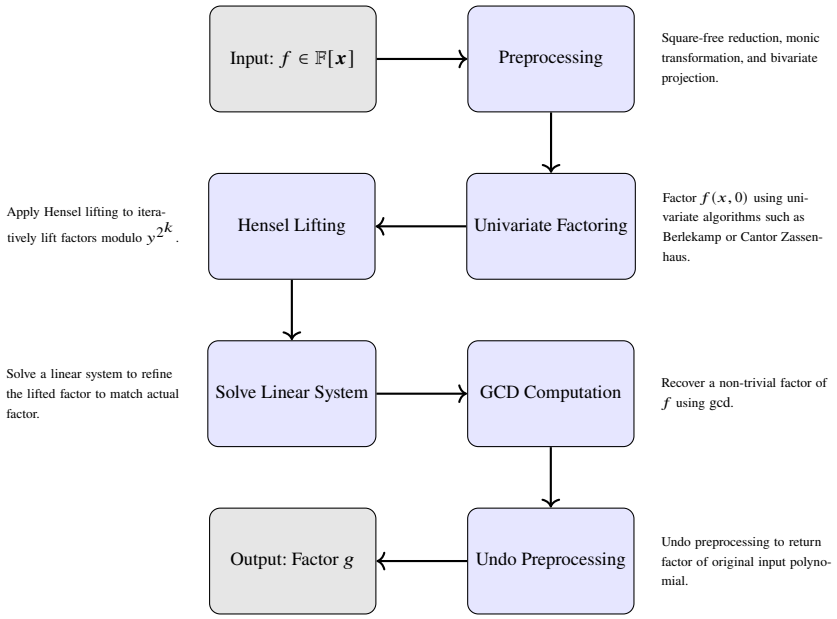

Fig. 5 Overview of Multivariate Factoring using Hensel Lifting.

Theorem 7 (VP factor closure) *Let $f \in \mathbb{F}[\boldsymbol{x}, y]$ be a $(n+1)$-variate, degree d polynomial computable by a circuit of size s. If there are co-prime factors g and h such that $f = g^e \cdot h$, where $e \geq 1$, then g is also computable by a circuit of size* $\text{poly}(s, n, e)$.

Proof Sketch. As in the univariate case, we begin by eliminating repeated factors from f via square-free reduction. This is done by repeatedly dividing f by $\gcd(f, \partial_x f)$, yielding a square-free polynomial. The efficiency of these operations follows from the structural results on algebraic circuits discussed in Section 2.1. After this step, we may assume that $f = gh$, where g and h are coprime. It is important to note that the multiplicity parameter e appearing in the final size bound for the factor originates from this square-free reduction step.

As before, we apply a linear transformation to make f monic in x, followed by a shift to reduce it to a bivariate polynomial (see Theorem 6). After this sequence of transformations, we may assume that $f = gh \in \mathbb{F}[x, y]$, where g and h are coprime, and f is monic in x. Moreover, $g(x, 0)$ and $h(x, 0)$ are also coprime. These conditions together set the stage for applying Hensel lifting. It is important to observe here that these preprocessing steps are reversible and can be efficiently performed on circuits.

We iteratively apply Hensel lifting to compute polynomials g_k and h_k such that $f \equiv g_k \cdot h_k \pmod{y^{2^k}}$. The lifting proceeds until $2^k > 2d^2$(refer to Proposition 1). Each iteration consists of basic algebraic operations outlined in Equation (3), all of which can be performed efficiently by leveraging the structural results of algebraic circuits. Since Hensel lifting yields only an *approximate* factorization modulo y^{2^k}, the actual factor g of f is recovered by solving the linear system $g' \equiv g_k \cdot \ell \pmod{y^{2^k}}$.

By computing the gcd of f with the solution g' of the linear system, we recover a non-trivial factor g of f. Finally, we reverse all preprocessing steps to obtain an algebraic circuit computing g. □

It remains open to extend the closure result to fields of small characteristic.

Question 3 (Open) Is the class VP closed under taking factors over fields of positive characteristic?

As mentioned earlier, Kaltofen [Kal87, Theorem 2] proved a special case of factor closure for polynomials in VP of the form g^e (i.e., $h = 1$ in Theorem 7) over fields of characteristic zero. Andrews [And20] removed the dependence on characteristic in this result for the log-variate regime, thereby making progress toward resolving Question 3. Later, we will discuss a more general progress on this question, where factors of polynomials in VP over finite fields are shown to be *explicit*.

There are other natural classes for which a factor closure result, similar to Theorem 7, does hold. Notably, these include classes that contain VP, and also classes that are contained in VP. We discuss these next.

4.5 Algebraic branching programs

The Algebraic Branching Program (ABP) is another model of computation for polynomials, and in terms of computational power it lies between algebraic circuits and formulas.

Definition 4 (Algebraic Branching Program) An *Algebraic Branching Program* (ABP) is a directed acyclic graph where the edges are labeled by a linear polynomial over a field. The graph has a unique source node s and a sink node t (refer Figure 6). For every path p from s to t, let f_p denote the product of the labels on the edges of p. The polynomial computed by the ABP is the sum of polynomials computed along all the different $s-t$ paths:

$$f = \sum_{p:s \rightsquigarrow t} f_p.$$

The *size* of the ABP refers to the total number of vertices in the graph, while the *length* of the ABP is the length of the longest path from s to t.

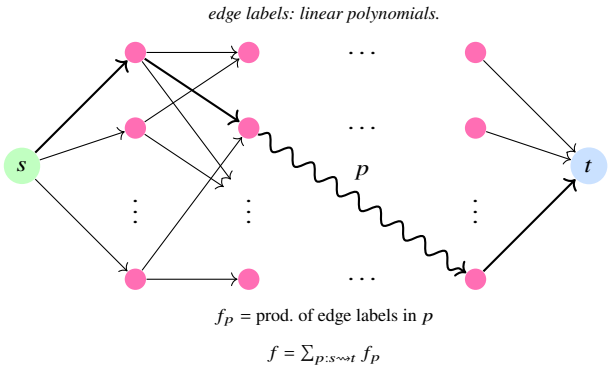

Fig. 6 Algebraic Branching Program (ABP)

Although not immediately evident, ABPs are equivalent to restricted algebraic circuits known as *skew circuits*. In a skew circuit, each multiplication gate is allowed to have at most one child that is not an input gate. For a proof of equivalence between ABPs and skew circuits, see [Mah14]. Similar to general circuits, ABPs are closed under addition and multiplication with only an additive increase in size. Strassen's classical technique for division elimination, originally developed for algebraic circuits,

can be adapted to ABPs as well (see Lemma 4). We state a few non-trivial structural results about ABPs that will be instrumental in the upcoming factor closure result.

Lemma 7 (ABP Closure Properties) *Let $f \in \mathbb{F}[\boldsymbol{x}, y]$ be a $(n+1)$-variate polynomial of degree d computed by an ABP of size s, such that $f = \sum_{i=1}^{d} f_i(\boldsymbol{x})y^i$. Then:*

1. **Coefficient Extraction.** *For any $i \in [d]$, f_i can be computed by an ABP of size* $\mathrm{poly}(s, n, d)$.
2. **GCD.** *For any polynomial g computable by an ABP of size s,* $\gcd(f, g)$ *can be computed by an ABP of size* $\mathrm{poly}(s, n, d)$.
3. **Composition.** *Let $g_1, \ldots, g_n$ be polynomials computable by ABPs of size s. Then the composition $f(g_1, \ldots, g_n)$ can be computed by an ABP of size* $\mathrm{poly}(s, n)$.

Similar to class of small degree polynomials computable by small circuits Definition 3, we can define the class of polynomials computable by small ABPs.

Definition 5 (VBP) A polynomial family $f = (f_n)$ is said to be in the class VBP if the number of variables and degree of f_n are bounded by $\mathrm{poly}(n)$ and furthermore, f_n can be computed by an algebraic branching program of size at most $\mathrm{poly}(n)$.

It is not hard to show that VBP $\subseteq$ VP. In the previous section, we discussed the closure of the class VP under factorization. A natural question is whether a similar closure property holds for VBP. Kaltofen and Koiran [KK08] made progress in this direction by showing that if $f, g \in$ VBP and $f = g \cdot h$, then h also belongs to VBP. Their approach relies on interpreting algebraic branching programs (ABPs) as a restricted class of circuits known as skew circuits discussed earlier. Using this division closure, Jansen [Jan11] leveraged the connection between ABPs and the determinant to adapt the power iteration algorithm for computing eigenvalues, to prove factor closure for polynomials of the form $f = f_1 \cdot (y - f_2) \cdots (y - f_n)$, where the f_i are pairwise distinct. More recently, Sinhababu and Thierauf [ST21] revisited the Hensel lifting method to completely solve the open problem by proving factor closure of VBP.

It is important to observe that the multivariate factoring algorithm from Theorem 7 does not directly extend to Algebraic Branching Programs (ABPs). The main challenge lies in the monic variant of Hensel lifting, which requires division at each step–a costly operation for ABPs when repeated frequently (see Equation (3)). Although composition is technically feasible in ABPs (Lemma 7), the multiplicative blow-up renders repeated use impractical. This limitation also rules out other factoring strategies that rely on efficient composition, which we shall explore in later sections.

Nevertheless, Sinhababu and Thierauf [ST21, Theorem 4.1] circumvented the obstacles by employing the classical version of Hensel lifting (Theorem 4), which avoids division altogether.

Theorem 8 (VBP Closure) *Let $f \in \mathbb{F}[\boldsymbol{x}]$ be a n-variate polynomial of degree d computed by an ABP of size s. Then its arbitrary factor g can be computed by an ABP of size* $\mathrm{poly}(s, n, d)$.

Proof Sketch. The special case where $f = g^e$ is handled using the binomial expansion, as detailed in Equation (5), in conjunction with structural results from Lemma 7. For the general case, we follow the same sequence of preprocessing steps as before: taking a derivative to perform square-free reduction, applying a random shift to ensure that f is monic and that the factors of g and h remain coprime, and finally reducing the polynomial to a bivariate form. These steps prepare the polynomial for the application of Hensel lifting. As before, we begin the lifting process by applying a univariate factoring algorithm to $f(x, 0)$ (see Section 1.1). However, instead of using the monic variant of Hensel lifting–as in the previous setting–which involves division at each step, we employ the classical version of Hensel lifting (see Theorem 4). The process is repeated as before for k iterations to obtain polynomials g_k and h_k such that

$$f \equiv g_k \cdot h_k \mod y^{2^k},$$

where $k = O(\log d)$ suffices. Finally all that remains is to solve the linear system arising as before from

$$\tilde{g} \equiv g_k \cdot h'_k \mod y^{2^k}.$$

Since a guaranteed solution requires the linear system to be homogeneous, the polynomial $\tilde{g}$ is not monic. Sinhababu and Thierauf observed that the monic irreducible factor g can be recovered by dividing $\tilde{g}$ by its leading coefficient (see [ST21, Lemma 4.19]). It is then easy to observe that all the steps, including the preprocessing steps, can be performed efficiently on ABPs using the structural results discussed earlier. □

As for VP, it is an open question to extend the above result to fields of small characteristic.

Question 4 (Open) Is the class VBP closed under taking factors over fields of positive characteristic?

4.6 Explicit polynomials

A compelling class of polynomials was defined by Valiant [Val79] as a non-deterministic analog of VP. This class is denoted by VNP and is defined as follows.

Definition 6 (VNP) A family of polynomials $f = (f_n)$ is said to be in VNP over the field $\mathbb{F}$ if there exist functions $k, \ell, m : \mathbb{N} \to \mathbb{N}$ all polynomially bounded, and a polynomial family $g = (g_n) \in \mathsf{VP}$ with $g_n \in \mathbb{F}[x_1, \ldots, x_{k(n)}, y_1, \ldots, y_{m(n)}]$ such that for all n,

$$f_n(x_1, \ldots, x_{k(n)}) = \sum_{\boldsymbol{a} \in \{0,1\}^{m(n)}} g_{\ell(n)}(x_1, \ldots, x_{k(n)}, a_1, \ldots, a_{m(n)}).$$

The polynomial family g is often referred to as the family of *verifier* polynomials, and the variables $y_1, \ldots, y_{m(n)}$ are called *witness* variables. This is analogous to the boolean class NP which contains functions $f(x_1, \ldots, x_n)$ that can be written as a logical OR of a verifier function $g(x_1, \ldots, x_n, y_1, \ldots, y_m)$, $m = \mathrm{poly}(n)$ over all boolean assignments to the witness variables $y_1, \ldots, y_m$. The OR is replaced by a sum in VNP, making it perhaps closer to #P, the counting version of NP.

Clearly, $\mathsf{VP} \subseteq \mathsf{VNP}$. Motivated by Kaltofen's results on the factors of VP, Bürgisser [Bür00a, Conjecture 2.1] conjectured that VNP is closed under factorization as well. Chou, Kumar and Solomon [CKS19b, Theorem 2.9] showed that Bürgisser's conjecture is in fact true when the field has characteristic 0. The key ideas were based on a related but different factoring tool that we are going to discuss in the upcoming section.

Further, Bhargav, Dwivedi, and Saxena [BDS24, Theorem 1.6] proved that the factor closure holds over finite fields as well, using Hensel lifting in combination with Valiant's criterion (and its converse) for *low-degree* polynomials to be in VNP (cf. [Bür00a, Proposition 2.20]). We denote by $\langle a \rangle$ the boolean encoding of any mathematical object a.

Proposition 2 (Valiant's Criterion) *Consider a polynomial family $f = (f_n)$ with the number of variables and degree of f_n bounded by* $\mathrm{poly}(n)$. *Suppose $f_n = \sum_{\boldsymbol{e}} c_{\boldsymbol{e}} \boldsymbol{x}^{\boldsymbol{e}}$. If for every n, there exists a function $\phi_n \in \#\mathsf{P}/\mathsf{poly}$ such that given any exponent vector $\boldsymbol{e}$, we have $\phi_n(\langle \boldsymbol{e} \rangle) = \langle c_{\boldsymbol{e}} \rangle$, then $f \in \mathsf{VNP}$.*

Over finite fields, a weaker variant of Valiant's criterion suffices, wherein the coefficient function ϕ lies in $\#_p\mathsf{P}/\mathsf{poly}$ (see [Bür00a, Section 4.3]). We omit this subtlety when it is clear from the context.

Finally, it is once again important to state that VNP is closed under several standard operations, such as addition, multiplication, and composition [Val82, Section 4]. However, the closure under composition is particularly subtle. A naive substitution of variables may not work, and a more refined approach is needed [CKS19b, Claim 8.4]. In particular, an alternative characterization of VNP proves useful–namely, that VNP polynomials can be expressed as hypercube sums of polynomial-size formulas (see [MP08, Theorem 2] for a proof). We refer the reader to the full version of [BDS24] for detailed proofs.

Theorem 9 (Closure of VNP) *Let $f(\boldsymbol{x})$ be an n–variate polynomial of degree d such that*

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{a}\in\{0,1\}^m} q(\boldsymbol{x}, \boldsymbol{a}),$$

where $q(\boldsymbol{x}, \boldsymbol{y})$ can be computed by a circuit of size s. If $g(\boldsymbol{x})$ is any factor of f of degree r, then it can be written as

$$g(\boldsymbol{x}) = \sum_{\boldsymbol{e}\in\{0,1\}^{m'}} u(\boldsymbol{x}, \boldsymbol{e}),$$

where m' and $\mathrm{size}(u)$ are both bounded by $\mathrm{poly}(n, s, r, d, m)$.

Proof Sketch. Let $\mathbb{F} = \mathbb{F}_q$, where $q = p^a$ for some prime $p < \deg(f)$. Without loss of generality, we can assume $f = g^e \cdot h$, where $e \geq 1$. Recall that in earlier proofs, the special case $h = 1$ was handled separately. For large fields, an identity of the form Equation (5) works in VNP as well. However, over finite fields, extra care is needed.

Let $e = p^i \cdot \hat{e}$. If $\hat{e} > 1$, then we define the polynomial

$$\hat{f} = z^{\hat{e}} - f = (z - g_1^{\hat{e}}) \cdot (z^{\hat{e}-1} + z^{\hat{e}-2} g_1 + \cdots + g_1^{\hat{e}-1}),$$

where $g_1 \coloneqq g^{p^i}$. It is easy to see that the two factors of $\hat{f}$ are coprime if p does not divide $\hat{e}$. Then the monic Hensel lifting template can be applied to show that $(z - g_1^{\hat{e}})$ is in VNP. The high-level idea is that, after all the required transformations, Hensel lifting performed on $\hat{f}$ yields a small circuit computing the factor (see [BDS24, Lemma 5.9]). Since VNP is closed under composition, the required factor $(z - g_1^{\hat{e}})$, and hence $g_1^{\hat{e}}$, belongs to VNP.

If g and h are coprime, then we can directly apply the Hensel lifting approach to compute g^e in VNP, and then proceed as before.

Therefore, the only remaining case is when $\hat{e} = 1$ and $f = g^{p^i}$. Note that this case requires separate attention only over small characteristic fields. Since the

characteristic of the field is p, we can associate the coefficients of f to those of g via the standard Frobenius map. In particular, if the coefficient of $\boldsymbol{x}^{\boldsymbol{a}}$ in g is c_g, and the coefficient of $\boldsymbol{x}^{p^i \cdot \boldsymbol{a}}$ in f is c_f, then $c_g = (c_f)^{1/p^i}$.

In [BDS24], the authors observed that the *converse* of Valiant's criterion holds over finite fields. In particular, all the coefficients of f can be computed by a $\#\mathsf{P}/\mathsf{poly}$ function. Together with the Frobenius correspondence between the coefficients, this yields a $\#\mathsf{P}/\mathsf{poly}$ function for computing the coefficients of the factor g. Then, using Proposition 2, we can conclude that g can be written as a hypercube sum of small sized circuits. □

Theorem 9 also partially answers Question 3 and Question 4, proving that factors of polynomials in VP (and VBP) are in VNP over finite fields. Robert Andrews communicated to us an (as yet) unpublished proof (based on the ideas of Malod and Portier [MP08] and Andrews [And20]) that extends the closure of VNP to all *perfect fields*. A field $\mathbb{F}$ is perfect if either it is of characteristic zero, or, if the characteristic is a prime p, then all elements of $\mathbb{F}$ are p-th powers. All finite fields are perfect. Thus, it only remains to settle the closure of VNP over infinite non-perfect fields (e.g. the fraction field $\mathbb{F}_p(t)$ for some indeterminate t).

Question 5 (Open) Is the class VNP closed under taking factors over infinite non-perfect fields of positive characteristic?

5 Factoring via Newton iteration

In 1669, Isaac Newton described a method to approximate roots of real valued functions. Also known as the Newton-Raphson method, the idea is to start with an initial approximate root x_0 of the equation $f(x) = 0$ and successively produce better approximations using the rule

$$x_{i+1} \;=\; x_i - \frac{f(x_i)}{f'(x_i)}\,,$$

provided $f'(x_i) \neq 0$. The value x_{i+1} is the x-intercept of the tangent to the curve of $f(x)$ at the point x_i. Alternately, we can think of this as finding a root of $f(x_i) + f'(x_i)(x - x_i)$, the Taylor approximation of $f(x)$ (up to linear terms) at the point x_i. It is intuitive that x_{i+1} is closer than x_i to a root of f. Similar to the notion of p-adic numbers (Section 4.3), two multivariate polynomials (or even power series) $p(\boldsymbol{x})$

and $q(\boldsymbol{x})$ are considered *close* up to degree t if they only differ in terms of degree greater than t, i.e., $p(\boldsymbol{x}) = q(\boldsymbol{x}) \mod \langle \boldsymbol{x} \rangle^{t+1}$.

We first state a slower version of Newton's method over the power series ring that is useful in factoring applications.

Lemma 8 ([CKS19b, Lemma 5.1]) *Let $f(\boldsymbol{x}, y) \in \mathbb{F}(\boldsymbol{x})[y]$ be a polynomial over $\mathbb{F}(\boldsymbol{x})$ and $\mu \in \mathbb{F}$ be such that $f(\mathbf{0}, \mu) = 0$ but $\delta := \partial_y f(\mathbf{0}, \mu) \neq 0$.*

Then, there is a unique *power series $\varphi \in \mathbb{F}[[\boldsymbol{x}]]$ with $\varphi(\mathbf{0}) = \mu$ such that $f(\boldsymbol{x}, \varphi) = 0$. Set $\varphi_0 = \varphi(\mathbf{0})$, and for all $t \geq 0$, define*

$$\varphi_{t+1} \;:=\; \varphi_t - \frac{f(\boldsymbol{x}, \varphi_t)}{\delta}.$$

The rate of convergence of the sequence of polynomials *$\varphi_t \in \mathbb{F}[\boldsymbol{x}]$ to the root φ is* linear*:*

$$\varphi = \varphi_t \mod \langle \boldsymbol{x} \rangle^{t+1} \text{ for all } t \geq 0.$$

Consider the case when a *polynomial* $f(\boldsymbol{x}, y) \in \mathbb{F}[\boldsymbol{x}, y]$ has a degree-r *polynomial* $g(\boldsymbol{x}) \in \mathbb{F}[\boldsymbol{x}]$ as a root (w.r.t. y). After r steps of Newton Iteration, we get a polynomial φ_r (of degree possibly greater than r) that agrees with g on all terms up to degree r. We obtain g by truncating φ_r up to terms of degree at most r. Seen differently, Newton Iteration lets us find a *linear* factor $(y - g(\boldsymbol{x}))$ of $f(\boldsymbol{x}, y)$ by iteratively approximating the root $g(\boldsymbol{x})$, provided the root is of multiplicity one, also called a *simple* root. If the root is not simple, i.e. $y - g(\boldsymbol{x})$ occurs with multiplicity $e > 1$ in f, the derivative at the root $\partial_y f(\boldsymbol{x}, g(\boldsymbol{x})) = 0$, so we consider instead the $(e-1)$–th partial derivative $\partial_{y^{e-1}} f(\boldsymbol{x}, y)$, which has $g(\boldsymbol{x})$ as a simple root.

A 'random' invertible shift of the variables $\boldsymbol{x} \mapsto \boldsymbol{x} + \alpha y + \beta$ where α and β are random elements from $\mathbb{F}^n$ ensures that f is monic in y and thus, so are its factors (see, for e.g., [DSS22, Lemma 14]). However, in general, the factors of $f(\boldsymbol{x}, y)$ may not be linear. Nevertheless, over the algebraically closed field $\overline{\mathbb{F}(\boldsymbol{x})}$, f uniquely factors as $\prod_i (y - \varphi_i(\boldsymbol{x}))^{e_i}$ where for all i, $e_i > 0$ and $\varphi_i(\boldsymbol{x}) \in \overline{\mathbb{F}(\boldsymbol{x})}$. In fact, it can be shown that after the variable shift, f splits over the power series ring i.e., $\varphi_i(\boldsymbol{x}) \in \mathbb{F}[[\boldsymbol{x}]]$ for all i (see [DSS22, Theorem 17] for a proof). Moreover, the constant terms of the roots $\mu_i := \varphi_i(\mathbf{0})$ are all *distinct* non-zero field elements. Using Newton Iteration, we can approximate these power series roots. A faster version with a *quadratic* rate of convergence is sometimes more useful.

Lemma 9 ([DSS22, Lemma 15]) *Let $f(\boldsymbol{x}, y) \in \mathbb{F}(\boldsymbol{x})[y]$ be a polynomial over $\mathbb{F}(\boldsymbol{x})$ and $\mu \in \mathbb{F}$ be such that $f(\mathbf{0}, \mu) = 0$ but $\delta := \partial_y f(\mathbf{0}, \mu) \neq 0$.*

Then, there is a unique *power series* $\varphi \in \mathbb{F}[[\boldsymbol{x}]]$ *with* $\varphi(\mathbf{0}) = \mu$ *such that* $f(\boldsymbol{x}, \varphi) = 0$. *Set* $\varphi_0 = \varphi(\mathbf{0})$, *and for all* $t \geq 0$ *define*

$$\varphi_{t+1} \coloneqq \varphi_t - \frac{f(\boldsymbol{x}, \varphi_t)}{\partial_y f(\boldsymbol{x}, \varphi_t)}.$$

Since $\varphi_t(\mathbf{0}) = \mu$ *for all* $t \geq 0$ *and since* $\delta \neq 0$, *the denominator above is invertible in the power series ring. The sequence of* rational functions $\varphi_t(\boldsymbol{x}) \in \mathbb{F}(\boldsymbol{x})$ *converges to* φ *at a* quadratic *rate:*

$$\varphi = \varphi_t \mod \langle \boldsymbol{x} \rangle^{2^t} \text{ for all } t \geq 0.$$

The above Newton-Iteration lemmas can also be viewed as power series versions of the *Implicit Function Theorem* (see, for e.g. [KP13, Chapter 1] and [Art22, Theorem 9.2.1]). Returning to the problem of factoring $f(\boldsymbol{x}, y)$, suppose it has a non-trivial factor $g(\boldsymbol{x}, y)$ of degree r. As discussed earlier, we can assume g is a factor of multiplicity one by considering appropriate partial derivatives. Since f is monic in y, so is g, and splits as $\prod_{i=1}^{r}(y - \varphi_i)$ for some power series roots $\varphi_i(\boldsymbol{x}) \in \mathbb{F}[[\boldsymbol{x}]]$. Crucially, these φ_i's are a subset of the roots of f. Given this set (via the *distinct* degree-0 terms of the φ_i's), we can approximate each of these power series roots up to degree r using Newton Iteration on f. Whether we use the faster or slower version will depend on the convenience afforded by the computational model. Finally, in order to obtain g, it is enough to multiply the approximations, truncate the result up to terms of degree r, and invert the variable shift.

5.1 Low depth circuits

In this section, we consider factors of polynomials computable by small-size circuits that are also of *bounded-depth*. These circuits form an extremely interesting subclass of arithmetic circuits since general circuits can be "efficiently" depth-reduced, unlike Boolean circuits. An influential body of work [Val+83; AV08; Koi12; Tav15; Gup+16] shows that any circuit of size s computing a polynomial of degree d can be equivalently written as a *homogeneous* depth-4 circuit of size $s^{O(\sqrt{d})}$, or even a depth-3 circuit of that size, if one foregoes homogeneity.

Dvir, Shpilka, and Yehudayoff [DSY09] studied factors of low-depth circuits in order to extend the Hardness-Randomness paradigm of Kabanets and Impagliazzo [KI04] (see Section 3.2) to this model. Of particular interest to them were

factors of the form $y - g(\boldsymbol{x})$ of a bounded-depth circuit $f(\boldsymbol{x}, y)$. They showed that if the individual degree of f with respect to y is bounded, then the root $g(\boldsymbol{x})$ has a small bounded-depth circuit.

Theorem 10 ([DSY09, Theorem 4]) *Let $f \in \mathbb{F}[\boldsymbol{x}, y]$ be a $(n+1)$–variate polynomial with $\deg(f) \leq d$ and $\deg_y(f) \leq k$, computed by a depth-Δ circuit of size s. Then, its factors of the form $y - g(\boldsymbol{x})$ with $\deg(g) = r$ can be computed by circuits of size $\mathrm{poly}(s, r^k, d, n)$ and depth $\Delta + O(1)$.*

Proof Sketch. The obvious approach would be to use Lemma 8 to successively approximate the root $g(\boldsymbol{x})$ via Newton Iteration. Let us assume the preconditions for using the lemma are met. The crucial observation [DSY09, Lemma 3.1] is that the t–th approximate φ_t can be expressed as a $(k+1)$–variate, degree–t polynomial Q_t in the coefficients of f, i.e., $\varphi_t = Q_t(f_0, \ldots, f_k)$, where $f(\boldsymbol{x}, y) = \sum_{j=0}^{k} f_j(\boldsymbol{x}) y^j$, with $f_j(\boldsymbol{x}) \in \mathbb{F}[\boldsymbol{x}]$. After r iterations, we get the polynomial $\varphi_r = Q_r(f_0, \ldots, f_k)$ of degree at most dr such that $g = \varphi_r \mod \langle \boldsymbol{x} \rangle^{r+1}$.

The sparsity of Q_r is at most $\binom{r+k}{k} = O(r^k)$. Composing the trivial depth-2 circuit for Q_r (of size $O(r^k)$) with the depth-Δ circuits for f_j's (of size $O(sk)$, obtained via interpolation on f Lemma 2), we have a circuit for φ_r of depth $\Delta + 2$ and size $\mathrm{poly}(r^k, s)$. To extract g, we can use interpolation on φ_r by first mapping $x_i \to z x_i$ and using Lemma 2 to obtain all the components of degree at most r (w.r.t z) with a size blow up of $\mathrm{poly}(r^k, s, d)$. The depth does not change.

By repeatedly taking partial derivatives of f using Lemma 3 (incurring a size blow-up of at most $\mathrm{poly}(s, k)$) and a translation of the coordinates (resulting in a depth increase by 1), we can ensure that g is a simple root of f (and thus, Lemma 8 is applicable). So we have a depth $\Delta + 3$ circuit for $y - g(x)$ of size $\mathrm{poly}(r^k, s, d, n)$. □

Note that the size bound in Theorem 10 is $\mathrm{poly}(n)$ if $s, d \leq \mathrm{poly}(n)$ and $k = O(1)$. In fact, one can also obtain a randomized polynomial time algorithm to *find* the root g. Rafael Oliveira [Oli16] generalized this result to show that polynomials of bounded individual degrees and computable by small bounded-depth circuits are closed under factorization. A little thought reveals that if a polynomial has its individual degree bounded, then so will its factors.

Theorem 11 ([Oli16, Theorem 1.2]) *Let $f \in \mathbb{F}[\boldsymbol{x}, y]$ be a $(n+1)$–variate polynomial with $\deg(f) \leq d$, and individual degree bounded by k. Suppose f can be computed by a depth-Δ circuit of size s. Then, any factor $g(\boldsymbol{x}, y)$ of degree r that divides f can be computed by a circuit of size $\mathrm{poly}(r^k, k, s, d, n)$ and depth $\Delta + O(1)$.*

Proof Sketch. The main idea is to use the argument at the beginning of Section 5 and essentially lift the above result of Dvir Shpilka and Yehudayoff to all factors. Let us begin by writing $f(\boldsymbol{x}, y) = \sum_{j=0}^{k} f_j(\boldsymbol{x}) y^j$ and $g(\boldsymbol{x}, y) = \sum_{j=0}^{\ell} g_j(\boldsymbol{x}) y^j$, where $\ell \leq k$. A shift of the variables $\boldsymbol{x} \mapsto \boldsymbol{x} + \alpha y + \beta$ to ensure that f is monic in y, however, would make its individual degree kn. Therefore, in order to reduce to the monic case, we rewrite

$$f(\boldsymbol{x}, y) \;=\; f_k(\boldsymbol{x}) \left(y^k + \sum_{j=0}^{k-1} f_j(\boldsymbol{x}) / f_k(\boldsymbol{x}) y^j \right) \text{ and}$$

$$g(\boldsymbol{x}, y) \;=\; g_\ell(\boldsymbol{x}) \left(y^\ell + \sum_{j=0}^{\ell-1} g_j(\boldsymbol{x}) / g_\ell(\boldsymbol{x}) y^j \right).$$

The idea is that, since g divides f, the leading term g_ℓ divides f_k. If we are able to recover the factor g_ℓ from f_k by an induction on the number of variables (note the one fewer variable), and we somehow obtain a circuit for g/g_ℓ using Newton Iteration (Section 5) for monic polynomial factoring, then multiplying the circuits for g_ℓ and g/g_ℓ would give us g.

Although we can extract f_k from f via interpolation, we cannot afford the resulting size blow-up (by a factor of k) in the induction argument. To avoid this, we work with the polynomial $\tilde{f}(\boldsymbol{x}, y) \coloneqq y^k f(\boldsymbol{x}, 1/y)$, which is the *reversal* of (the coefficients of) the polynomial f. The leading coefficient of $\tilde{f}$ is now $f_0(\boldsymbol{x}) = f(\boldsymbol{x}, 0)$ which has a smaller circuit than f. Moreover, the factors of the reversal of a polynomial are just the reversals of the original factors.

Once we find (by induction) a circuit of depth $\Delta + O(1)$ and size $\mathrm{poly}(r^k, s, d)$ for $g_0(\boldsymbol{x})$ (the leading coefficient of $\tilde{g}(\boldsymbol{x}, y)$), we can use Theorem 10 to approximate the roots $\varphi_i(\boldsymbol{x}) \in \mathbb{F}[[\boldsymbol{x}]]$ (up to degree d) of the monic rational function $\tilde{g}/g_0 = \prod_{i=1}^{\ell} (y - \varphi_i)$. Using Newton Iteration on $\tilde{f}$, whose degree in y is bounded by k, we can approximate these roots by circuits of depth $\Delta + O(1)$ and size $\mathrm{poly}(r^k, s, d)$. We further obtain a circuit of size $\mathrm{poly}(r^k, s, k, d)$ for $\tilde{g}/g_0$ by multiplying the approximations for the circuits and truncating using interpolation, resulting in a depth increase by at most a constant. Finally, we get $g(\boldsymbol{x}, y)$ by multiplying g_0 and $\tilde{g}/g_0$ and computing the reversal of $\tilde{g}$. □

The above result can be used to obtain a *randomized* algorithm running in time $\mathrm{poly}(s, (nk)^k)$ to compute all the factors of the polynomial f with individual degree bounded by k.

Chou, Kumar, and Solomon [CKS19b] removed the restriction on individual degrees at the expense of requiring that the total degree of the factor f be small.

Theorem 12 ([CKS19b, Theorem 2.1]) *Let $f \in \mathbb{F}[\boldsymbol{x}, y]$ be a $(n+1)$–variate polynomial of degree d. Suppose f can be computed by a depth-Δ circuit of size s. Then, any factor $g(\boldsymbol{x}, y)$ of degree r that divides f can be computed by a circuit of size $\mathrm{poly}(r^{\sqrt{r}}, n, d, s)$ and depth $\Delta + O(1)$.*

Proof Sketch. The broad outline is still the reduction of factoring to approximating roots, same as in the proof of Theorem 11. Consider the simple case when g is a root of f. If $\deg_y(f) \geq n$, the earlier observation by Dvir et.al that there exists a $(\deg_y(f)+1)$–variate polynomial φ_r that approximates g up to degree r becomes trivial since g is already an n–variate polynomial.

However, irrespective of $\deg_y(f)$, Chou, Kumar and Solomon show that φ_r can in fact be written as a $(r+1)$–variate, $\mathrm{poly}(r)$–sized, degree–r polynomial A_r in the (at most) r–th order partial derivatives of f [CKS19b, Lemma 5.3][3]. Using the depth-reduction results mentioned at the beginning of this section, we can further squash this to a depth–3 circuit for A_r of size $r^{O(\sqrt{r})}$. Composition with the depth–Δ, $\mathrm{poly}(s, d, r)$–sized circuits for the partial derivatives results in a depth–$(\Delta+3)$ circuit for φ_r, from which we can extract g using interpolation, as before.

The rest of the argument to lift this to a general factor remains almost the same as in Theorem 11. The argument does not need special handling of the leading coefficient as we do not need to worry about maintaining individual degree after the coordinate shift. □

Hence, if the degree r of the factor is at most $\log^2 n / \log\log n$, then the size bound for g above is $\mathrm{poly}(n)$. When r is large, it still leaves open the following question of Shpilka and Yehudayoff [SY10, Open Problem 19]:

Question 6 (Open) If a polynomial $f(\boldsymbol{x}, y)$ can be computed by a depth-Δ circuit of size s, can any factor $g(\boldsymbol{x}, y)$ be computed by a circuit of depth $O(\Delta)$ and size $\mathrm{poly}(s)$?

Recently, there have been a series of works on derandomizing factorization for constant depth circuits [KRS24; Kum+24; DST24]. There has also been progress on computing GCD, resultants, and various other linear algebra problems in constant

[3] A proof of closure of VNP under factoring also follows from here using closure properties of VNP under coefficient extraction and composition (see [CKS19b, Claim 8.3 and Claim 8.4]).

depth [AW24]. Building on this, Bhattacharjee, Kumar, Ramanathan, Saptharishi and Saraf [Bha+25] gave the first subexponential deterministic algorithm for factorization of constant depth circuits.

5.2 High degree circuits

The families of polynomials we considered till now had 'low' degree, bounded by a polynomial in the number of variables. Malod [Mal03; Mal07] considered the case when the degrees are not bounded.

Definition 7 A family of polynomials $f = (f_n)$ is said to be in VP_{nb} over the field $\mathbb{F}$ if the number of variables and the size of the smallest circuit over $\mathbb{F}$ are bounded by $\mathrm{poly}(n)$.

Note that the degree of a polynomial family in VP_{nb} can be exponential in the size of the circuit. Kaltofen [Kal87] showed (Theorem 7) that for an n-variate polynomial $f = g^e h$ where g and h are coprime and g has multiplicity e,

$$\mathrm{size}(g) = \mathrm{poly}(\mathrm{size}(f), \deg(g), e, n) \tag{6}$$

Hence, the best size upper bound one can deduce for (exponential–degree) factors of polynomials in VP_{nb} from Kaltofen's result is exponential. This is unavoidable in general: the polynomial $x^{2^n} - 1 = \prod_{i=1}^{2^n}(x - \xi^i)$ where ξ is a 2^n–th root of unity, can be computed by a circuit of size $O(n)$. Lipton and Stockmeyer [LS78] showed that a random exponential–degree factor $\prod_{i\in S}(x - \omega^i)$ where $S \subset [2^n]$ and $|S| = \exp(n)$ requires $\exp(n)$–size circuits. So VP_{nb} is *not* closed under taking factors. Nevertheless, Bürgisser's *Factor Conjecture* [Bür00a, Conj. 8.3] states that the size of the factor g in Equation (6) should be independent of its multiplicity e. In particular, $\mathrm{poly}(n)$–degree factors of a polynomial family in VP_{nb} should be in VP.

Question 7 (Open) Show that if $g(\boldsymbol{x})$ is a factor of an n–variate polynomial $f(\boldsymbol{x})$ then,

$$\mathrm{size}(g) = \mathrm{poly}(\mathrm{size}(f), \deg(g), n).$$

See [Bür04, Section 4] for some applications of the above conjecture to decision complexity. Over the years, there has been partial progress on the problem. In the case when f is a power of g, Kaltofen [Kal87] (cf. [Bür04, Proposition 6.1]) already showed the conjecture to be true.

Theorem 13 ([Kal87, Theorem 2]) *Suppose $f(\boldsymbol{x}) = g(\boldsymbol{x})^e \in \mathbb{F}[\boldsymbol{x}]$ is an n-variate polynomial. Then,*

$$\text{size}(g) = \text{poly}(\text{size}(f), \deg(g), n).$$

Proof Sketch. We cannot directly approximate the roots of f (and hence g) as outlined in Section 5 since we cannot reduce to the multiplicity one case by taking $(e-1)$ partial derivatives – the size blow up will be $\text{poly}(\text{size}(f), e)$. Instead, we find the root of the equation $y^e - f(\boldsymbol{x})$ by Newton Iteration (Lemma 9).

By shifting variables we can ensure that all the $O(\log(\deg(g)))$ steps can be performed. Crucially, we only need the homogeneous components of f up to $\deg(g)$ for the intermediate steps. Straightforward analysis will show that $\text{size}(g) = \text{poly}(\text{size}(f), \deg(g), n, \log e)$, whereby noting that $e \leq \exp(\text{size}(f))$ gives the result. □

As another special case, Dutta, Saxena and Sinhababu [DSS22] showed that the conjecture also holds if the *squarefree* part of f is of low degree. Suppose $f = \prod_{i=1}^{m} f_i^{e_i}$ is the complete factorization of f with f_i's irreducible. Recall that the squarefree part of f (also called the *radical*) is the polynomial $\text{rad}(f) \coloneqq \prod_{i=1}^{m} f_i$.

Theorem 14 *Let $f(\boldsymbol{x}, y)$ be an n–variate polynomial and let $g(\boldsymbol{x}, y)$ be a factor of f. We then have*

$$\text{size}(g) = \text{poly}(\text{size}(f), \deg(\text{rad}(f)), n).$$

Proof Sketch. We follow the exposition of Sinhababu [Sin19]. Consider the derivative

$$\partial_y f(\boldsymbol{x}, y) = \partial_y \left(\prod_{i=1}^{m} f_i^{e_i} \right) = \sum_{i=1}^{m} e_i f_i^{e_i - 1} \partial_y f_i \left(\prod_{j \neq i} f_j^{e_j} \right).$$

Rewrite it as $\partial_y f = \prod_{i=1}^{m} f_i^{e_i - 1} u$ with $u = \sum_{i=1}^{m} e_i \partial_y f_i \left(\prod_{j \neq i} f_j \right)$. The auxiliary polynomial $F \coloneqq f + z\partial_y f$ factors as $\prod_{i=1}^{m} f_i^{e_i - 1} \left(\prod_{i=1}^{m} f_i + z \cdot u \right)$, where z is a fresh variable. The factor $G \coloneqq \text{rad}(f) + z \cdot u$ is coprime to $\prod_{i=1}^{m} f_i^{e_i - 1}$, has the same degree as $\text{rad}(f)$ and is of multiplicity one. Using Kaltofen's result (Theorem 7), we get $\text{size}(G) = \text{poly}(\text{size}(f), \deg(\text{rad}(f)), n)$. One can obtain $\text{rad}(f)$ by setting $z = 0$ in G, and any irreducible factor f_i in size $\text{poly}(\text{size}(f), \deg(\text{rad}(f)), n)$ by further factoring $\text{rad}(f)$. The result follows by suitably combining powers of f_i (via repeated squaring) to form g. □

Shortly after proposing the factor conjecture, Bürgisser [Bür04] showed its plausibility: low-degree factors can be *approximated* by small circuits!

5.3 Algebraic approximation

A natural notion of approximation for algebraic computation was defined by Bürgisser [Bür04]. A polynomial $f \in \mathbb{F}[\boldsymbol{x}]$ is approximated by a polynomial $F \in \mathbb{F}[\varepsilon][\boldsymbol{x}]$ to an *order of approximation* M if

$$F(\boldsymbol{x}, \varepsilon) = \varepsilon^M f(\boldsymbol{x}) + \varepsilon^{M+1} Q(\boldsymbol{x}, \varepsilon), \tag{7}$$

for some polynomial $Q(\boldsymbol{x}, \varepsilon) \in \mathbb{F}[\boldsymbol{x}, \varepsilon]$. The approximate/border size of f, denoted $\overline{\text{size}}(f)$, is defined as the size of the smallest circuit *over the ring of constants* $\mathbb{F}[\varepsilon]$ computing a polynomial $F \in \mathbb{F}[\varepsilon][\boldsymbol{x}]$ that approximates f.

Over fields like $\mathbb{R}$ or $\mathbb{C}$, one can think of the above as an approximation in the sense $\lim_{\varepsilon \to 0} \varepsilon^{-M} F = f$. Another way of formulating the notion of approximation is to consider $F_\varepsilon \in \mathbb{F}(\varepsilon)[\boldsymbol{x}]$ over the rational function field $\mathbb{F}(\varepsilon)$ instead, and require an approximation of the form

$$F_\varepsilon(\boldsymbol{x}) = f(\boldsymbol{x}) + \varepsilon Q_\varepsilon(\boldsymbol{x}),$$

where $Q_\varepsilon \in \mathbb{F}[\varepsilon][\boldsymbol{x}]$ is a polynomial. The border complexity of f is defined as before, but with the circuit size now calculated over $\mathbb{F}(\varepsilon)$. In this case, although $F_{\varepsilon=0} = f$ is defined at $\varepsilon = 0$, the intermediate computations in the circuit for F (over $\mathbb{F}(\varepsilon)$) might not be. Scaling arguments show that these two notions of approximation are equivalent [Bür04, Lemma 5.6]. For a detailed discussion on other natural definitions of approximation (both topological and algebraic), and their equivalence, we point the reader to [Bür04, Section 5] and [BIZ18, Section 2]. The notion of border complexity naturally suggests an approximation version of the class VP.

Definition 8 ($\overline{\mathsf{VP}}$) A polynomial family $f = (f_n)$ is in the class $\overline{\mathsf{VP}}$ if the number of variables of f_n, its degree, and the size of the smallest circuit over $\mathbb{F}[\varepsilon]$ *approximating* f_n are bounded by $\text{poly}(n)$.

It is clear that $\mathsf{VP} \subseteq \overline{\mathsf{VP}}$. In an attempt to utilize sophisticated tools from algebraic geometry and representation theory to study Valiant's conjecture, Mulmuley and Sohoni [MS01; MS08] proposed $\overline{\mathsf{VP}} \not\subseteq \mathsf{VNP}$ as the mathematically nicer (but possibly harder) conjecture. Further details on the *Geometric Complexity Theory* (GCT) program for proving lower bounds can be found in [Reg02; Mul11; Bür+11; Mul12; Gro12; Lan17; BI25].

Returning to the factorization problem, Bürgisser showed that $\text{poly}(n)$-degree factors of families in VP_{nb} are in $\overline{\mathsf{VP}}$.

Theorem 15 ([Bür04, Theorem 1.3]) *Let $\mathbb{F}$ be a field of characteristic 0 and let $f(\boldsymbol{x}, y)$ be an $(n+1)$–variate polynomial over $\mathbb{F}$. Suppose $g(\boldsymbol{x}, y)$ is a factor of f. Then,*

$$\overline{\text{size}}(g) = \text{poly}(\text{size}(f), \deg(g), n).$$

Proof Sketch. It suffices to show the result for irreducible g. Let $f = g^e h$ where g and h are coprime. As earlier, we would like to approximate the roots of g. Due to the possibly exponential multiplicity e, we cannot reduce to the case of a simple root by taking derivatives of f. We can, however, find a point p (equal to $(\mathbf{0}, 0)$ after shifting coordinates), such that g vanishes at p but both h and $\partial_y g$ do not [Bür04, Lemma 3.3]. We can then try to build the corresponding (unique) power series root $\varphi \in \mathbb{F}[[\boldsymbol{x}]]$ of g (which it shares with f) using Newton Iteration (Lemma 9) on f. This seems quite straightforward, except that $\partial_y f = e g^{e-1} h \partial_y g + g^e \partial_y h$ always vanishes at p when $e > 1$. The main idea is to consider the *perturbed* polynomial

$$F_\varepsilon(\boldsymbol{x}, y) \coloneqq f(\boldsymbol{x}, y + \varepsilon) - f(\mathbf{0}, \varepsilon).$$

Note that $F_{\varepsilon=0} = f$ and $F_\varepsilon(\mathbf{0}, 0) = 0$. We encourage the reader to check that the derivative $\partial_y F_\varepsilon(\mathbf{0}, 0)$ does not vanish anymore, and hence we can use Lemma 9 to construct the root ψ_ε of F_ε, which in fact is an algebraic approximation of the root φ of f (and also g), i.e. $\psi_{\varepsilon=0} = \varphi$ (see [Bür04, Proposition 3.4]). This way, we can efficiently *approximate* the factor g. □

The approximate circuit constructed for the low-degree factor in Theorem 15 has additional structure. Note that a priori, the order of approximation M for a polynomial f can be arbitrarily large. However, Bürgisser [Bür04; Bür20, Theorem 5.7] showed that over algebraically closed fields, $M = \exp(\overline{\text{size}}(f))$. Therefore, the free constants from $\mathbb{F}[\varepsilon]$ used for approximation can be assumed to be 'only' exponential in degree and hence, size. Since the circuit for the factor g above is constructed via Newton Iteration, the constants in $\mathbb{F}[\varepsilon]$ are in fact circuits of size $\text{poly}(\text{size}(f))$ (but exponential in degree). Hence, imposing this restriction on the size of the univariate polynomials in ε is quite natural.

Definition 9 ($\overline{\mathsf{VP}}_\varepsilon$) A polynomial family $f = (f_n)$ over the field $\mathbb{F}$ is in the class $\overline{\mathsf{VP}}_\varepsilon$ if the number of variables of f_n, its degree, and the size of the smallest circuit *over* $\mathbb{F}$ approximating f_n are bounded by $\text{poly}(n)$.

Note that the circuit size of the approximating polynomial is over the base field $\mathbb{F}$, thus also incorporating the size of the constants in $\mathbb{F}[\varepsilon]$. Over finite fields, the present authors showed that $\overline{\mathsf{VP}}_\varepsilon$ is in fact in VNP, something that is not known about $\overline{\mathsf{VP}}$.

Theorem 16 ([BDS24, Theorem 1]) *Let $\mathbb{F}$ be a finite field, and $f \in \mathbb{F}[\boldsymbol{x}]$ be an n–variate polynomial of degree d such that the size of the smallest circuit (over $\mathbb{F}$) approximating f is of size s. Then, f can be written as*

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{a} \in \{0,1\}^m} g(\boldsymbol{x}, \boldsymbol{a}),$$

where m, and $\mathrm{size}_{\mathbb{F}}(g(\boldsymbol{x}, \boldsymbol{y}))$ are bounded by $\mathrm{poly}(s, n, d)$.

Proof Sketch. The main idea is to use Valiant's criterion (Proposition 2) to show that the polynomial $f = \sum_{\boldsymbol{e}} c_{\boldsymbol{e}} \boldsymbol{x}^{\boldsymbol{e}}$ has coefficients $c_{\boldsymbol{e}}$ that are "easy to describe". Notice that the polynomial F approximating f has a small circuit but exponential degree, where the high degree is only due to ε.

In the approximating expression for f (Equation (7)), we interpolate F on *all* the variables (including ε) using appropriate powers of unity. Consequently, each coefficient $c_{\boldsymbol{e}}$ of f can be written as a hypercube sum over a small-size high-degree circuit. When $\mathbb{F}$ is a finite field, we can *simulate* this algebraic circuit using a small boolean circuit. As a result, we further obtain that the coefficient function of f is in $\#\mathsf{P}/\mathrm{poly}$, whence we can apply Valiant's criterion. □

As a corollary of Theorem 16, and the observation that the approximate circuit in Theorem 15 is presentable, we can conclude that the families of low-degree factors of high-degree circuits are in VNP.

Corollary 1 *Let $f \in \mathbb{F}_q[\boldsymbol{x}]$ be an n–variate polynomial over a finite field $\mathbb{F}_q$ of characteristic p. Suppose g is a $\mathrm{poly}(n)$–degree irreducible factor of f with multiplicity coprime to the characteristic p. If $\mathrm{size}(f) = s$, then g can be written as*

$$g(\boldsymbol{x}) = \sum_{\boldsymbol{a} \in \{0,1\}^m} u(\boldsymbol{x}, \boldsymbol{a}),$$

where m, and $\mathrm{size}(u(\boldsymbol{x}, \boldsymbol{y}))$ are bounded by $\mathrm{poly}(s, n, d)$.

It is an open problem to extend these results to all fields.

Question 8 (Open) Can we show that $\overline{\mathsf{VP}}_\varepsilon \subseteq \mathsf{VNP}$ over *any* field $\mathbb{F}$?

6 To Hensel lift or Newton iterate?

Hensel Lifting (Section 4) and Newton Iteration (Section 5), though different when viewed from afar, are quite similar upon closer inspection.

Let $f(\boldsymbol{x}, y)$ be a monic (with respect to y) polynomial of degree d with a root $g(\boldsymbol{x})$ (with respect to y) of multiplicity one, i.e. $f(\boldsymbol{x}, g(\boldsymbol{x})) = 0$, but $\partial_y f(\boldsymbol{x}, g(\boldsymbol{x})) \neq 0$. When we wanted to find iteratively better approximations to g, we started with an approximation g_0 that was consistent with g modulo the ideal $\mathcal{I} \coloneqq \langle \boldsymbol{x} \rangle$ and used Newton's method to obtain a better approximation g_1 that was consistent with g modulo $\mathcal{I}^2 = \langle \boldsymbol{x} \rangle^2$. The polynomial g being a root of f implies that f can be factored as $f(\boldsymbol{x}, y) = (y - g(\boldsymbol{x}))h(\boldsymbol{x}, y)$ for some polynomial h. So, we can instead view our Newton Iteration process as a Hensel lift of the factorization $f = (y - g_0)h_0 \mod \mathcal{I}$ to the factorization $f = (y - g_1)h_1 \mod \mathcal{I}^2$. Since g was a root of multiplicity one, the factors $(y - g(\boldsymbol{x}))$ and $h(\boldsymbol{x}, y)$ were coprime. In this sense, Newton Iteration is a special case of Hensel Lifting.

Suppose now that we have the factorization $f(\boldsymbol{x}, y) = g(\boldsymbol{x}, y)h(\boldsymbol{x}, y)$. Let us write $f = \sum_{i=0}^{d} f_i y^i$, and similarly $g = \sum_{i=0}^{d_1} g_i y^i$, $h = \sum_{i=0}^{d_2} h_i y^i$ where for all i, $f_i, g_i, h_i \in \mathbb{F}[\boldsymbol{x}]$ and $d_1 + d_2 = d$. Noting that $f_d = 1$, we can view this factorization as a system of equations:

$$\begin{aligned} f_0 &= g_0 h_0 \\ f_1 &= g_0 h_1 + g_1 h_0 \\ &\vdots \\ f_{d-1} &= g_{d_1-1} h_{d_2} + g_{d_1} h_{d_2-1} = g_{d_1-1} + h_{d_2-1}\,. \end{aligned}$$

For a new set of d variables $\{\boldsymbol{u}, \boldsymbol{w}\} = u_0, \ldots, u_{d_1-1}, w_0, \ldots, w_{d_2-1}$, consider the d equations

$$\begin{aligned} \varphi_0 &\coloneqq f_0 - u_0 w_0 \\ \varphi_1 &\coloneqq f_1 - u_0 w_1 - u_1 w_0 \\ &\vdots \\ \varphi_{d-1} &\coloneqq f_{d-1} - u_{d_1-1} - w_{d_2-1}. \end{aligned}$$

The coefficients $(g_0, \ldots, g_{d_1-1}, h_0, \ldots, h_{d_2-1})$ are a common zero of the equations $\varphi = (\varphi_0, \ldots, \varphi_{d-1})$. Given an approximate root $(\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{F}[\boldsymbol{x}]^d$ modulo the ideal $\mathcal{I}$,

a multivariate generalization of Newton Iteration gives a better approximation

$$(\boldsymbol{a}^*, \boldsymbol{b}^*) = (\boldsymbol{a}, \boldsymbol{b}) - J^{-1}\varphi(\boldsymbol{a}, \boldsymbol{b})$$

modulo the ideal $\mathcal{I}^2$, where the matrix of polynomials $J = (\partial_t \varphi_i)_{0 \le i \le d-1, t \in \{\boldsymbol{u}, \boldsymbol{w}\}}$ is the *Jacobian* of φ. We can now view a Hensel Lift of the factorization $f = g^{(0)}h^{(0)} \mod \mathcal{I}$ to $f = g^{(1)}h^{(1)} \mod \mathcal{I}^2$ as a multivariate Newton Iteration step of improving an approximate root to the above set of equations modulo $\mathcal{I}$, to modulo $\mathcal{I}^2$. The Jacobian and the Sylvester Matrix (Definition 2) are the same up to permutation of rows and columns [CKS19a, Lemma 3.2]. Hence, the invertibility of the Jacobian is equivalent to the coprimality of g and h [GG13, Exercise 15.21]. Thus, Newton Iteration generalizes Hensel Lifting. More generally over valuation rings, the two methods can be derived from one another. This is what Gathen [Gat84] has to say:

"Note that while Yun [Yun76] motivates the Hensel method as a special form of the Newton method ("Hensel meets Newton"), here the Newton method is a corollary of the Hensel method ("Hensel beats Newton")."

This folklore connection between Hensel Lifting and Newton Iteration has also appeared explicitly in a few places [Zip81; Gat84; Art22]. Chou, Kumar, and Solomon [CKS19a] used this connection to give a simplified proof of Kaltofen's VP closure result using the multivariate Newton Iteration. For more restricted models, the method that is convenient while factoring depends on the efficiency of the factoring primitive that the model affords.

7 Factoring 'weak' models

There are models weaker than circuits and branching programs. Algebraic formulas, as seen previously, are a very natural example.

7.1 Formulas

The class VF is a natural restriction of VP where the DAG underlying the circuit is a tree (Figure 2). Since a formula of size s can never compute a polynomial of degree greater than $O(s)$, we do not need to impose any additional degree restriction.

Definition 10 (VF) A family of polynomials $f = (f_n)$ is said to be in VF over the field $\mathbb{F}$ if the minimum size of the *formula* (over $\mathbb{F}$) computing f_n is bounded by a polynomial function in the number of variables n.

Remark 1 It is not hard to see that $\mathsf{VF} \subseteq \mathsf{VBP} \subseteq \mathsf{VP}$. None of the containments are known to be strict.

The factors obtained via monic Hensel lifting (Section 4.4) turn out to require polynomial divisions with remainder (essentially circuits), even if we start with a formula. We do not know how to convert circuits to formulas without a superpolynomial blow up in the formula size. The best result we know in this direction is the factor closure of superpolynomial sized formulas, first shown by Dutta, Saxena and Sinhababu [DSS22, Theorem 3]. We state here the version from the work of Chou, Kumar and Solomon [CKS19b] which gives an alternate proof of the same result using the ideas in Section 5.1.

Theorem 17 ([CKS19b, Theorem 9.1]) *Let $f(\boldsymbol{x})$ be a polynomial of degree d in n variables which can be computed by a formula of size s. If $g(\boldsymbol{x})$ is a degree-r factor of f, then it can be computed by a formula of size* $\mathrm{poly}(s, n, d, r^{O(\log r)})$.

To avoid divisions, a reasonable try would be to extend the techniques of Sinhababu and Thierauf for factoring branching programs (Section 4.5). All but the last step of solving a linear system (equivalently, computing a determinant) can indeed be done using small formulas. Unfortunately, we do not know of any $\mathrm{poly}(n)$-sized formulas for the $n \times n$ symbolic determinant. Thus the question of factor closure for formulas is still open.

Question 9 (Open) Is the class VF closed under factoring over any field $\mathbb{F}$?

Note that a negative answer to the above question would separate VF from VBP and VP, which are closed under factoring (at least when $char(\mathbb{F})$ is 0 or large).

When the depth is bounded (constant), formulas and circuits are essentially the same. Every circuit of depth Δ and size s can be equivalently written as a formula of size $O(s^{2\Delta})$ while maintaining the same depth Δ. The only results we know for formulas in this case come from Section 5.1 – bounded-depth formulas are closed under factoring either when the individual degree is bounded (Theorem 11), or the total degree is very low (Theorem 12).

7.2 Sparse polynomials

Sparse polynomials are another interesting and perhaps the simplest class that one can study. The sparsity of a polynomial f, denoted by $\|f\|$, is the number of monomials in it. Alternatively, one can think of the sparsity as the size of a $\Sigma\Pi$ circuit computing f. The work of von zur Gathen and Kaltofen [GK85], which initiated the study of factoring sparse polynomials, gave a *randomized* algorithm that outputs a factor in time polynomial in the sparsity of the factor. Naturally, it makes sense to ask if factors of sparse polynomials are sparse. Unfortunately, as they showed, this is *not* true in general. The sparsity of a factor can be superpolynomial in the sparsity of the original polynomial.

Example 1 ([GK85]) The polynomial $f = \prod_{i=1}^{n}(x_i^d - 1)$ has sparsity $\|f\| = 2^n$, but one of its factors $g = \prod_{i=1}^{n}(1 + x_i + \ldots + x_i^{d-1})$ has sparsity $\|g\| = d^n = \|f\|^{\log d}$.

If the individual degree is comparable to the number of variables, then the blow-up can be exponential.

Example 2 ([BSV20]) Over the field $\mathbb{F}_p$, the polynomial $f = \sum_{i=1}^{n} x_i^p$ has sparsity $\|f\| = n$, but the factor $g = \left(\sum_{i=1}^{n} x_i\right)^d$, for $0 < d < p$ has sparsity $\|g\| = \binom{n+d-1}{d} \approx \|f\|^d$.

One might still hope that factors of a polynomial with *bounded* individual degree are sparse.

Question 10 (Open) Let $f = g \cdot h$ be a polynomial with bounded (constant) individual degree. Then, does the following hold: $\|g\| = \text{poly}(\|f\|)$?

Tools from convex geometry have been useful in recent progress in studying this question. For a polynomial $f = \sum_{\boldsymbol{e}} c_{\boldsymbol{e}} \boldsymbol{x}^{\boldsymbol{e}}$, consider the set of exponent vectors in its support,

$$\sup(f) \coloneqq \{\boldsymbol{e} : c_{\boldsymbol{e}} \neq 0\} \subseteq \mathbb{Z}^n.$$

The convex hull of the points in the support denoted $\text{Conv}(\sup(f))$ is called the *Newton Polytope* corresponding to f:

$$P_f \coloneqq \text{Conv}(\sup(f)) = \left\{ \sum_{\boldsymbol{e}} \alpha_{\boldsymbol{e}} \boldsymbol{e} : \sum_{\boldsymbol{e}} \alpha_{\boldsymbol{e}} = 1, 0 \leq \alpha_{\boldsymbol{e}} \in \mathbb{R}, \boldsymbol{e} \in \sup(f) \right\} \subseteq \mathbb{R}^n.$$

A *vertex* of P_f is a point in the polytope that *cannot* be written as a *non-trivial* convex combination (i.e., $\alpha_{\boldsymbol{e}} < 1$ for all $\boldsymbol{e}$) of points in P_f. We will denote the vertex set of a polytope P by $V(P)$. Ostrowski [Ost99] observed that for polynomials g and h,

$$P_{gh} \;=\; P_g + P_h,$$

where the addition is a *Minkowski sum*, consisting of all points $\boldsymbol{a}+\boldsymbol{b}$, such that $\boldsymbol{a} \in P_g$ and $\boldsymbol{b} \in P_h$. It can be shown that

$$\max\{|V(P_g)|, |V(P_h)|\} \;\leq\; |V(P_g + P_h)| \;\leq\; |V(P_g)| \cdot |V(P_h)|. \tag{8}$$

The upper bound on $|V(P_g+P_h)|$ is straightforward. For the lower bound, see Bhargava et al. [BSV20, Proposition 3.2] and Appendix K in the book of Schinzel [Sch00]. This suggests a way to prove sparsity bounds. For a polynomial $f = gh$, if we can show g is 'dense' by showing a lower bound on $|V(P_g)|$, then by the above inequality, we also get a lower bound on $|V(P_g + P_h)|$, and in turn the sparsity of f. In other words, if f is sparse, so is g.

Consider the case when $f = gh$ and g is *multilinear*, i.e., the individual degree of g is at most 1. A moment's thought shows that every monomial of g corresponds to a vertex, i.e., $\sup(g) = V(P_g)$. Combining this with the lower bound in Equation (8), we get

$$\|f\| \;\geq\; |V(P_f)| \;=\; |V(P_g + P_h)| \geq |V(P_g)| \;=\; \|g\|\,. \tag{9}$$

Therefore, if $f = gh$ is a multilinear polynomial, then g is also multilinear, and we get $\|g\| \leq \|f\|$ from above, showing that sparse multilinear polynomials are closed under factoring. Shpilka and Volkovich [SV10] gave an efficient *deterministic* algorithm for factoring sparse multilinear polynomials. Volkovich [Vol15] used the sparsity bound in Equation (9) to first extend their result to sparse polynomials that split into multilinear factors. In a later work [Vol17], he proved that factors of *multiquadratic* polynomials are also sparse, and gave an efficient deterministic algorithm to factor such polynomials.

However, in general, the size of the vertex set $|V(P_g)|$ could be much smaller than the sparsity $\|g\|$ of the polynomial. The polynomial $g = \left(\sum_{i=1}^{n} x_i\right)^d$ in Example 2 has only n vertices in its Newton Polytope (corresponding to $x_1^d, \ldots, x_n^d$), but has $O(n^d)$ monomials. If the individual degree of a polynomial is bounded, Bhargava et al. [BSV20] showed that the vertex set is not too small either.

Theorem 18 *Let g be a polynomial in n variables of individual degree d. Then,*

$$|V(P_g)| \;\geq\; \|g\|^{1/O(d^2 \log n)}.$$

The bound is tight with regards to dependence on n (see [BSV20, Claim 4.4]). As an immediate corollary of the above theorem, we deduce a sparsity bound for the factors of polynomials with bounded individual degrees.

Corollary 2 *Let $f = gh$ be a polynomial in n variables of individual degree d and sparsity $\|f\| = s$. Then, the sparsity of g is $\|g\| \leq s^{O(d^2 \log n)}$.*

Proof. Note that the individual degree of g is bounded by d as well. Using Equation (8) and Theorem 18, we get

$$\|f\| \geq |V(P_f)| = |V(P_g + P_h)| \geq |V(P_g)| \geq \|g\|^{1/O(d^2 \log n)},$$

and thus, the required bound. □

Using the above sparsity bound, one can obtain a deterministic *superpolynomial* time algorithm for sparse polynomials of bounded individual degree [BSV20, Theorem 2] (also see [HG23]). We now give a brief overview of the ideas in the proof of Theorem 18.

Proof Sketch.[Theorem 18] The classical Carathéodory's theorem of convex geometry states that any point $\boldsymbol{x} \in \mathrm{Conv}(X)$ in the convex hull of a set of points $X \subseteq \mathbb{R}^n$ can be written as a convex combination of at most $n+1$ points in X. In fact, the convex combination only needs to use the vertices of $\mathrm{Conv}(X)$. We need much fewer than $n+1$ points if we are okay with *approximating* $\boldsymbol{x}$ by a convex combination.

The approximate version of Carathéodory's theorem is about a set X with bounded ℓ_∞ norm (i.e., $\max_{\boldsymbol{y} \in X} \|\boldsymbol{y}\|_\infty \leq 1$). Given an $\varepsilon > 0$, any point $\boldsymbol{x} \in \mathrm{Conv}(X)$ that lies in the convex hull of X can be ε-approximated by a point $\boldsymbol{x}'$ (i.e., $\|\boldsymbol{x} - \boldsymbol{x}'\|_\infty \leq \varepsilon$) that is a *uniform* convex combination of at most $k = O(\frac{\log n}{\varepsilon^2})$ vertex points from $V(\mathrm{Conv}(X))$. For us, X will be the scaled-down version of $\mathrm{sup}(g)$, i.e.,

$$X \;=\; \{1/d \cdot \boldsymbol{e} : \boldsymbol{e} \in \mathrm{sup}(g)\}.$$

Note that $|X| = \|g\|$ and $|V(\mathrm{Conv}(X))| = |V(P_g)|$. Moreover, for distinct $\boldsymbol{x} \neq \boldsymbol{y} \in X$, we have $\|\boldsymbol{x} - \boldsymbol{y}\|_\infty \geq 1/d$. Hence, if we choose ε to be something slightly smaller than $1/2d$, by triangle inequality, a point $\boldsymbol{x}' \in \mathrm{Conv}(X)$ can ε-approximate only one of $\boldsymbol{x}$ or $\boldsymbol{y}$, not both. So, every point $\boldsymbol{x} \in X$ is guaranteed a distinct point $\boldsymbol{x}' \in \mathrm{Conv}(X)$ that approximates it *and* is a uniform convex combination of at most k points of $V(\mathrm{Conv}(X))$. The number of such $\boldsymbol{x}'$ is at most $|V(\mathrm{Conv}(X))|^k$. Hence,

$|X| \leq |V(\mathrm{Conv}(X))|^k$, and we can choose ε sufficiently close to (but smaller than) $1/2d$ to get $k = O(d^2 \log n)$, and thus our desired bound. □

Note that the polynomials in Example 1 and Example 2 with dense factors were *symmetric*. Bisht and Saxena [BS25] answered Question 10 in the affirmative in this case.

Theorem 19 ([BS25, Lemma 4.12]) *Let $f = gh$ be a factorization with g being a* symmetric *polynomial in n variables of individual degree d and sparsity $\|f\| = s$. Then, $\|g\| \leq s^{O(d^2 \log d)}$.*

Furthermore, they used the above result to give a deterministic polynomial-time factoring algorithm [BS25, Theorem 1.2] in this case. For general sparse polynomials, the aforementioned result of Bhattacharjee, Kumar, Ramanathan, Saptharishi and Saraf [Bha+25] gives a subexponential deterministic algorithm as a special case when the depth is two.

Acknowledgements

The authors thank the conducive atmosphere of the *Workshop on Algebraic Complexity Theory 2023* in University of Warwick to initiate new ideas; and the *Workshop on Recent Trends in Computer Algebra 2023* in Institut Henri Poincaré, Paris for giving N.S. the opportunity to chalk out the details of [Sax23]. C.S.B. and N.S. thank the organizers and the participants in Ruhr University Bochum for the interesting discussions in the *8th Workshop on Algebraic Complexity Theory* (WACT 2025). We also thank Amit Sinhababu for insightful discussions on Newton Iteration and for pointing us to some relevant references.

N.S. thanks the DST-SERB agencies for funding support through the Core Research Grant (CRG/2020/000045) and the J.C. Bose National Fellowship (JCB/2022/57), as well as the N. Rama Rao Chair (2019–) of the Department of CSE, IIT Kanpur.

P.D. thanks the *Independent Research Fund Denmark* for funding support (FLows 10.46540/3103-00116B), and also acknowledges the support of Basic Algorithms Research Copenhagen (BARC) through the Villum Investigator Grant 54451.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity*. Cambridge University Press, Cambridge, 2009 (cit. on pp. 4, 9).

[Agr05] Manindra Agrawal. "Proving Lower Bounds via Pseudo-Random Generators". In: *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science*. Vol. 3821. Lecture Notes in Comput. Sci. Springer, Berlin, 2005, pp. 92–105 (cit. on p. 12).

[Agr06] Manindra Agrawal. "Determinant versus Permanent". In: *International Congress of Mathematicians. Vol. III*. Eur. Math. Soc., Zürich, 2006, pp. 985–997 (cit. on p. 9).

[AGS19] Manindra Agrawal, Sumanta Ghosh, and Nitin Saxena. "Bootstrapping Variables in Algebraic Circuits". In: *Proc. Natl. Acad. Sci. USA* 116.17 (2019), pp. 8107–8118 (cit. on p. 14).

[And20] Robert Andrews. "Algebraic Hardness versus Randomness in Low Characteristic". In: *35th Computational Complexity Conference*. Vol. 169. LIPIcs. Leibniz Int. Proc. Inform. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2020, Art. No. 37, 32 (cit. on pp. 13, 27, 33).

[Art22] Michael Artin. *Algebraic Geometry—Notes on a Course*. Vol. 222. Graduate Studies in Mathematics. American Mathematical Society, Providence, RI, 2022 (cit. on pp. 35, 45).

[AS03] Sanjeev Arora and Madhu Sudan. "Improved Low-Degree Testing and Its Applications". In: *Combinatorica* 23.3 (2003), pp. 365–426 (cit. on p. 15).

[AV08] Manindra Agrawal and V. Vinay. "Arithmetic Circuits: A Chasm at Depth Four". In: *49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society, 2008, pp. 67–75 (cit. on p. 35).

[AW24] Robert Andrews and Avi Wigderson. "Constant-Depth Arithmetic Circuits for Linear Algebra Problems". In: *2024 IEEE 65th Annual Symposium on Foundations of Computer Science—FOCS 2024*. IEEE Computer Society, 2024, pp. 2367–2386 (cit. on p. 39).

[BCS97] Peter Bürgisser, Michael Clausen, and M. Amin Shokrollahi. *Algebraic Complexity Theory*. Vol. 315. Grundlehren Der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]. Springer-Verlag, Berlin, 1997 (cit. on p. 9).

[BDS24] C. S. Bhargav, Prateek Dwivedi, and Nitin Saxena. "Learning the Coefficients: A Presentable Version of Border Complexity and Applications to Circuit Factoring". In: *STOC'24—Proceedings of the 56th Annual ACM Symposium on Theory of Computing*. ACM, New York, 2024, pp. 130–140 (cit. on pp. 31–33, 43).

[Bha+25] Somnath Bhattacharjee, Mrinal Kumar, Varun Ramanathan, Ramprasad Saptharishi, and Shubhangi Saraf. *Deterministic Factorization of Constant-Depth Algebraic Circuits in Subexponential Time*. 2025. arXiv: 2504.08063 [cs] (cit. on pp. 39, 50).

[BI25] Markus Bläser and Christian Ikenmeyer. "Introduction to Geometric Complexity Theory". In: *Theory Comput.* Graduate Surveys 10 (2025), p. 166 (cit. on p. 41).

[BIZ18] Karl Bringmann, Christian Ikenmeyer, and Jeroen Zuiddam. "On Algebraic Branching Programs of Small Width". In: *J. ACM* 65.5 (2018), Art. 32, 29 (cit. on p. 41).

[Bog05] Andrej Bogdanov. "Pseudorandom Generators for Low Degree Polynomials". In: *STOC'05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing*. ACM, New York, 2005, pp. 21–30 (cit. on p. 15).

[Bre00] Richard P. Brent. "Recent Progress and Prospects for Integer Factorisation Algorithms". In: *Computing and Combinatorics (Sydney, 2000)*. Vol. 1858. Lecture Notes in Comput. Sci. Springer, Berlin, 2000, pp. 3–22 (cit. on p. 15).

[BS25] Pranav Bisht and Nitin Saxena. "Derandomization via Symmetric Polytopes: Poly-time Factorization of Certain Sparse Polynomials". In: *ACM Trans. Comput. Theory* 17.2 (2025), 12:1–12:20 (cit. on p. 50).

[BSV20] Vishwas Bhargava, Shubhangi Saraf, and Ilya Volkovich. "Deterministic Factorization of Sparse Polynomials with Bounded Individual Degree". In: *J. ACM* 67.2 (2020), Art. 8, 28 (cit. on pp. 47–49).

[Bür+11] Peter Bürgisser, J. M. Landsberg, Laurent Manivel, and Jerzy Weyman. "An Overview of Mathematical Issues Arising in the Geometric Complexity Theory Approach to VP ≠ VNP". In: *SIAM J. Comput.* 40.4 (2011), pp. 1179–1209 (cit. on p. 41).

[Bür00a] Peter Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory*. Vol. 7. Algorithms and Computation in Mathematics. Springer-Verlag, Berlin, 2000 (cit. on pp. 9, 31, 39).

[Bür00b] Peter Bürgisser. "Cook's versus Valiant's Hypothesis". In: *Theoret. Comput. Sci.* 235 (2000), pp. 71–88 (cit. on p. 9).

[Bür04] Peter Bürgisser. "The Complexity of Factors of Multivariate Polynomials". In: *Found. Comput. Math.* 4.4 (2004), pp. 369–396 (cit. on pp. 39–42).

[Bür20] Peter Bürgisser. "Correction to: The Complexity of Factors of Multivariate Polynomials". In: *Found. Comput. Math.* 20.6 (2020), pp. 1667–1668 (cit. on p. 42).

[Bür24] Peter Bürgisser. *Completeness Classes in Algebraic Complexity Theory*. 2024. arXiv: 2406.06217 [cs] (cit. on p. 8).

[Bür99] Peter Bürgisser. "On the Structure of Valiant's Complexity Classes". In: *Discrete Math. Theor. Comput. Sci.* 3.3 (1999), pp. 73–94 (cit. on p. 9).

[CDS24] Sayak Chakrabarti, Ashish Dwivedi, and Nitin Saxena. "Solving Polynomial Systems over Non-Fields and Applications to Modular Polynomial Factoring". In: *J. Symbolic Comput.* 125 (2024), Paper No. 102314, 29 (cit. on p. 23).

[CG00] David G. Cantor and Daniel M. Gordon. "Factoring Polynomials over p-adic Fields". In: *Algorithmic Number Theory (Leiden, 2000)*. Vol. 1838. Lecture Notes in Comput. Sci. Springer, Berlin, 2000, pp. 185–208 (cit. on p. 22).

[Chi87] A. L. Chistov. "Efficient Factorization of Polynomials over Local Fields". In: *Doklady Akademii Nauk SSSR* 293.5 (1987), pp. 1073–1077 (cit. on p. 22).

[CKS19a] Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. *Closure of VP under Taking Factors: A Short and Simple Proof*. 2019. arXiv: 1903.02366 [cs] (cit. on p. 45).

[CKS19b] Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. "Closure Results for Polynomial Factorization". In: *Theory Comput.* 15 (2019), Paper No. 13, 34 (cit. on pp. 31, 32, 34, 38, 46).

[CKW10] Xi Chen, Neeraj Kayal, and Avi Wigderson. "Partial Derivatives in Arithmetic Complexity and Beyond". In: *Found. Trends Theor. Comput. Sci.* 6.1-2 (2010), front matter, 1–138 (cit. on p. 9).

[Coo71] Stephen A. Cook. "The Complexity of Theorem-Proving Procedures". In: *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 1971, pp. 151–158 (cit. on p. 9).

[CR79] Stephen A. Cook and Robert A. Reckhow. "The Relative Efficiency of Propositional Proof Systems". In: *J. Symbolic Logic* 44.1 (1979), pp. 36–50 (cit. on p. 15).

[DG24] Pranjal Dutta and Sumanta Ghosh. "Complexity Theory Column 121: Advances in Polynomial Identity Testing". In: *ACM SIGACT News* 55.2 (2024), pp. 53–88 (cit. on p. 11).

[DGV24] Ashish Dwivedi, Zeyu Guo, and Ben Lee Volk. "Optimal Pseudorandom Generators for Low-Degree Polynomials over Moderately Large Fields". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Vol. 317. LIPIcs. Leibniz Int. Proc. Inform. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2024, Art. No. 44, 19 (cit. on p. 15).

[DL78] Richard A. DeMillo and Richard J. Lipton. "A Probabilistic Remark on Algebraic Program Testing". In: *Inf. Process. Lett.* 7.4 (1978), pp. 193–195 (cit. on p. 11).

[DMS21] Ashish Dwivedi, Rajat Mittal, and Nitin Saxena. "Efficiently Factoring Polynomials modulo p^4". In: *J. Symbolic Comput.* 104 (2021), pp. 805–823 (cit. on p. 23).

[DPS20] Jintai Ding, Albrecht Petzoldt, and Dieter S. Schmidt. *Multivariate Public Key Cryptosystems*. Vol. 80. Advances in Information Security. Springer, New York, 2020 (cit. on p. 15).

[DSS22] Pranjal Dutta, Nitin Saxena, and Amit Sinhababu. "Discovering the Roots: Uniform Closure Results for Algebraic Classes under Factoring". In: *J. ACM* 69.3 (2022), Art. 18, 39 (cit. on pp. 34, 40, 46).

[DST24] Pranjal Dutta, Amit Sinhababu, and Thomas Thierauf. "Derandomizing Multivariate Polynomial Factoring for Low Degree Factors". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Vol. 317. LIPIcs. Leibniz Int. Proc. Inform. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2024, Art. No. 75, 20 (cit. on p. 38).

[DSY09] Zeev Dvir, Amir Shpilka, and Amir Yehudayoff. "Hardness-Randomness Tradeoffs for Bounded Depth Arithmetic Circuits". In: *SIAM J. Comput.* 39.4 (2009), pp. 1279–1293 (cit. on pp. 35, 36).

[For+21] Michael A. Forbes, Amir Shpilka, Iddo Tzameret, and Avi Wigderson. "Proof Complexity Lower Bounds from Algebraic Circuit Complexity". In: *Theory Comput.* 17 (2021), Paper No. 10, 88 (cit. on p. 15).

[For14] Michael Andrew Forbes. "Polynomial Identity Testing of Read-Once Oblivious Algebraic Branching Programs". PhD thesis. Massachusetts Institute of Technology, 2014 (cit. on p. 13).

[For15] Michael A. Forbes. "Deterministic Divisibility Testing via Shifted Partial Derivatives". In: *2015 IEEE 56th Annual Symposium on Foundations of Computer Science—FOCS 2015*. IEEE Computer Soc., Los Alamitos, CA, 2015, pp. 451–465 (cit. on p. 12).

[Fre07] Günther Frei. "The Unpublished Section Eight: On the Way to Function Fields over a Finite Field". In: *The Shaping of Arithmetic after C. F. Gauss's It Disquisitiones Arithmeticae*. Springer, Berlin, 2007, pp. 159–198 (cit. on p. 15).

[FS13] Michael A. Forbes and Amir Shpilka. "Quasipolynomial-Time Identity Testing of Non-Commutative and Read-Once Oblivious Algebraic Branching Programs". In: *2013 IEEE 54th Annual Symposium on Foundations of Computer Science—FOCS 2013*. IEEE Computer Soc., Los Alamitos, CA, 2013, pp. 243–252 (cit. on p. 12).

[FS15] Michael A. Forbes and Amir Shpilka. "Complexity Theory Column 88: Challenges in Polynomial Factorization". In: *ACM SIGACT News* 46.4 (2015), pp. 32–49 (cit. on pp. 3, 12, 15).

[Gat06] Joachim von zur Gathen. "Who Was Who in Polynomial Factorization: 1". In: *Proceedings of the 2006 International Symposium on Symbolic and Algebraic Computation*. ISSAC '06. Association for Computing Machinery, 2006, p. 2 (cit. on p. 3).

[Gat84] Joachim von zur Gathen. "Hensel and Newton Methods in Valuation Rings". In: *Math. Comp.* 42.166 (1984), pp. 637–661 (cit. on p. 45).

[GG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Third. Cambridge University Press, Cambridge, 2013 (cit. on pp. 3, 4, 18, 45).

[GK85] Joachim von zur Gathen and Erich Kaltofen. "Factoring Sparse Multivariate Polynomials". In: *J. Comput. System Sci.* 31 (1985), pp. 265–287 (cit. on p. 47).

[GP01] Joachim von zur Gathen and Daniel Panario. "Factoring Polynomials over Finite Fields: A Survey". In: *J. Symbolic Comput.* 31.1 (2001), pp. 3–17 (cit. on p. 3).

[Gro12] Joshua A. Grochow. "Symmetry and Equivalence Relations in Classical and Geometric Complexity Theory". PhD thesis. University of Chicago, 2012 (cit. on p. 41).

[Gro13] Joshua Grochow. *Degree Restriction for Polynomials in* VP. Theoretical Computer Science Stack Exchange. 2013 (cit. on p. 26).

[Gro20] Joshua A. Grochow. "Complexity in Ideals of Polynomials: Questions on Algebraic Complexity of Circuits and Proofs". In: *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS* 130 (2020), pp. 40–62 (cit. on p. 15).

[GRS23] Venkat Guruswami, Atri Rudra, and Madhu Sudan. "Essential Coding Theory". Book Draft. 2023 (cit. on p. 14).

[GS13] S. B. Gashkov and I. S. Sergeev. "Complexity of Computation in Finite Fields". In: *J. Math. Sci.* 191.5 (2013), pp. 661–685 (cit. on p. 4).

[GS92] Joachim von zur Gathen and Victor Shoup. "Computing Frobenius Maps and Factoring Polynomials". In: *Comput. Complexity* 2.3 (1992), pp. 187–224 (cit. on pp. 5, 7).

[GS99] Venkatesan Guruswami and Madhu Sudan. "Improved Decoding of Reed-Solomon and Algebraic-Geometry Codes". In: *Institute of Electrical and Electronics Engineers* 45.6 (1999), pp. 1757–1767 (cit. on p. 15).

[GTZ88] Patrizia Gianni, Barry Trager, and Gail Zacharias. "Gröbner Bases and Primary Decomposition of Polynomial Ideals". In: *J. Symbolic Comput.* 6.2-3 (1988), pp. 149–167 (cit. on p. 15).

[Guo+22] Zeyu Guo, Mrinal Kumar, Ramprasad Saptharishi, and Noam Solomon. "Derandomization from Algebraic Hardness". In: *SIAM J. Comput.* 51.2 (2022), pp. 315–335 (cit. on pp. 13, 14).

[Gup+16] Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. "Arithmetic Circuits: A Chasm at Depth 3". In: *SIAM J. Comput.* 45.3 (2016), pp. 1064–1079 (cit. on p. 35).

[Hen04] Kurt Hensel. "Neue Grundlagen Der Arithmetik". In: *J. Reine Angew. Math.* 127 (1904), pp. 51–84 (cit. on p. 15).

[Hen08] Kurt Hensel. *Theorie der algebraischen Zahlen*. Vol. 1. BG Teubner, 1908 (cit. on p. 15).

[Hen18] Kurt Hensel. "Eine Neue Theorie Der Algebraischen Zahlen". In: *Math. Z.* 2.3-4 (1918), pp. 433–452 (cit. on p. 15).

[Hen97] Kurt Hensel. "Über eine neue Begründung der Theorie der algebraischen Zahlen." In: *Jahresbericht der Deutschen Mathematiker-Vereinigung* 6 (1897), pp. 83–88 (cit. on p. 15).

[HG23] Qiao-Long Huang and Xiao-Shan Gao. "New Sparse Multivariate Polynomial Factorization Algorithms over Integers". In: *Proceedings of the International Symposium on Symbolic & Algebraic Computation (ISSAC 2023)*. ACM, New York, 2023, pp. 315–324 (cit. on p. 49).

[HS80] Joos Heintz and Claus-Peter Schnorr. "Testing Polynomials Which Are Easy to Compute (Extended Abstract)". In: *Proceedings of the 12th Annual ACM Symposium on Theory of Computing, April 28-30, 1980, Los Angeles, California, USA*. ACM, 1980, pp. 262–272 (cit. on p. 12).

[Iva+12] Gábor Ivanyos, Marek Karpinski, Lajos Rónyai, and Nitin Saxena. "Trading GRH for Algebra: Algorithms for Factoring Polynomials and Related Structures". In: *Math. Comp.* 81.277 (2012), pp. 493–531 (cit. on pp. 7, 15).

[Jan11] Maurice J. Jansen. "Extracting Roots of Arithmetic Circuits by Adapting Numerical Methods". In: *Innovations in Computer Science - ICS 2011*. Tsinghua University Press, 2011, pp. 87–100 (cit. on p. 29).

[Juk12] Stasys Jukna. *Boolean Function Complexity*. Vol. 27. Algorithms and Combinatorics. Springer, Heidelberg, 2012 (cit. on p. 15).

[Kal82] Erich Kaltofen. "Factorization of Polynomials". In: *Computer Algebra: Symbolic and Algebraic Computation*. Vienna: Springer, 1982, pp. 95–113 (cit. on p. 3).

[Kal85] Erich Kaltofen. "Polynomial-Time Reductions from Multivariate to Bi- and Univariate Integral Polynomial Factorization". In: *SIAM J. Comput.* 14.2 (1985), pp. 469–489 (cit. on p. 25).

[Kal87] Erich Kaltofen. "Single-Factor Hensel Lifting and Its Application to the Straight-Line Complexity of Certain Polynomials". In: *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*. STOC '87. Association for Computing Machinery, 1987 (cit. on pp. 27, 39, 40).

[Kal89] Erich Kaltofen. "Factorization of Polynomials given by Straight-Line Programs". In: *Adv. Comput. Res.* 5 (1989), pp. 375–412 (cit. on p. 25).

[Kal90] Erich Kaltofen. "Polynomial Factorization 1982–1986". In: *Computers in mathematics*. CRC Press, 1990, pp. 285–309 (cit. on p. 3).

[Kal92] Erich Kaltofen. "Polynomial Factorization 1987–1991". In: *LATIN '92 (São Paulo, 1992)*. Vol. 583. Lecture Notes in Comput. Sci. Springer, Berlin, 1992, pp. 294–313 (cit. on p. 3).

[Kal95] Erich Kaltofen. "Effective Noether Irreducibility Forms and Applications". In: *J. Comput. System Sci.* 50.2 (1995), pp. 274–295 (cit. on p. 25).

[KI04] Valentine Kabanets and Russell Impagliazzo. "Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds". In: *Comput. Complexity* 13.1-2 (2004), pp. 1–46 (cit. on pp. 12–14, 35).

[KK08] Erich Kaltofen and Pascal Koiran. "Expressing a Fraction of Two Determinants as a Determinant". In: *ISSAC 2008*. ACM, New York, 2008, pp. 141–146 (cit. on p. 29).

[KL94] Erich L. Kaltofen and Austin Lobo. "Factoring High-Degree Polynomials by the Black Box Berlekamp Algorithm". In: *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC '94*. ACM, 1994, pp. 90–98 (cit. on p. 7).

[Koi12] Pascal Koiran. "Arithmetic Circuits: The Chasm at Depth Four Gets Wider". In: *Theoret. Comput. Sci.* 448 (2012), pp. 56–65 (cit. on p. 35).

[Kop14] Swastik Kopparty. *Algorithmic Number Theory*. 2014. Lecture notes, Rutgers University (cit. on p. 6).

[KP13] Steven G. Krantz and Harold R. Parks. *The Implicit Function Theorem*. Modern Birkhäuser Classics. Birkhäuser/Springer, New York, 2013 (cit. on p. 35).

[Kra19] Jan Krajíček. *Proof Complexity*. Vol. 170. Encyclopedia of Mathematics and Its Applications. Cambridge University Press, Cambridge, 2019 (cit. on p. 15).

[Kra95] Jan Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*. Vol. 60. Encyclopedia of Mathematics and Its Applications. Cambridge University Press, Cambridge, 1995 (cit. on p. 15).

[KRS24] Mrinal Kumar, Varun Ramanathan, and Ramprasad Saptharishi. "Deterministic Algorithms for Low Degree Factors of Constant Depth Circuits". In: *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, Philadelphia, PA, 2024, pp. 3901–3918 (cit. on p. 38).

[KS06] Neeraj Kayal and Nitin Saxena. "Complexity of Ring Morphism Problems". In: *Comput. Complexity* 15.4 (2006), pp. 342–390 (cit. on p. 15).

[KS09] Zohar S. Karnin and Amir Shpilka. "Reconstruction of Generalized Depth-3 Arithmetic Circuits with Bounded Top Fan-In". In: *24th Annual IEEE Conference on Computational Complexity*. IEEE Computer Soc., Los Alamitos, CA, 2009, pp. 274–285 (cit. on p. 15).

[KS19] Mrinal Kumar and Ramprasad Saptharishi. "Hardness-Randomness Tradeoffs for Algebraic Computation". In: *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS* 129 (2019), pp. 56–87 (cit. on p. 14).

[KS98] Erich Kaltofen and Victor Shoup. "Subquadratic-Time Factoring of Polynomials over Finite Fields". In: *Math. Comp.* 67.223 (1998), pp. 1179–1197 (cit. on p. 7).

[KSS15] Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka. "Equivalence of Polynomial Identity Testing and Polynomial Factorization". In: *Comput. Complexity* 24.2 (2015), pp. 295–331 (cit. on pp. 10, 11, 17).

[KST23] Mrinal Kumar, Ramprasad Saptharishi, and Anamay Tengse. "Near-Optimal Bootstrapping of Hitting Sets for Algebraic Models". In: *Theory Comput.* 19 (2023), Paper No. 12, 30 (cit. on p. 14).

[KT90] Erich Kaltofen and Barry M. Trager. "Computing with Polynomials given by Black Boxes for Their Evaluations: Greatest Common Divisors, Factorization, Separation of Numerators and Denominators". In: *J. Symbolic Comput.* 9.3 (1990), pp. 301–320 (cit. on p. 24).

[KU11] Kiran S. Kedlaya and Christopher Umans. "Fast Polynomial Factorization and Modular Composition". In: *SIAM J. Comput.* 40.6 (2011), pp. 1767–1802 (cit. on p. 7).

[Kum+24] Mrinal Kumar, Varun Ramanathan, Ramprasad Saptharishi, and Ben Lee Volk. *Towards Deterministic Algorithms for Constant-Depth Factors of Constant-Depth Circuits*. 2024. arXiv: 2403.01965 [cs] (cit. on p. 38).

[Lan17] J. M. Landsberg. *Geometry and Complexity Theory*. Vol. 169. Cambridge Studies in Advanced Mathematics. Cambridge University Press, Cambridge, 2017 (cit. on p. 41).

[LLL82] A. K. Lenstra, H. W. Lenstra Jr., and L. Lovász. "Factoring Polynomials with Rational Coefficients". In: *Math. Ann.* 261.4 (1982), pp. 515–534 (cit. on p. 21).

[LS78] Richard J. Lipton and Larry J. Stockmeyer. "Evaluation of Polynomials with Super-Preconditioning". In: *J. Comput. System Sci.* 16.2 (1978), pp. 124–139 (cit. on p. 39).

[Mah14] Meena Mahajan. "Algebraic Complexity Classes". In: *Perspectives in Computational Complexity*. Vol. 26. Progr. Comput. Sci. Appl. Logic. Birkhäuser/Springer, Cham, 2014, pp. 51–75 (cit. on pp. 9, 28).

[Mal03] Guillaume Malod. "Polynômes et coefficients". PhD thesis. Université Claude Bernard - Lyon, 2003 (cit. on p. 39).

[Mal07] Guillaume Malod. "The Complexity of Polynomials and Their Coefficient Functions". In: *Twenty-Second Annual IEEE Conference on Computational Complexity (CCC'07)*. 2007, pp. 193–204 (cit. on p. 39).

[MOV97] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press Series on Discrete Mathematics and Its Applications. CRC Press, Boca Raton, FL, 1997 (cit. on p. 15).

[MP08] Guillaume Malod and Natacha Portier. "Characterizing Valiant's Algebraic Complexity Classes". In: *J. Complexity* 24.1 (2008), pp. 16–38 (cit. on pp. 32, 33).

[MS01] Ketan D. Mulmuley and Milind Sohoni. "Geometric Complexity Theory. I. An Approach to the P vs. NP and Related Problems". In: *SIAM J. Comput.* 31.2 (2001), pp. 496–526 (cit. on p. 41).

[MS08] Ketan D. Mulmuley and Milind Sohoni. "Geometric Complexity Theory. II. Towards Explicit Obstructions for Embeddings among Class Varieties". In: *SIAM J. Comput.* 38.3 (2008), pp. 1175–1206 (cit. on p. 41).

[Mul11] Ketan D. Mulmuley. "On P vs. NP and Geometric Complexity Theory". In: *J. ACM* 58.2 (2011), Art. 5, 26 (cit. on p. 41).

[Mul12] Ketan D. Mulmuley. "The GCT Program toward the P vs. NP Problem". In: *Commun. ACM* 55.6 (2012), pp. 98–107 (cit. on p. 41).

[Mul17] Ketan D. Mulmuley. "Geometric Complexity Theory V: Efficient Algorithms for Noether Normalization". In: *J. Amer. Math. Soc.* 30.1 (2017), pp. 225–309 (cit. on p. 15).

[NV10] Phong Q. Nguyen and Brigitte Vallée, eds. *The LLL Algorithm: Survey and Applications*. Information Security and Cryptography. Berlin, Heidelberg: Springer, 2010 (cit. on p. 15).

[NW94] Noam Nisan and Avi Wigderson. "Hardness vs. Randomness". In: *J. Comput. System Sci.* 49.2 (1994), pp. 149–167 (cit. on pp. 12, 13).

[Oli16] Rafael Oliveira. "Factors of Low Individual Degree Polynomials". In: *Comput. Complexity* 25.2 (2016), pp. 507–561 (cit. on p. 36).

[Ore22] Øystein Ore. "Über höhere kongruenzen". In: *Norsk Mat. Forenings Skrifter* (1922) (cit. on p. 11).

[Ost99] Alexander Ostrowski. "On the Significance of the Theory of Convex Polyhedra for Formal Algebra". In: *SIGSAM Bull.* 33.1 (1999), p. 5 (cit. on p. 48).

[Reg02] Kenneth W. Regan. "Understanding the Mulmuley-Sohoni Approach to P vs. NP". In: *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS* 78 (2002), pp. 86–99 (cit. on p. 41).

[RS05] Ran Raz and Amir Shpilka. "Deterministic Polynomial Identity Testing in Non-Commutative Models". In: *Comput. Complexity* 14.1 (2005), pp. 1–19 (cit. on p. 12).

[Sap17] Ramprasad Saptharishi. "Algebra and Computation". Lecture Notes. TIFR, Mumbai, 2017 (cit. on p. 21).

[Sap21] Ramprasad Saptharishi. "A Survey of Lower Bounds in Arithmetic Circuit Complexity". Github Survey, 2021 (cit. on p. 9).

[Sax09] Nitin Saxena. "Progress on Polynomial Identity Testing". In: *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS* 99 (2009), pp. 49–79 (cit. on p. 11).

[Sax14] Nitin Saxena. "Progress on Polynomial Identity Testing-II". In: *Perspectives in Computational Complexity*. Vol. 26. Progr. Comput. Sci. Appl. Logic. Birkhäuser/Springer, Cham, 2014, pp. 131–146 (cit. on p. 11).

[Sax23] Nitin Saxena. *Closure of Algebraic Classes Under Factoring*. 2023. Talk at Recent Trends in Computer Algebra (2023) in Institut Henri Poincaré, Paris. (Cit. on p. 50).

[Sch00] Andrzej Schinzel. *Polynomials with Special Regard to Reducibility*. Vol. 77. Encyclopedia of Mathematics and Its Applications. Cambridge University Press, Cambridge, 2000 (cit. on p. 48).

[Sch80] J. T. Schwartz. "Fast Probabilistic Algorithms for Verification of Polynomial Identities". In: *J. ACM* 27.4 (1980), pp. 701–717 (cit. on p. 11).

[Sho09] Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. 2nd ed. Cambridge University Press, Cambridge, 2009 (cit. on p. 3).

[Sin16] Gaurav Sinha. "Reconstruction of Real Depth-3 Circuits with Top Fan-in 2". In: *31st Conference on Computational Complexity*. Vol. 50. LIPIcs. Leibniz Int. Proc. Inform. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2016, Art. No. 31, 53 (cit. on p. 15).

[Sin19] Amit Sinhababu. "Power series in complexity: Algebraic Dependence, Factor Conjecture and Hitting Set for Closure of VP". PhD thesis. Indian Institute of Technology Kanpur, 2019 (cit. on p. 40).

[Sin22] Gaurav Sinha. "Efficient Reconstruction of Depth Three Arithmetic Circuits with Top Fan-in Two". In: *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*. Vol. 215. Leibniz International Proceedings in Informatics (Lipics). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 118:1–118:33 (cit. on p. 15).

[SS25] Shubhangi Saraf and Devansh Shringi. *Reconstruction of Depth 3 Arithmetic Circuits with Top Fan-in 3*. 2025. Electronic Colloquium on Computational Complexity: TR25-008 (cit. on p. 15).

[SSS13] Chandan Saha, Ramprasad Saptharishi, and Nitin Saxena. "A Case of Depth-3 Identity Testing, Sparse Factorization and Duality". In: *Comput. Complexity* 22.1 (2013), pp. 39–69 (cit. on p. 11).

[ST21] Amit Sinhababu and Thomas Thierauf. "Factorization of Polynomials given by Arithmetic Branching Programs". In: *Comput. Complexity* 30.2 (2021), Paper No. 15, 47 (cit. on pp. 29, 30).

[Str73] Volker Strassen. "Vermeidung von Divisionen". In: *J. Reine Angew. Math.* 264 (1973), pp. 184–202 (cit. on p. 10).

[Sud97] Madhu Sudan. "Decoding of Reed Solomon Codes beyond the Error-Correction Bound". In: *J. Complexity* 13.1 (1997), pp. 180–193 (cit. on p. 15).

[Sud98] Madhu Sudan. "Algebra and Computation". Lecture Notes. Massachusetts Institute of Technology, 1998 (cit. on p. 25).

[SV10] Amir Shpilka and Ilya Volkovich. "On the Relation between Polynomial Identity Testing and Finding Variable Disjoint Factors". In: *Automata, Languages and Programming. (ICALP 2010)*. Vol. 6198. Lecture Notes in Comput. Sci. Springer, Berlin, 2010, pp. 408–419 (cit. on pp. 11, 48).

[SY10] Amir Shpilka and Amir Yehudayoff. "Arithmetic Circuits: A Survey of Recent Results and Open Questions". In: *Found. Trends Theor. Comput. Sci.* 5.3-4 (2010), 207–388 (2010) (cit. on pp. 9, 11, 12, 15, 38).

[Tav15] Sébastien Tavenas. "Improved Bounds for Reduction to Depth 4 and Depth 3". In: *Inform. and Comput.* 240 (2015), pp. 2–11 (cit. on p. 35).

[Val+83] L. G. Valiant, S. Skyum, S. Berkowitz, and C. Rackoff. "Fast Parallel Computation of Polynomials Using Few Processors". In: *SIAM J. Comput.* 12.4 (1983), pp. 641–644 (cit. on p. 35).

[Val79] L. G. Valiant. "Completeness Classes in Algebra". In: *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing (STOC)*. ACM, New York, 1979, pp. 249–261 (cit. on pp. 8, 9, 31).

[Val82] L. G. Valiant. "Reducibility by Algebraic Projections". In: *Logic and Algorithmic (Zurich, 1980)*. Vol. 30. Monogr. Enseign. Math. Univ. Genève, Geneva, 1982, pp. 365–380 (cit. on pp. 8, 32).

[Vol15] Ilya Volkovich. "Deterministically Factoring Sparse Polynomials into Multilinear Factors and Sums of Univariate Polynomials". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Vol. 40. LIPIcs. Leibniz Int. Proc. Inform. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2015, pp. 943–958 (cit. on p. 48).

[Vol17] Ilya Volkovich. "On Some Computations on Sparse Polynomials". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Vol. 81. LIPIcs. Leibniz Int. Proc. Inform. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2017, Art. No. 48, 21 (cit. on p. 48).

[Yun76] David Y. Y. Yun. "Hensel Meets Newton–Algebraic Constructions in an Analytic Setting". In: *Analytic Computational Complexity (Proc. Sympos., Carnegie-Mellon Univ., Pittsburgh, Pa., 1975)*. Academic Press, New York-London, 1976, pp. 205–215 (cit. on p. 45).

[Zip79] Richard Zippel. "Probabilistic Algorithms for Sparse Polynomials". In: *Symbolic and Algebraic Computation (EUROSAM '79, Internat. Sympos., Marseille, 1979)*. Vol. 72. Lecture Notes in Comput. Sci. Springer, Berlin-New York, 1979, pp. 216–226 (cit. on p. 11).

[Zip81] Richard Zippel. "Newton's Iteration and the Sparse Hensel Algorithm (Extended Abstract)". In: *Proceedings of the Fourth ACM Symposium on Symbolic and Algebraic Computation*. SYMSAC '81. New York, NY, USA: Association for Computing Machinery, 1981, pp. 68–72 (cit. on p. 45).