Electronic Colloquium on Computational Complexity, Report No. 168 (2014)

Deterministically Factoring Sparse Polynomials into Multilinear Factors

Ilya Volkovich *

Abstract

We present the first efficient deterministic algorithm for factoring sparse polynomials that split into multilinear factors. Our result makes partial progress towards the resolution of the classical question posed by von zur Gathen and Kaltofen in [GK85] to devise an efficient deterministic algorithm for factoring (general) sparse polynomials. We achieve our goal by introducing *essential factorization schemes* which can be thought of a relaxation of the regular factorization notion.

1 Introduction

In this paper we study the problem of factorization of sparse polynomials.

1.1 Multivariate Polynomial Factorization

One of the fundamental problems in algebraic complexity is the problem of polynomial factorization: given a polynomial $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ over a field $\mathbb{F}$, find its irreducible factors. Other than being natural, the problem has many applications such as list decoding [Sud97, GS99] and derandomization [KI04]. A large amount of research has been devoted to finding efficient algorithms for this problem (see e.g. [GG99]) and numerous *randomized* algorithms were designed [GK85, Kal89, KT90, GG99, Kal03, Gat06]. However, the question of whether there exist *deterministic* algorithms for this problem remains an interesting open question (see [GG99, Kay07]).

1.2 Sparse Polynomials

Let $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a n-variate polynomial over the field $\mathbb{F}$. We denote by $\|P\|$ the *sparsity* of P. That is, the number of non-zero monomials in P. Suppose that the individual degree of each variable x_i is bounded by d, then the above number can reach $(d+1)^n$. Our case of interest is when $\|P\| \ll (d+1)^n$. Indeed, in various applications [Zip79, GK85, BOT88, GJR10, SV10, SV11] the desired regime is when $\|P\| = \text{poly}(n, d)$. Such polynomials are refereed to as *sparse* polynomials. Coming up with an efficient deterministic factorization algorithm for sparse polynomials (given as a list of monomials) is a classical open question posed by von zur Gathen and Kaltofen in [GK85]. An inherent difficulty in tackling the problem lies within the fact that a factor of a sparse polynomial need not be sparse. The following example demonstrates that a blow-up in the sparsity of a factor can be super-polynomial over every field. A similar example appears as Example 5.1 in [GK85].

*Departments of Math and EECS, University of Michigan, Ann Arbor, MI. Email: ilyavol@umich.edu.

ISSN 1433-8092

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

Consequently, just writing down the irreducible factors as lists of monomials can take super-polynomial time. In fact, the randomized algorithm of [GK85] assumes that the upper bound on the sparsity of the factors is known. In light of this difficulty, a simpler problem was posed in that same paper: Given $m+1$ sparse polynomials $f_1, f_2, \ldots f_m, g$ test if $g = f_1 \cdot f_2 \cdot \ldots \cdot f_m$. This problem is referred to as "testing sparse factorization".

Over the last three decades this question has seen only a very partial progress. For the testing version of the problem, Saha et al. [SSS13] presented an efficient deterministic algorithm for the special case when the sparse polynomials are sums of univariate polynomials. Shpilka & Volkovich [SV10] gave efficient deterministic factorization algorithms for multilinear sparse polynomials (see Lemma 2.3 for more details). In this work, we make another step towards the resolution of the problem. We consider the model of sparse polynomials that split into multilinear factors or "multilinearly-split" for short. Formally, we say that a polynomial P is *multilinearly-split* if it can be written as a product of multilinear polynomials.

Clearly, this model extends the one considered by Shpilka & Volkovich. Moreover, it can be seen as a multivariate version of algebraically closed fields in the following sense. The only irreducible univariate polynomials over algebraically closed fields are linear polynomials (i.e. $\alpha x+\beta$) since every univariate polynomial splits into linear factors. However, this is not the case in the multivariate setting. For example, the polynomial $P(x, y) = x^2 + y$ is irreducible over any field. In our model, the above phenomenon does not occur as every polynomial splits into multilinear factors. In addition, our model evades the aforementioned inherent difficulty since a multilinear factor of a sparse polynomial is itself a sparse polynomial (see Lemma 4.1 for more details). Below is our main result:

Theorem 1 (Main)**.** *There exists a deterministic algorithm that given a s-sparse multilinearly-split polynomial $F \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of degree d outputs its irreducible multilinear factors. The running time of the algorithm is* $\mathrm{poly}(n, d, s, p, \ell)$ *when $\mathbb{F} = \mathbb{F}_{p^\ell}$ and* $\mathrm{poly}(n, d, s, b)$ *when $\mathbb{F} = \mathbb{Q}$ and b is the bit complexity of the coefficients in P.*

The running time of our algorithm is essentially optimal, as we are invoking the state-of-the-art deterministic factoring algorithm for a constant number of variables. We note that even for the univariate case the best known deterministic factoring algorithm has a polynomial dependence on the characteristic p. While in the randomized setting, the dependence is polynomial in $\log p$. A lot of effort was invested in trying to derandomize the factorization algorithm when the characteristic of $\mathbb{F}$ is large (see e.g. [Sho91, GG99, GKL04, Kay07]).

1.3 Techniques

Let $\mathcal{C}$ be a class of polynomials and let $\mathrm{fact}(\mathcal{C})$ denote the class of factors of $P \in \mathcal{C}$. Suppose we want to map n-variate polynomials from $\mathcal{C}$ to (other) polynomials with, potentially, a smaller number of variables in a way that two distinct (composite) polynomials $P, Q \in \mathcal{C}$ remain distinct under the map. In particular, this implies that each irreducible polynomial in $\mathrm{fact}(\mathcal{C})$ must be mapped into a non-constant polynomial. Moreover, a pair of non-similar, irreducible polynomials must be

mapped into a pair of non-similar polynomials. And, ideally, an irreducible polynomial should remain irreducible. Those goals are achieved in [GK85, Kal85, Kal89, KT90] and other works by considering a projection to a random low-dimensional space (i.e a line or a plane). The purpose of the last two requirements is to ensure that factors from different images could not be combined together. In other words, there is only one way to interpret a product of images under the map. As preserving irreducibility deterministically is still an open question, we introduce a relaxation of the original requirements with the hope that it would be easier to fulfill. We call it an *essential factorization scheme*.

Consider a polynomial map $\{\mathrm{H}\}_n = \mathbb{F}^{t(n)} \to \mathbb{F}^n$ with $t \ll n$ such that for every irreducible $P \in \mathrm{fact}(\mathcal{C})$ the composition $P(\mathrm{H})$ might be reducible, yet results in a polynomial that contains an irreducible "essential" factor g_P which describes P uniquely. In addition, g_P cannot be a factor of any $Q(\mathrm{H})$ if $P \neq Q$. Given that property of H, for any $F \in \mathcal{C}$ the polynomial $F(\mathrm{H})$ describes F uniquely. Consequently, for any $F, R \in \mathcal{C}$ we get that $F \equiv R \iff F(\mathrm{H}) = R(\mathrm{H})$. We formalize this notion in Definition 3.1.

Observe that this reasoning can be extended to handle products of polynomials for $\mathcal{C}$. That is, $\prod_{i=1}^{k} F_1$. Consequently, if we could establish an essential factorization scheme for sparse polynomials, we would solve the sparse factorization testing problem.

Unfortunately, we are not there yet for the entire set of sparse polynomials. In this paper, we make a step towards this goal by establishing an essential factorization scheme for multilinear sparse polynomials. In fact, our scheme has an additional property: given a factor, we can efficiently decide whether or not it is an essential factor of some polynomial P. Moreover, we can efficiently compute P from its essential factor g_P. Consequently, in order to compute the irreducible factors of multilinearly-split polynomial F, we first compute the irreducible factor of $F(\mathrm{H})$ and then recover the "original" factors of F. We note that since $F(\mathrm{H})$ is t-variate polynomial with $t \ll n$, we can carry out the factorization phase deterministically by a brute-force derandomization of the best randomized algorithm while still being efficient. Formally, see Lemma 4.2

We show that our essential factorization scheme works for some other classes of multilinear polynomials as well. Our construction can be seen as another link in the line of works [KK05, SV10, KSS14] that connect polynomial factoring and polynomial identity testing.

1.4 Organization

We start by some basic definitions and notation in Section 2. In Section 3, we formally introduce essential factorization schemes and demonstrate their properties. In that same section, we also construct such a scheme for classes of multilinear polynomials. Finally in Section 4, we present our factoring algorithm for sparse multilinearly-split polynomials, thus proving our main theorem. We conclude the paper with some remarks in Section 5.

2 Preliminaries

Let $\mathbb{F}$ denote a field, finite or otherwise, and let $\overline{\mathbb{F}}$ denote its algebraic closure. We assume that elements of $\mathbb{F}$ are represented in binary using some standard encoding. Moreover, we assume that there is an algorithm that given an integer r, outputs in time $\mathrm{poly}(r)$ a set of r distinct elements in $\mathbb{F}$ (or an extension field of $\mathbb{F}$) each of which is represented in this encoding using $O(\log r)$ bits.

2.1 Polynomials

A polynomial $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ *depends* on a variable x_i if there are two inputs $\bar{\alpha}, \bar{\beta} \in \overline{\mathbb{F}}^n$ differing only in the i^{th} coordinate for which $P(\bar{\alpha}) \neq P(\bar{\beta})$. We denote by $\mathrm{var}(P)$ the set of variables that P depends on. We say that P is Q and denote by it $P \sim Q$ if $P = cQ$ for some $c \in \mathbb{F}$.

For a polynomial $P(x_1, \ldots, x_n)$, a variable x_i and a field element α, we denote with $P|_{x_i=\alpha}$ the polynomial resulting from substituting α to x_i. Similarly given a subset $I \subseteq [n]$ and an assignment $\bar{a} \in \mathbb{F}^n$, we define $P|_{x_I=\bar{a}_I}$ to be the polynomial resulting from substituting a_i to x_i for every $i \in I$.

Definition 2.1. *For $P, Q \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ and $\ell \in [n]$ let $D_\ell(P, Q)$ be the polynomial defined as follows:*

$$D_\ell(P,Q)(\bar{x}) \triangleq \left| \begin{pmatrix} P & P|_{x_\ell=0} \\ Q & Q|_{x_\ell=0} \end{pmatrix} \right| (\bar{x}) = (P \cdot Q|_{x_\ell=0} - P|_{x_\ell=0} \cdot Q)(\bar{x}).$$

Note that D_ℓ is a bilinear transformation. The following lemma from [SV11] gives a useful property of D_ℓ that is easy to verify.

Lemma 2.2 ([SV11])**.** *Let $P, Q \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be irreducible multilinear polynomials and let $\ell \in \mathrm{var}(P)$. Then $D_\ell(Q, P) \equiv 0$ iff $P \mid Q$.*

The next corollary from [SV10] shows that a multilinear sparse polynomial can be factored efficiently. Moreover, all its factors are sparse.

Lemma 2.3 (Corollary from [SV10])**.** *Given a multilinear polynomial $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$, there is a* $\mathrm{poly}(n, \|P\|)$ *time deterministic algorithm that outputs the* irreducible *factors, $h_1, \ldots, h_k$ of P. Furthermore, $\|h_1\| \cdot \|h_2\| \cdot \ldots \cdot \|h_k\| = \|P\|$.*

2.2 Commutator

The Commutator was originally defined in [SV10] where it was used to devised efficient factorization algorithms for classes of multilinear polynomials. Later, it was also used in reconstruction algorithms [GKL12, SV14] for arithmetic formulae. The following definitions are taken from [SV10].

Definition 2.4. *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial. We say that f is* (x_i, x_j)-decomposable *if f can be written as $f = g \cdot h$ for polynomials g and h such that $i \in \mathrm{var}(g) \setminus \mathrm{var}(h)$ and $j \in \mathrm{var}(h) \setminus \mathrm{var}(g)$.*

Definition 2.5 (Commutator)**.** *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial and let $i, j \in [n]$. We define the* commutator *between x_i and x_j as $\Delta_{ij} f \triangleq f|_{x_i=1,x_j=1} \cdot f|_{x_i=0,x_j=0} - f|_{x_i=1,x_j=0} \cdot f|_{x_i=0,x_j=1}$.*

The crucial property of the commutator is given by the lemma below.

Lemma 2.6 ([SV10])**.** *Let $i, j \in \mathrm{var}(f)$ then f is (x_i, x_j)-decomposable if and only if $\Delta_{ij} f \equiv 0$.*

The following observation connects between Δ_{ij} and D_i.

Observation 2.7. $\Delta_{ij}(P) = D_i\left(P|_{x_j=1}, P|_{x_j=0}\right)$.

2.3 Maps and Generators for Classes of Polynomials

In this section, we formally define the notion of generators and hitting sets for polynomials as well as describe a few of their useful properties. For a further discussion see [SV09, SY10, KMSV13].

A map $\mathcal{G} = (\mathcal{G}^1, \ldots, \mathcal{G}^n) : \mathbb{F}^q \to \mathbb{F}^n$ is a *generator* for the polynomial class $\mathcal{C}$ if for every non-zero n-variate polynomial $P \in \mathcal{C}$, it holds that $P(\mathcal{G}) \not\equiv 0$. The image of the map $\mathcal{G}$ is denoted as $\mathrm{Im}(\mathcal{G}) = \mathcal{G}(\overline{\mathbb{F}}^q)$. Ideally, q should be very small compared to n. A set $\mathcal{H} \subseteq \mathbb{F}^n$ is a *hitting set* for a polynomial class $\mathcal{C}$, if for every non-zero polynomial $P \in \mathcal{C}$, there exists $\bar{a} \in \mathcal{H}$, such that $P(\bar{a}) \neq 0$. A generator can also be viewed as a map containing a hitting set for $\mathcal{C}$ in its image. That is, for every non-zero $P \in \mathcal{C}$, there exists $\bar{a} \in \mathrm{Im}(\mathcal{G})$ such that $P(\bar{a}) \neq 0$. In identity testing, generators and hitting sets play the same role. Given a generator one can easily construct a hitting set by evaluating the generator on a large enough set of points. Conversely in [SV09], an efficient method of constructing a generator from a hitting set was given.

Lemma 2.8 ([SV09])**.** *Let $|\mathbb{F}| > n$. Given a set $\mathcal{H} \subseteq \mathbb{F}^n$, there is an algorithm that runs in time* $\mathrm{poly}(|\mathcal{H}|, n, \log|\mathbb{F}|)$ *and constructs a map $\mathcal{G}(\bar{w}) : \mathbb{F}^t \to \mathbb{F}^n$ such that $\mathcal{G}(\bar{0}) = \bar{0}$, $\mathcal{H} \subseteq \mathrm{Im}(\mathcal{G})$ with $t \triangleq \lceil \log_n |\mathcal{H}| \rceil$ and the individual degrees of $\mathcal{G}^i$ are bounded by $n-1$. Moreover, for each $\bar{a} \in \mathcal{H}$, its preimage, $\bar{\beta} \in \mathbb{F}^q$ s.t. $\bar{a} = \mathcal{G}(\beta)$, can be computed in time* $\mathrm{poly}(|\mathcal{H}|, n)$.

2.4 SV-Generator

The $G_{n,k}$ generator was defined in [SV09] where it was shown that for certain values of k the map $G_{n,k}$ is generator for read-once polynomials. In [KMSV13] this was generalized to multilinear read-k polynomials[1]. We will use the $G_{n,k}$ in our construction.

Definition 2.9 (SV-Generator [SV09])**.** *Let $a_1, \ldots, a_n$ denote n distinct elements from a field $\mathbb{F}$ and for $i \in [n]$ let $L_i(x) \doteq \prod_{j \neq i} \frac{x - a_j}{a_i - a_j}$ denote the corresponding Lagrange interpolant. For every $k \in \mathbb{N}$, define*

$$G_{n,k}(y_1, \ldots, y_k, z_1, \ldots, z_k) \doteq \left(\sum_{j=1}^{k} L_1(y_j) z_j, \sum_{j=1}^{k} L_2(y_j) z_j, \ldots, \sum_{j=1}^{k} L_n(y_j) z_j \right).$$

Let $(G_{n,k})_i$ denote the i^{th} component of $G_{n,k}$; we refer to a_i as the Lagrange constant *associated with this i^{th} component.*

For intuition, it is helpful to view the action of $G_{n,1}(y_1, z_1)$ on a random element of $\mathbb{F}^2$ as selecting a random variable (via the value of y_1) and a random value for that variable (via the value of z_1). This is not completely accurate because for values outside the Lagrange constants the generator does not uniquely select a component. Since the SV-generator is a polynomial map, it is natural to define the sum of two copies of the generator by their component-wise sum and to furthermore view $G_{n,k}$ as the sum of k independent choices of variables and values. For this reason, we take the convention that for two generators $\mathcal{G}_1$ and $\mathcal{G}_2$ with the same output length that $\mathcal{G}_1 + \mathcal{G}_2$ is the generator obtained by adding a sample from $\mathcal{G}_1$ to an independent sample from $\mathcal{G}_2$, and where the seed variables are implicitly relabelled so as to be disjoint. With this convention in mind, the SV-generator has a number of useful properties that follow immediately from its definition.

[1] A read-k polynomial is a polynomial computable by a formula where each variable appears at most k times.

Proposition 2.10 ([SV09, KMSV13])**.** *Let k, k' be positive integers.*

1. $G_{n,k}(\bar{y}, \bar{0}) \equiv \bar{0}$.
2. $G_{n,k}(y_1, \ldots, y_k, z_1, \ldots, z_k)|_{y_k = a_i} = G_{n,k-1}(y_1, \ldots, y_{k-1}, z_1, \ldots, z_{k-1}) + z_k \cdot \bar{e}_i$, *where $\bar{e}_i$ is the 0-1-vector with a single 1 in position i and a_i the i^{th} Lagrange constant.*
3. $G_{n,k}(y_1, \ldots, y_k, z_1, \ldots, z_k) + G_{n,k'}(y_{k+1}, \ldots, y_{k+k'}, z_{k+1}, \ldots, z_{k+k'})$
   $= G_{n,k+k'}(y_1, \ldots, y_{k+k'}, z_1, \ldots, z_{k+k'})$

The first item states that zero is in the image of the SV-generator. The second item shows how to make a single output component (and no others) depend on a particular z_j. The final item shows that sums of independent copies of the SV-generator are equivalent to a single copy of the SV-generator with the appropriate parameter k.

3 Essential Factorization Scheme

In this section, we formally define the notions of essential factors and essential factorization schemes. For a class of polynomials $\mathcal{C}$ we denote by $\text{fact}(\mathcal{C}) = \{P \mid \exists g, P \cdot g \in \mathcal{C}\}$ the class of factors of $\mathcal{C}$.

Definition 3.1 (Essential Factorization Scheme)**.** *Let $\mathcal{C}$ be a class of polynomials over a field $\mathbb{F}$. We say that a polynomial map $\{\mathrm{H}\}_n = \mathbb{F}^{t(n)} \to \mathbb{F}^n$ is an* essential factorization scheme *for $\mathcal{C}$ if there exists (another) map $\Psi_{\mathrm{H}} : \mathbb{F}[x_1, x_2, \ldots, x_n] \to \mathbb{F}[x_1, x_2, \ldots, x_n]$ such that given two irreducible polynomials $P, Q \in \text{fact}(\mathcal{C})$:*

1. *$\Psi_{\mathrm{H}}(P)$ is a non-constant, irreducible factor of $P(\mathrm{H})$, called the* essential *factor of P.*
2. *$\Psi_{\mathrm{H}}(P) \sim \Psi_{\mathrm{H}}(Q)$ iff $P \sim Q$.*
3. *If $\Psi_{\mathrm{H}}(P) \mid Q(\mathrm{H})$ then $\Psi_{\mathrm{H}}(P) \mid \Psi_{\mathrm{H}}(Q)$.*

Let us discuss the definition. Let $P \in \text{fact}(\mathcal{C})$ be an irreducible factor of some $F \in \mathcal{C}$. The intuition is that the essential factor $\Psi_{\mathrm{H}}(P)$ itself should contain "enough" information about P and cannot appear as a factor of any other polynomial $Q \in \text{fact}(\mathcal{C})$. The next lemma shows that our definition is sufficient in achieving our original goal. That is, ensuring that two distinct (composite) polynomials $F, R \in \mathcal{C}$ remain distinct.

Lemma 3.2 (Uniqueness from essential factorization)**.** *Let $F, R \in \mathcal{C}$ be two polynomials (not necessarily irreducible) and let* H *be as in the above definition. Then $F \equiv R$ iff $F(\mathrm{H}) = R(\mathrm{H})$.*

Proof. The proof is by induction on $\deg(F) + \deg(R)$. The base case is when both F and R are constant polynomials and the claim clearly follows. Now suppose wlog that F is non-constant. By the properties of H, $F(\mathrm{H})$ is also non-constant and since $F(\mathrm{H}) = R(\mathrm{H})$, R must be non-constant as well. Let $F = P_1 \cdot \ldots \cdot P_k$ and $R = Q_1 \cdot \ldots \cdot Q_\ell$ denote F's and R's factorization into irreducible factors (possibly with repetitions), respectfully where $P_i, Q_j \in \text{fact}(\mathcal{C})$. We have that:

$$P_1(\mathrm{H}) \cdot \ldots \cdot P_k(\mathrm{H}) = F(\mathrm{H}) = R(\mathrm{H}) = Q_1(\mathrm{H}) \cdot \ldots \cdot Q_\ell(\mathrm{H}).$$

By definition, $\Psi_{\mathrm{H}}(P_1) \mid P_1(\mathrm{H})$. Therefore, by uniqueness of factorization, there exist $j \in [\ell]$ such that $\Psi_{\mathrm{H}}(P_1) \mid Q_j(\mathrm{H})$, since $\Psi_{\mathrm{H}}(P_1)$ is an irreducible polynomial. Combining Properties 2 and 3, we get that $P_1 = \alpha Q_j$ for some $\alpha \neq 0 \in \mathbb{F}$. Now, consider: $F' \triangleq \frac{F}{P_1}$ and $R' \triangleq \frac{R}{\alpha Q_j}$. It follows that $F'(\mathrm{H}) = R'(\mathrm{H})$ when $\deg(F') + \deg(R') < \deg(F) + \deg(R)$. By the induction hypothesis $F' \equiv R'$ and thus $F = P' \cdot P_1 \equiv Q' \cdot \alpha Q_j = R$. □

3.1 Essential Factorization Schemes for Multilinear Polynomials

In this section, we show how to construction essential factorization schemes for classes of multilinear polynomials that admit efficient identity testing algorithms. In fact, if we want to apply our results for a class $\mathcal{C}$, we require algorithms for a somewhat larger class.

Let $\mathcal{C}$ be a class of multilinear polynomials over the field $\mathbb{F}$. From Lemma 2.3, it follows that $\text{fact}(\mathcal{C}) = \mathcal{C}$. Let $\mathcal{G}_n(\bar{w}) \triangleq \left(\mathcal{G}_n^1(\bar{w}), \ldots, \mathcal{G}_n^n(\bar{w})\right)$ be a generator for polynomials of the form $D_i(P,Q)$ where $P, Q \in \mathcal{C}$ are irreducible, n-variate polynomials and $i \in [n]$. We show that the map $\mathrm{H}_n \triangleq \mathcal{G}_n(\bar{w}) + G_{n,2}(y_1, y_2, z_1, z_2)$ is an essential factorization scheme for $\mathcal{C}$. As was mentioned earlier, this construction demonstrates another connection between polynomial factorization and polynomial identity testing. We begin by specifying Ψ_{H}. To that end, we require the following definition:

Definition 3.3 (Reviving)**.** *Let $i \in [n]$. We call the operation:*

$$\mathcal{R}_{x_i}(P(\mathrm{H}_n)) \triangleq P(\mathrm{H}_n)|_{y_2=a_i, z_2=x_i-\mathcal{G}^i, z_1=0}$$

a revival of x_i. *By Proposition 2.10, the result of such a revival equals:*

$$P(\mathcal{G}_n^1(\bar{w}), \ldots, \mathcal{G}_n^{i-1}(\bar{w}), x_i, \mathcal{G}_n^{i+1}(\bar{w}), \ldots, \mathcal{G}_n^n(\bar{w}))$$

which can be seen as lifting the polynomial map substituted into x_i. In other words, $\mathcal{R}_{x_i}(P(\mathrm{H}_n)) = P_i(\mathcal{G}_n) \cdot x_i + P_0(\mathcal{G}_n)$ when $P = P_i x_i + P_0$. In fact, we can revive two variables x_i, x_j at a time by considering $\mathcal{R}_{x_i,x_j}(P(\mathrm{H}_n)) \triangleq P(\mathrm{H}_n)|_{y_2=a_i, z_2=x_i-\mathcal{G}_n^i, y_1=a_j, z_1=x_j-\mathcal{G}_n^j}$.

We can now formally define essential factors.

Definition 3.4 (Essential Factor)**.** *Let $P(\mathrm{H}_n) = f_1 \cdot f_2 \cdots f_m$ be the unique factorization of $P(\mathrm{H}_n)$ into irreducible factors. We define $\Psi_{\mathrm{H}}(P)$, the* essential factor *of P, as f_e such that for each $x_i \in \mathrm{var}(P)$, we have that $x_i \in \mathrm{var}(\mathcal{R}_{x_i}(f_e))$. To avoid ambiguity we take $\Psi_{\mathrm{H}}(P)$ to be the normalized[2] f_e.*

First, observe that applying $\mathcal{R}_{x_i}$ on $P(\mathrm{H}_n)$ results in applying $\mathcal{R}_{x_i}$ on each f_ℓ. Therefore since P is a multilinear polynomial there can be at most one factor f_e that depends on x_i, when revived. Consequently, there can be at most one factor f_e with the required property. However, this still does not guarantee an existence of such a factor. We will show that such a factor always exists.

Lemma 3.5. *Let $P \in \text{fact}(\mathcal{C}) = \mathcal{C}$ be an irreducible polynomial. Then $\Psi_{\mathrm{H}}(P)$ is well-defined.*

Proof. As previously, let $P(\mathrm{H}_n) = f_1 \cdot f_2 \cdots f_m$ be the unique factorization of $P(\mathrm{H}_n)$ into irreducible factors. First, we claim that for each $x_i \in \mathrm{var}(P)$ there exists $k_i \in [m]$ such that $x_i \in \mathrm{var}(\mathcal{R}_{x_i}(f_{k_i}))$. By definition $\mathcal{R}_{x_i}(P(\mathrm{H}_n)) = P_i(\mathcal{G}_n) \cdot x_i + P_0(\mathcal{G}_n)$ when $P = P_i x_i + P_0$. Since $P_i = D_i(P, 1)$ we get that the map $\mathcal{G}_n$ hits P_i which implies that $P(\mathrm{H}_n)$ depends on x_i, and the claim follows. To finish the proof, we need to show that $k_i = k_j$ for all $x_j, x_i \in \mathrm{var}(P)$.

Assume for a contradiction and wlog that $k_1 = 1, k_2 = 2$. As P is an irreducible polynomial, $\Delta_{12}(P) \not\equiv 0$ by Lemma 2.6. By Observation 2.7, the map $\mathcal{G}_n$ hits $\Delta_{12}(P)$. In other words, there

[2]The coefficient of the largest monomial according to the lexicographic order in 1.

exists $\bar{\beta} \in \mathrm{Im}\,(\mathcal{G}_n)$ such that $\Delta_{12}(P)(\bar{\beta}) \neq 0$ and $P(x_1, x_2, \beta_3, \ldots, \beta_n)$ depends of x_i and x_j. Let $\bar{\gamma} \in \mathcal{G}^{-1}(\bar{\beta})$. Consider

$$\tilde{P} \triangleq \mathcal{R}_{x_1,x_2}(P(\mathrm{H}_n))|_{\bar{w}=\bar{\gamma}} = P(x_1, x_2, \mathcal{G}_n^3(\bar{\gamma}), \ldots, \mathcal{G}_n^n(\bar{\gamma})) = P(x_1, x_2, \beta_3, \ldots, \beta_n).$$

By the choice of $\bar{\beta}$, the LHS depends on both x_i and x_j. On the other hand,

$$P(x_1, x_2, \beta_3, \ldots, \beta_n) = \tilde{P} = f_1(x_1, x_2, \beta_3, \ldots, \beta_n) \cdot f_2(x_1, x_2, \beta_3, \ldots, \beta_n) \cdots f_m(x_1, x_2, \beta_3, \ldots, \beta_n)$$

so $x_i \in \mathrm{var}(f_i)$ for $i = 1, 2$ and by Lemma 2.6 $\Delta_{12}(P)(\bar{\beta}) = 0$, thus reaching a contradiction. □

As was established, Ψ_{H} is well-defined and satisfies Property 1 of Definition 3.1. Note that given a list of purported factors it is easy to identify the essential ones by reviving one variable at a time and testing dependence. Since $P(\mathrm{H})$ is a $t(n)$-variate polynomial of polynomial degree and typically $t(n) \ll n$, dependence testing can be carried out efficiently by a computing the monomial expansion of $P(\mathrm{H})$. This also takes care of Property 3. We turn to showing that Ψ_{H} satisfies Property 2. The intuition is that the essential factor $\Psi_{\mathrm{H}}(P)$ should contain all the information about P since $\Psi_{\mathrm{H}}(P)$ encapsulates in itself all the variables of P.

Lemma 3.6. *Let $P, Q \in \mathcal{C}$ be two irreducible polynomials. Then $\Psi_{\mathrm{H}}(P) \sim \Psi_{\mathrm{H}}(Q)$ iff $P \sim Q$.*

Proof. The first direction is trivial. For the other direction note that since $\Psi_{\mathrm{H}}(P)$ and $\Psi_{\mathrm{H}}(Q)$ are both normalized we actually have that $f \triangleq \Psi_{\mathrm{H}}(P) = \Psi_{\mathrm{H}}(Q)$. In other words, $P(\mathrm{H}_n) = f \cdot P'$ and $Q(\mathrm{H}_n) = f \cdot Q'$. For $x_i \in \mathrm{var}(P)$, we can write: $P = P_i x_i + P_0$, $Q = Q_i x_i + Q_0$. Consider the revival of x_i in both $P(\mathrm{H}_n)$ and $Q(\mathrm{H}_n)$. By the definition of the essential factor $x_i \in \mathrm{var}(\mathcal{R}_{x_i}(f))$. Therefore:

$$\mathcal{R}_{x_i}(P(\mathrm{H}_n)) = (P_i(\mathcal{G}_n) \cdot x_i + P_0(\mathcal{G}_n)) = (\hat{f}_i x_i + \hat{f}_0) \cdot \mathcal{R}_{x_i}(P')$$
$$\mathcal{R}_{x_i}(Q(\mathrm{H}_n)) = (Q_i(\mathcal{G}_n) \cdot x_i + Q_0(\mathcal{G}_n)) = (\hat{f}_i x_i + \hat{f}_0) \cdot \mathcal{R}_{x_i}(Q')$$

where $\mathcal{R}_{x_i}(f) = \hat{f}_i x_i + \hat{f}_0$. By setting $x_i = 0$ we obtain:

$$P_i(\mathcal{G}_n) = \hat{f}_i \cdot \mathcal{R}_{x_i}(P')\ ,\ P_0(\mathcal{G}_n) = \hat{f}_0 \cdot \mathcal{R}_{x_i}(P')$$
$$Q_i(\mathcal{G}_n) = \hat{f}_i \cdot \mathcal{R}_{x_i}(Q')\ ,\ Q_0(\mathcal{G}_n) = \hat{f}_0 \cdot \mathcal{R}_{x_i}(Q').$$

And hence: $D_i(P, Q)(\mathcal{G}_n) = P_i(\mathcal{G}_n) \cdot Q_0(\mathcal{G}_n) - Q_i(\mathcal{G}_n) \cdot P_0(\mathcal{G}_n) \equiv 0$. Since $\mathcal{G}_n$ hits $D_i(P, Q)$ we know that $D_i(P, Q) \equiv 0$ to begin with. As P, Q are both irreducible, $P \sim Q$ by Lemma 2.2. □

The following theorem summarizes this section.

Theorem 3.7. *Let $\mathcal{C}$ be a class of multilinear polynomials over the field $\mathbb{F}$ and let $\mathcal{G}_n(\bar{w})$ be a generator for the polynomials of the form $D_i(P, Q)$ where $P, Q \in \mathcal{C}$ are irreducible, n-variate polynomials and $i \in [n]$. Then $\mathrm{H}_n \triangleq \mathcal{G}_n(\bar{w}) + G_{n,2}(y_1, y_2, z_1, z_2)$ is an essential factorization scheme for $\mathcal{C}$. And in particular, for all $F, R \in \mathcal{C}$: $F \equiv R \iff F(\mathrm{H}_n) = R(\mathrm{H}_n)$.*

4 Factoring Sparse Multilinearly-Split Polynomials

In this section we prove our main result - Theorem 1. First, we give the outline of the proof. We say that a set $\mathcal{H}$ is an *interpolating set* for a class $\mathcal{C}$ if for every $P \in \mathcal{C}$ the evaluations $P|_{\mathcal{H}}$ determine P uniquely. In particular, an interpolating set can serve as a hitting set since $P \equiv 0 \iff P|_{\mathcal{H}} \equiv 0$.

Let $\mathcal{H}$ be the interpolating set for sparse polynomials given by Lemma 4.4. Our plan is to evaluate each essential factor separately on $\mathcal{H}$ and then apply the reconstruction algorithm of Lemma 4.4 to recover the original factors. However, there are couple of obstacles that stand in our way. First of all, how do we get access to every essential factor separately? To overcome this obstacle, we use $\mathcal{H}$ in conjunction with Theorem 3.7. Observe that $\mathcal{H}$ hits polynomials of the form $D_i(P, Q)$ where P and Q are sparse. Therefore, it satisfies the conditions of Theorem 3.7 (invoking Lemma 2.8). We then invoke Lemma 4.2 to factor our polynomial. As the new number of variables is small, this step can be carried out efficiently. This leads us to a second obstacle: we only obtain evaluation of the essential factors rather than there original factors.

Although by definition the essential factors contain "enough" information, this information might still be insufficient for the reconstruction algorithm since in order to reconstruct a sparse polynomial P the algorithm requires the values of P on $\mathcal{H}$ while we only have the values of a factor of P at hand. For the second obstacle, we make our reconstruction algorithm more "resilient" to information loss by extedning it to handle rational functions (Lemma 4.3).

We now move to the formal proof. To this end, we require the following results. The first result states that a multilinear factor of sparse polynomial is itself a sparse polynomial. Example 1.1 demonstrates that this is not the case for general sparse polynomials.

Lemma 4.1 ([GKL12])**.** *Let $0 \not\equiv P, Q \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be polynomials such that P is multilinear. Then $P \mid Q \implies \|P\| \leq \|Q\|$.*

The next result which is implicit in many factorization algorithms, exhibits an efficient factorization algorithm for certain regime of parameters. In particular, for polynomials with constantly-many variables and a polynomial degree. We note that this the state-of-the-art algorithm for this regime of parameters.

Lemma 4.2 (Implicit [GG99, Kal89])**.** *There exists a deterministic algorithm that given a t-variate, degree d polynomial P over $\mathbb{F}$ outputs its irreducible multilinear factors. The running time of the algorithm is $(d, p, \ell)^{\mathcal{O}(t)}$ when $\mathbb{F} = \mathbb{F}_{p^\ell}$ and $(d, b)^{\mathcal{O}(t)}$ when $\mathbb{F} = \mathbb{Q}$ and b is the bit complexity of the coefficients in P.*

The following result converts a reconstruction algorithm for sparse polynomials into a reconstruction algorithm for sparse rational functions, introducing only a polynomial overhead.

Lemma 4.3 ([CL11])**.** *Let A be a deterministic algorithm that can reconstruct a s-sparse polynomial $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of degree d in time $T(n, s, d, |\mathcal{H}|)$ given the evaluations $P|_{\mathcal{H}}$. Let $R, Q \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be two coprime s-sparse polynomials of degree d and $\bar{\sigma} \in \mathbb{F}^n$ such that $Q(\bar{\sigma}) \neq 0$. Finally, let $V \subseteq \mathbb{F}$ be a subset of size $2d$. Then there exists a deterministic algorithm B that given the evaluations $(R/Q)|_{V \cdot \mathcal{H} + \sigma}$ outputs R', Q' such that $R' = cR$ and $Q' = cQ$ for some $c \neq 0 \in \mathbb{F}$ in time $\mathrm{poly}(|\mathcal{H}|, n, d, T(n, s, d, |\mathcal{H}|))$ and uses the algorithm A as an oracle. If $Q(\bar{\sigma}) = 0$ the algorithm fails.*

We conclude the list with an efficient reconstruction algorithm for sparse polynomials.

Lemma 4.4 ([KS01])**.** *Let $n, s, d > 1$. There exists a deterministic algorithm that in time* $\mathrm{poly}(n, s, d)$ *outputs an interpolating set $\mathcal{H}$ such that given the evaluations $P|_{\mathcal{H}}$ of a s-sparse polynomial $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of degree d in time* $\mathrm{poly}(n, s, d)$ *the algorithm can reconstruct P.*

We are ready to proceed with the proof of our main theorem (Theorem 1). Our algorithm combines the above results. The description of the algorithm is given in Algorithm 1.

Input: s-sparse, multilinearly-split polynomial $F \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of degree d
Output: The irreducible factors of F

Choose a subset $\{1\} \in V \subseteq \mathbb{F}$ of size $2d$;
Invoke the algorithm in Lemma 4.4 with $n, 2s^2, d = 2n$ to obtain an interpolating set $\mathcal{H}$;
Apply Lemma 2.8 on $\mathcal{H}' = V \cdot \mathcal{H}$ to obtain the map $\mathcal{G}$ `/* note that` $\mathcal{H} \subseteq \mathcal{H}' \subseteq \mathrm{Im}(\mathcal{G})$ `*/`
Set $\mathrm{H}_n \stackrel{\Delta}{=} \mathcal{G}(\bar{u}) + \mathcal{G}(\bar{w}) + G_{n,2}(\bar{y}, \bar{z}))$;
Use Lemma 4.2 to Factor $P(\mathrm{H}_n)$. Let S be the set of the irreducible factors ;
Initialize $E \leftarrow \emptyset$ `/* The set of all the essential factors */`
foreach $f \in S, i \in [n]$ **do**
  if $x_i \in \mathrm{var}(\mathcal{R}_{x_i}(f))$ `/* Check by looking at the monomial expansion */`
  then
    $E \leftarrow E \cup \{(\mathcal{R}_{x_i}(f), i)\}$
`/* Reconstruct the original factors */`
foreach $(\hat{f}, i) \in E, \bar{\beta} \in \mathcal{G}^{-1}(\mathcal{H})$ **do**
  Set: $\hat{f}_0(\bar{u}, \bar{w}) \stackrel{\Delta}{=} \hat{f}|_{x_i=0}, \hat{f}_i(\bar{u}, \bar{w}) \stackrel{\Delta}{=} \hat{f}|_{x_i=1} - \hat{f}_0$;
  Apply Lemma 4.3 jointly with the reconstruction algorithm from Lemma 4.4 on $\hat{f}_0(\bar{u}, \bar{\beta})/\hat{f}_i(\bar{u}, \bar{\beta})$ to obtain R', Q'. ;
  On a success, output $P = Q' \cdot x_i + R'$;

Algorithm 1: Factoring algorithm for sparse multilinearly-split polynomials.

Proof. (of Theorem 1) We analyze Algorithm 1. For the running time, we get $(n, s, d, p, \ell)^{\mathcal{O}(t)}$ when $\mathbb{F} = \mathbb{F}_{p^\ell}$ and $(n, s, d, b)^{\mathcal{O}(t)}$ when $\mathbb{F} = \mathbb{Q}$. By Lemmas 2.8 and 4.4 $t = \mathcal{O}(\log_n |\mathcal{H}|) = \mathcal{O}(\log_n(nsd))$. Therefore, if all the parameters are $\mathrm{poly}(n)$ we get the claimed running time.

We now move to the correctness. Let $F = P_1 \cdot P_2 \cdot \ldots \cdot P_m$ be F's factorization into irreducible factors (possibly with repetitions). By Lemma 4.1, each P_j above is s-sparse. First, observe that $\mathcal{H}$ (and consequently $\mathcal{H}'$) is a hitting set for $D_i(P, Q)$ where $P, Q \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ are s-sparse multilinear polynomials. By Lemma 2.8, the map $\mathcal{G}(\bar{u})$ hits those polynomials. As $\mathcal{G}(\bar{0}) = \bar{0}$, the same holds true for $\mathcal{G}(\bar{u}) + \mathcal{G}(\bar{w})$ as well. By Theorem 3.7, H_n is an essential factorization scheme for s-sparse multilinear polynomials. Therefore, by the properties of Definition 3.1 each $\Psi_{\mathrm{H}}(P_j)$ is a non-constant factor of $F(\mathrm{H})$ and $\Psi_{\mathrm{H}}(P_j) \sim \Psi_{\mathrm{H}}(P_k)$ iff $P_j \sim P_k$. Therefore, we can access all $\Psi_{\mathrm{H}}(P_j)$-s by factoring $F(\mathrm{H})$ and reviving one variable at a time to distinguish essential factors from the non-essential ones. Now, let $f = \Psi_{\mathrm{H}}(P_j)$ and $P_j(\mathrm{H}_n) = f \cdot P'_j$. By repeating the reasoning in the proof of Lemma 3.6 we get:

$$[P_j]_i(\mathcal{G}(\bar{u}) + \mathcal{G}(\bar{w})) \cdot x_i + [P_j]_0(\mathcal{G}(\bar{u}) + \mathcal{G}(\bar{w})) = \mathcal{R}_{x_i}(P_j(\mathrm{H}_n)) =$$
$$\hat{f}(\bar{u}, \bar{w}) \cdot \mathcal{R}_{x_i}(P'_j) = (\hat{f}_i(\bar{u}, \bar{w})x_i + \hat{f}_0(\bar{u}, \bar{w})) \cdot \mathcal{R}_{x_i}(P'_j)$$

and hence

$$[P_j]_i(\mathcal{G}(\bar{u}) + \mathcal{G}(\bar{w})) = \hat{f}_i(\bar{u}, \bar{w}) \cdot \mathcal{R}_{x_i}(P'_j)$$

$$[P_j]_0(\mathcal{G}(\bar{u}) + \mathcal{G}(\bar{w})) = \hat{f}_0(\bar{u}, \bar{w}) \cdot \mathcal{R}_{x_i}(P'_j).$$

when $P_j = [P_j]_i \cdot x_i + [P_j]_0$. Since $[P_j]_i$ is a non-zero s-sparse polynomial, there exists $\bar{\sigma} \in \mathcal{H}$ such that $[P_j]_i(\sigma) \neq 0$. By Lemma 2.8 we can efficiently iterate over $\mathcal{H}$ to find $\bar{\beta} \in \mathcal{G}^{-1}(\sigma)$. Finally observe that

$$\hat{f}_0(\bar{u}, \bar{\beta}) / \hat{f}_i(\bar{u}, \bar{\beta}) = [P_j]_0(\mathcal{G}(\bar{u}) + \bar{\sigma}) \;/\; [P_j]_i(\mathcal{G}(\bar{u}) + \bar{\sigma}).$$

Therefore given access to $\hat{f}_0(\bar{u}, \bar{\beta}) / \hat{f}_i(\bar{u}, \bar{\beta})$ the algorithm can query the polynomial $[P_j]_0/[P_j]_i$ on every point of the forms $V \cdot \mathcal{H} + \sigma$ as required by Lemma 4.3. Consequently, we can apply Lemma 4.3 jointly with the reconstruction algorithm from Lemma 4.4 to obtain $R' = c[P_j]_0, Q' = c[P_j]_i$ resulting in $P = Q' \cdot x_i + R' = c[P_j]_i \cdot x_i + c[P_j]_0 = cP_j$ and we are done. □

5 Conclusions and Remarks

In this paper we give the first factorization algorithm for sparse polynomials that split into multilinear factors. The key ingredient in the algorithm is the Essential Factorization Schemes. We hope that these schemes could be applied to handle richer classes of sparse polynomials.

A natural question to ask is whether it would possible to extend the algorithm to compute multilinear factors of an arbitrary sparse polynomial. Another open question is to improve the dependence on the characteristic from polynomial to polylogarithmic.

On a final note, Example 5.1 in [GK85] is followed by a question (quote): "Can the output size for the factoring problem be actually more than quasi-polynomial in the input size?" Our next example provides a positive answer to this question over fields with super-polylogarithmic characteristics.

Example 5.1. *Let $p = 2k - 1$ be an odd prime, $\mathbb{F} = \mathbb{F}_{p^\ell}$ and $n, \ell \geq 1$. Consider the polynomial $f(\bar{x}) = (x_1 + x_2 + \ldots + x_n)^{p+1}$ which can be written as a square of $g(\bar{x}) = (x_1 + x_2 + \ldots + x_n)^k$. Observe that $f(\bar{x}) = (x_1^p + x_2^p + \ldots + x_n^p) \cdot (x_1 + x_2 + \ldots + x_n)$ and therefore $\|f\| \leq n^2$. On the other hand, $\|g\| = \binom{n+p/2-1}{p/2} = \Omega\left((\frac{n+p}{p})^p + (\frac{n+p}{n})^n\right)$.*

References

[BOT88] M. Ben-Or and P. Tiwari. A deterministic algorithm for sparse multivariate polynominal interpolation. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 301–309, 1988.

[CL11] A. M. Cuyt and W. Lee. Sparse interpolation of multivariate rational functions. *Theor. Comput. Sci.*, 412(16):1445–1456, 2011.

[Gat06] J. von zur Gathen. Who was who in polynomial factorization:. In *ISSAC*, page 2, 2006.

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

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

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

[GKL04] S. Gao, E. Kaltofen, and A. G. B. Lauder. Deterministic distinct-degree factorization of polynomials over finite fields. *J. Symb. Comput.*, 38(6):1461–1470, 2004.

[GKL12] A. Gupta, N. Kayal, and S. V. Lokam. Reconstruction of depth-4 multilinear circuits with top fanin 2. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, pages 625–642, 2012. Full version at http://eccc.hpi-web.de/report/2011/153.

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

[Kal85] E. Kaltofen. Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization. *SIAM J. on computing*, 14(2):469–489, 1985.

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

[Kal03] E. Kaltofen. Polynomial factorization: a success story. In *ISSAC*, pages 3–4, 2003.

[Kay07] N. Kayal. *Derandomizing some number-theoretic and algebraic algorithms.* PhD thesis, Indian Institute of Technology, Kanpur, India, 2007.

[KI04] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[KK05] E. Kaltofen and P. Koiran. On the complexity of factoring bivariate supersparse (lacunary) polynomials. In *ISSAC*, pages 208–215, 2005.

[KMSV13] Z. S. Karnin, P. Mukhopadhyay, A. Shpilka, and I. Volkovich. Deterministic identity testing of depth 4 multilinear circuits with bounded top fan-in. *SIAM J. on Computing*, 42(6):2114–2131, 2013.

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

[KSS14] S. Kopparty, S. Saraf, and A. Shpilka. Equivalence of polynomial identity testing and deterministic multivariate polynomial factorization. In *Proceedings of the 29th Annual IEEE Conference on Computational Complexity (CCC)*, pages 169–180, 2014.

[KT90] E. Kaltofen and B. M. Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *J. of Symbolic Computation*, 9(3):301–320, 1990.

[Sho91] V. Shoup. A fast deterministic algorithm for factoring polynomials over finite fields of small characteristic. In *ISSAC*, pages 14–21, 1991.

[SSS13] C. Saha, R. Saptharishi, and N. Saxena. A case of depth-3 identity testing, sparse factorization and duality. *Computational Complexity*, 22(1):39–69, 2013.

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

[SV09] A. Shpilka and I. Volkovich. Improved polynomial identity testing for read-once formulas. In *APPROX-RANDOM*, pages 700–713, 2009. Full version at http://eccc.hpi-web.de/report/2010/011.

[SV10] A. Shpilka and I. Volkovich. On the relation between polynomial identity testing and finding variable disjoint factors. In *Automata, Languages and Programming, 37th International Colloquium (ICALP)*, pages 408–419, 2010. Full version at http://eccc.hpi-web.de/report/2010/036.

[SV11] S. Saraf and I. Volkovich. Blackbox identity testing for depth-4 multilinear circuits. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 421–430, 2011. Full version at http://eccc.hpi-web.de/report/2011/046.

[SV14] A. Shpilka and I. Volkovich. On reconstruction and testing of read-once formulas. *Theory of Computing*, 10:601–648, 2014.

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

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

ECCC ISSN 1433-8092

http://eccc.hpi-web.de